School of Chemical and Petroleum Engineering

In-situ Investigation of the Oil-Water Interface Under Dynamic Conditions

Anita Evelyn Hyde

This thesis is presented for the Degree of
Doctor of Philosophy
of
Curtin University

June 2018

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgement has been made. This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

June 19, 2018

Anita Hyde

To family and friends who always believed that I would finish it,
and to my supervisor, who made sure that the journey was worthwhile.
Thank you for everything.

This research was conducted with the support of an Australian Government Research Training Program Scholarship and a Monbukagakusho Scholarship. My sincere thanks to staff and colleagues at Curtin University who made this research possible, and to new friends and colleagues in Japan for their unwavering welcome and support.

Abstract

Liquid-liquid systems are encountered extensively in industry. Oil-water systems, in particular, are extremely common from petroleum processing through to cosmetics and food production. However, measurement of the interfacial tension of liquid-liquid systems can be particularly difficult. Complicated correction factors are required to measure liquid-liquid systems using force methods, and drop-shape methods can be beset by pumping and stability issues. Both types of methods require static, vibration-free environments and often additional complex machinery. While the method that underlies the pendant drop technique (ADSA-P) is in theory applicable to any axisymmetric fluid body, to date, drop-shape fitting is used predominately with variations of pendant and sessile drops, and a variant has been produced for liquid bridges. This research focuses on the application of the drop-shape methodology to the final axisymmetric fluid interface – the holm meniscus.

Unlike pendant and sessile drops, which are highly suited to surface tension measurement but can be difficult to use with liquid-liquid systems, the holm interface is stable, easily produced and easily maintained. A program was written in MATLAB to calculate the interfacial tension from images of a holm meniscus using the drop shape-fitting methodology. The technique is better suited to the measurement of liquid-liquid- than to liquid-vapour- systems (surface tension), and measurement of the equilibrium oil-water interfacial tension between hexadecane and water was demonstrated with an error comparable to what is reported for ADSA–P (pendant drop). The technique was successfully used to measure the tension of a silicone-water interface with a density difference of less than 3%, which makes for erroneous fitting with pendant and sessile drops due to poor deformation, and where pumping is made difficult by high viscosity.

The program accepts multi-frame inputs, facilitating the measurement of time-dependent systems. The possibilities of using this technique for dynamic analysis were highlighted when it was used to measure changes to the interfacial tension of the oil-water interface inside of an operating microwave reactor, where vibrations and convection currents limit the application of other methods. The technique was also used to measure the influence of a magnetic field on magnetic surfactants, where the magnetic field applies a force directly to the drop. Lastly, the effects of pH on the dynamic interfacial tension of fatty acids was predicted using a simple model. The stability of the holm meniscus allowed for dynamic measurements on a single large interface over long periods of time without introducing movement in the sample, demonstrating the ability to measure *in situ* the effect of reactions on interfacial tension.

This technique provides a method to measure changes to the bulk interface in situations that are unsuited to current techniques due to interface instability or otherwise hindered environments. Due to its stable and compact nature, the technique facilitates continuous measurements of interfacial tension in systems that mimic industrial environments more closely, and has the potential to be adapted for *in situ* measurement in real environments.

Publication list

Publications associated with this thesis

- A. Hyde; Phan, C.; Ingram, G.; Determining liquid–liquid interfacial tension from a submerged meniscus, Colloids Surfaces A Physiochem. Eng. Asp. 459 (2014) 267–273.
- A. Hyde; Horiguchi, M.; Minamishima, N.; Asakuma, Y.; Phan, C;. Effects of microwave irradiation on the decane–water interface in the presence of Triton X–100. Colloids Surfaces A Physiochem. Eng. Asp. 524 (2017) 178–184.
- Hyde, A.; Phan, C.; Yusa, S.; Dynamic interfacial tension of nonanoic acid/ hexadecane/water system in response to pH adjustment. Colloids Surfaces A Physiochem. Eng. Asp. (2018) *[In press – Accepted manuscript]*.

Additional publications

- A. Hyde; Fujii, S.; Sakurai, K.; Phan, C.; Yusa, S.; Concentration-dependent aggregation behavior of asymmetric cationic surfactant hexyldimethyloctylammonium bromide, Chem. Lett. 46 (2017) 271–273.
- Y. Ohara; Kawata, Y.; Hyde, A.; Phan, C.; Takeda, R.; Takemura, Y.; Yusa, S.;. Preparation of a magnetic-responsive polycation with a tetrachloroferrate anion. Chem. Lett. 2017, 1473–1475.
- S. Badban; Hyde, A. E.; Phan, C. M; Hydrophilicity of nonanoic acid and its conjugate base at the air/water interface. ACS Omega 2017, 2 (9), 5565–5573.

Publications in progress

- A. Hyde; Ohshio, M.; Nguyen, C.; Yusa, S.; Yamada, N.; Phan, C.; Surface composition of the ethanol/water mixture.
- A. Hyde; Minamishima, N.; Shibata, Y.; Asakuma, Y.; Phan, C.; Mechanism of interfacial tension reduction during microwave irradiation by changing concentration and length of Triton X.
- A. Hyde; Phan, C.; Rowles, M.; Modelling the interfacial adsorption of hexadecylethyldimethylammonium bromide.

Contents

List of Figures xxi

List of Tables xxv

1 Introduction 1

1.1 Overview 1

1.1.1 Novelty 3

1.1.2 Structure of this thesis 4

I An introduction to interfacial tension 5

2 Background 9

2.1 The basis of interfacial tension 9

2.2 Pressure discontinuity across the interface 10

2.2.1 The Young-Laplace equation 10

2.2.2 Excess pressure accross an interface 12

2.2.3 Capillary length and gravitational effects 12

2.2.4 Distortion in a gravitational field: Axisymmetric menisci 13

2.2.5 Describing the shape of axisymmetric menisci 13

2.2.6 Dimensionless systems and reducing factors 17

2.3 Surface chemistry: impurities and surface-active molecules . . . 18
2.3.1 "Real systems": surfactants and impurities in industrial processing . . . 18
2.3.2 Impurities . . . 19
2.3.3 Interfacial behaviour of surfactants . . . 19
2.3.4 Surfactant use with emulsions . . . 23
2.3.5 Equilibration and dynamic systems . . . 23

3 Existing methods for the measurement of interfacial tension 25

3.1 Chapter overview . . . 25
3.2 Force-based methods . . . 25
3.2.1 The Wihelmy Plate Method . . . 26
3.2.2 The Du Nuöy Ring . . . 26
3.2.3 Calibration and modifications for liquid-liquid measurement (Du Nuöy Ring and Wilhelmy Plate methods) . . . 27
3.2.4 The drop-volume/drop-weight methods . . . 28
3.2.5 Limitations of the force methods . . . 28
3.3 Shape based methods . . . 29
3.3.1 Axisymmetric drop shape analysis (ADSA) . . . 30
3.3.2 The pendant drop technique (ADSA-P) . . . 31
3.3.3 Modifications to ADSA for specific applications . . . 33
3.3.4 Theoretical Image Fitting Analysis (TIFA) . . . 34
3.3.5 Issues with drop-shape methods . . . 34
3.4 Other methods for specific applications . . . 34
3.4.1 Systems with low Bond numbers . . . 34

3.4.2 The spinning drop method 35
3.4.3 The Capillary rise method 36
3.4.4 The maximum bubble pressure method 36
3.4.5 Analysis of the capillary rise profile around a cylinder (ACRPAC) 36
3.5 Where the niche is 37

4 The holm meniscus **39**
4.1 Overview 39
4.2 Early work 40
4.2.1 Tabulated solutions 40
4.2.2 Early attempts at integration 41
4.2.3 Integrating using Bessel functions 42
4.3 Axisymmetric fluid bodies with one asymptote 43
4.4 Modeling the meniscus 44
4.4.1 Development of an initial boundary problem 44
4.4.2 A new shape fitting technique 45
4.4.3 Choice of the submerged solid 47
4.5 Chapter summary 47

II Coding and technique development **49**

5 Image acquisition **53**
5.1 Overview 53
5.2 Experimental technique 54
5.2.1 General setup 54
5.2.2 Scaling 54

5.2.3 Additional experimental parameters 55
5.3 Choice of solid 56
5.3.1 Solid shape 56
5.3.2 Sphere size 56
5.3.3 Sphere material 56
5.4 Choice of cell 57
5.4.1 Cell shape 57
5.4.2 Cell size 57
5.4.3 Cell material 58
5.5 Improving the measurement accuracy 59
5.5.1 Curvature of the interface 59
5.5.2 Adjusting the penetration depth 60
5.5.3 Impurities 61
5.6 Image quality 61
5.6.1 Lighting 61
5.6.2 Image resolution 62
5.6.3 Length of the holm 62
5.7 Adaption for specific applications 63
5.7.1 Coloured or opaque solutions 63
5.7.2 Dynamic (multiframe) analysis 63
5.8 Chapter summary 64

6 Applying the principals of drop-shape analysis **65**

6.1 Chapter overview 65
6.2 Program overview 65
6.3 Machine requirements 67

6.4 Program feeds 68
6.5 Defining areas of interest 68
6.6 Edge detection and image analysis 70
6.7 The spherical profile and contact line 70
6.7.1 Fitting the spherical profile 71
6.7.2 Adjusting the image angle 71
6.7.3 Identifying the contact line 72
6.7.4 Initial coordinates (contact point) 73
6.8 Theoretical profiles 74
6.8.1 Integrating the Young-Laplace equation 74
6.8.2 Trimming the theoretical curve to the feasible region 75
6.9 Numerical optimisation 75
6.9.1 Calculating the fitting error 76
6.9.2 Optimisation constraints 76
6.9.3 Calculation of interfacial tension 77
6.9.4 Random coordinate selection and repeat analysis 77
6.9.5 Initial filtering and error analysis 78
6.10 Angle adjustment 78
6.10.1 Determining the tilt angle 78
6.10.2 Left- and right-side fitting 80
6.11 Adaption for dynamic (multiframe) analysis 81
6.11.1 Code adjustments 81
6.11.2 Multiframe filtering (Excel) 81
6.12 Chapter summary 82

7 Image Analysis **83**

7.1 Introductory remarks 83

7.2 Edge detection - evolution from thresholding to gradient methods . . . 84

7.2.1 Image thresholding 84

7.2.2 Gradient edge detection 85

7.3 Complex and noisy images 86

7.3.1 Determining the desired edge 86

7.3.2 Categorizing noise 89

7.4 Identifying the appropriate edge within a complex image 90

7.4.1 Defining regions of interest 90

7.4.2 Edge detection 91

7.4.3 Whole image mask (WIM) and perimeter mask 91

7.4.3.1 Black-white thresholding 91

7.4.3.2 Morphological operations 92

7.4.3.3 Holes and flood fill 93

7.4.3.4 Component labelling 93

7.4.3.5 Separating regions 94

7.4.3.6 Perimeter mask 95

7.4.4 White mask 97

7.4.5 Bubble mask 97

7.5 Applying the masks 98

7.6 Final edge filtering 99

7.7 Alternative algorithms 99

7.8 Improving edge accuracy 99

7.8.1 General remarks 99

7.8.2 Subpixel resolution 100

7.8.3 Image angle adjustment 100
7.9 Flowcharts 101
7.10 Chapter summary 103

III Application to static and dynamic systems **105**

8 Static measurements of oil-water interfaces **109**
8.1 Overview 109
8.2 Interfacial tension measurements 110
8.2.1 A note regarding fluid properties 110
8.2.2 Holm measurments 110
8.2.3 Pendant drop measurements 111
8.2.4 Comparison of ADSA and the holm method 112
8.2.5 A note on measuring the air-water interfacial (surface) tension . 112
8.3 Shape fitting in a finite (bounded) fluid 114
8.3.1 Non-axisymmetric containers 114
8.3.2 Wall effects 114
8.3.3 The effect of sphere size on the measured value 114
8.4 Measurement of interfacial tension in systems with low Bond numbers 117
8.4.1 Failure of pendant-drop fitting 117
8.4.2 Criteria for accurate fitting (pendant and sessile drops) 118
8.4.3 Alternative measurement methods for low Bond number systems 120
8.5 Alleviating complications with fluid handling 120
8.5.1 The measurement of viscous and/or opaque samples 120
8.6 Alleviating stability issues 121
8.7 Chapter summary 121

9 Measuring the effect of microwave irradiaton on interfacial tension 123

9.1 Overview 123

9.2 The mechanism of microwave heating 124

9.3 Interfacial tension measurements 125

9.3.1 Using the holm meridian for microwave measurements 125

9.3.2 Microwave experiments 126

9.3.3 Monitoring the system temperature 128

9.3.4 Systems without surfactants 129

9.3.5 Brine 129

9.3.6 Charged surfactants: CTAB 131

9.3.7 Non-ionic surfactants: Triton X-100 132

9.4 Comparing conventional and microwave heating (Triton X-100) 132

9.4.1 The effect of microwave heating on interfacial tension 132

9.4.2 The effect of conventional heating on interfacial tension 134

9.4.3 Hysteresis 134

9.4.4 The effect of temperature on interfacial tension 138

9.4.5 The effect of electromagnetic waves 139

9.4.6 The formation of nano-bubbles and micro-emulsions 140

9.4.7 Disolved gasses 141

9.4.8 Thermal degradation 141

9.5 Advantages offered by the holm method 141

9.5.1 Measurements in non-quiescent samples 142

9.5.2 Measurements over long periods of time 143

9.5.3 Measurements in a fully-enclosed space 143

9.6 Chapter summary 143

10 Modelling the dynamic tension of the decane-carboxylic acid interface 147

10.1 Introduction 147

10.2 The system and reaction equilibria 148

10.2.1 Equilibrium states 148

10.2.2 The carboxylate/carboxylic acid equilibrium 150

10.2.3 The oil-water equilibrium 150

10.2.4 Acid distribution between phases 151

10.2.5 Sodium salts 152

10.3 Surface adsorption 153

10.4 Mass transport 153

10.5 Equilibrium interfacial tension 154

10.6 Interfacial tension measurements 155

10.7 Dynamic modelling 157

10.8 The effect of pH on interfacial tension 159

10.8.1 Evaluating the empirical model 159

10.8.2 Evaluating the dynamic model 159

10.8.3 Measuring hydroxide diffusion: the reverse direction 160

10.9 Chapter summary 160

11 The effect of a magnetic field on interfacial tension 163

11.1 Magneto-responsive surfactants 163

11.1.1 Ferrofluids 163

11.1.2 Metal complexes ($FeCl_3/FeCl_4^-$) 164

11.1.3 Metallo-coordinated surfactants 165

11.2 Interfacial tension measurements 166
11.2.1 Silicone oil-water interface in the presence of $FeCl_{3(aq)}$ 166
11.2.2 Decane-water interface with magneto-surfactants (DTAB-Gd) . 167
11.2.3 Decane-water interface in the presence of a water-soluble ferrofluid. 169
11.3 Discussion 170
11.3.1 Significance of surface/interfacial tension changes 170
11.3.2 Physical effects of a magnetic field on responsive drops 170
11.3.3 Magnet location, and the strength and direction of the magnetic field. 171
11.4 Chapter summary 172

12 Concluding remarks **173**
12.1 Method overview 173
12.2 Key objectives of the holm technique 174
12.3 Key differences between the holm technique and existing methods . . 175
12.4 Novelty 177

A Synthesis of magnetic surfactants **179**
A.1 Synthesis 179
A.2 Characterisation 180
A.2.1 Melting point 180
A.2.2 Solubility 180
A.2.3 FTIR 181
A.2.4 UV-Vis spectroscopy 181

B Coding **183**

B.1 An overview of the coding structure 183

B.2 Code included in this abridged appendix 184

B.3 Holm main GUI (controller GUI) 184

B.4 ThreshGUI (interactive thresholding and edge identification) 185

B.5 AngleGUI (automated angle adjustment) 186

B.6 Timeline (formatting of temperature files) 187

B.7 SaveOutputGUI (export output `.txt` file to Excel template) 188

B.8 Excel template 188

B.9 Holm_Main (code for back-end analysis) 188

List of Figures

1.1 An example of the holm meridian, formed by deforming an oil-water interface around a Teflon sphere, showing (a) the original image and (b) a thresholded image highlighting the holm meridian in black. . . . 2

2.1 Defining axisymmetric menisci and their coordinate systems. 11

2.2 Theoretical profiles of (a) pendant drops, (b) sessile drops and (c) submerged holms: three of the four interface shapes defined by Boucher et. al. 14

2.3 The sigmoid curve describing the relationship between interfacial tension and surfactant concentration. 20

2.4 The effect of increasing surfactant concentration on interfacial and aggregation behaviour. 22

3.1 Force-based measurement methods for interfacial tension. 27

3.2 An overview of the standard ADSA program. 31

4.1 The coordinate system used for the holm meridian. 41

4.2 Comparison of the holm meridian formed around (a) a cylinder and (b) a sphere. 46

5.1 Schematic of the experimental setup and a brief overview of the methodology. 54

5.2 An example of an electrochemical cell. 55

5.3 (a) DTAB/water and (b) DTAB-Gd/water interfaces measured using a 5.5 mm Teflon sphere in a 1 cm^2 cuvette. (c) Silicone/water+$FeCl_3$ around a 9.33 mm ball in a 3 cm × 3 cm cell. 58

5.4 The raised holm obscured by the fluids rising towards the wall. 59

5.5 The effect of fluid height on the curvature of the bulk interface. 60

5.6 The (a) submerged and (b) raised holms formed between a dense, coloured ferro-fluid (10% in water) and decane around a ceramic sphere. 63

6.1 An overview of the program and its basic methodology. 66

6.2 An example of the raw image file, (a) before and (b) after cropping. Only the lit area is required. 69

6.3 An image, with the detected edges overlaid in red, showing the user-defined regions to be used for analysis. 69

6.4 Flowchart showing the method to determine the initial coordinates to be used for integration of the differential Young-Laplace equation. . . . 73

6.5 Schematic of the process used to estimate the image tilt. 79

6.6 Result of the nearest neighbour search between polynomials fitted to the left and right sides of the holm. 79

6.7 An example of fitting to an asymmetric image. Note that while the initial coordinates change, an effective fit is managed from both sides. 80

7.1 A greyscale image with clear contrast differences between the solid and upper fluid phases. 85

7.2 Original greyscale image showing reflections and both adhering and detached noise. 87

7.3 Results of Canny edge detection of the image in FIGURE 7.2 using the default thresholds.. 88

7.4 The result of Canny edge detection on an image where the interface is broken by strong reflections. 89

7.5 The initial binary image produced from thresholding FIGURE 7.1 and the effects of the morphological "close" operation. 92

7.6 The black and white image showing basic thresholding. 94

7.7 The BW mask after closing and filling FIGURE 7.6. 95

7.8 (a) the "WIM" mask and (b) corresponding perimeter mask. 96

7.9 A closeup overlaying the perimeter mask over the detected edges on the right-hand holm. 96

7.10 (a) The original greyscale image showing circular objects detected by MATLAB's **imfindcircles**. (b) The perimeter mask with the bubble mask overlayed. 98

7.11 An overview of the image analysis algorithm, showing images of the key steps. 101

7.12 Flowchart describing the image analysis algorithm. This flowchart describes the same processes as FIGURE 7.11 on a function level. . . . 102

8.1 Sample images used for analysis 111

8.2 A sample image of the air-water interface. 113

8.3 Schematic representation of distances in the cell. 115

8.4 The hexadecane-water interfacial tension calculated from four distinct sphere sizes. 116

8.5 Successful fitting of the silicon-water interface with the sphere placed less than $2 \times L_c$ from the wall. 117

8.6 The theoretical effect of changing the density difference between fluids, with all other parameters kept constant. 119

9.1 The effect of microwaves on intermolecular bonds. (a) Alignment of water molecules to the magnetic field. (b) Disruption of intermolecular bonds. 124

9.2 A schematic of the setup used for the measurement of interfacial tension inside of the microwave reactor. 126

9.3 An example of the temperature profile of the aqueous layer during microwave irradiation. 127

9.4 The interfacial tension of the decane-water interface, subjected to irradiation at 60 W for 60 s. 130

9.5 A sample of brine (0.01 M) and decane irradiated at 60 W for 60 s. . . . 130

9.6 The change in interfacial tension of the decane-water interface with the cationic surfactant, CTAB, irradiated at 60 W for 60 s. 131

9.7 The interfacial tension of the decane-Triton (0.66 mM) interface, subjected to irradiation at 60 W for 60 s. The interface changed rapidly during irradiation. 133

9.8 The interfacial tension of the decane-water interface in the presence of Triton X-100, subjected to irradiation at 60 W for 60 s. Similar trends were observed across three concentrations. 133

9.9 A schematic of the setup used for the measurement of interfacial tension using a waterbath for heating. 134

9.10 Changes in the interfacial tension of the decane-water system with time in the presence of Triton X-100 (0.66 mM) when heated in a waterbath. 135

9.11 Changes in the interfacial tension with temperature of the decane-water system in the presence of Triton X-100 (0.66 mM) when heated in a waterbath. 136

9.12 Comparing the temperature-IFT curves of the decane-Triton X-100 (0.66 mM) interface when heated in a waterbath and using a microwave. 137

9.13 Comparing the initial and final values of interfacial tension for samples heated multiple times. 137

9.14 Visualising the effect of microwaves on hydrogen bonding networks. . 139

9.15 An oil-water interface heated to boiling inside of the microwave reactor. 144

10.1 The status of the systems at equilibrium, showing the distribution between phases at (a) high and (b) low pH. 149
10.2 Key species, their equilibria and distribution through the two phases. 151
10.3 Schematic showing the experimental setup. 156
10.4 Images of the hexadecane-nonanoic acid interface, showing the white precipitate marking the diffusion of hydronium ions through the cell. 157
10.5 Modelling the dynamic interfacial tension. 158

11.1 The cell used for interfacial tension measurements in the presence of a magnetic field. 167
11.2 The change in interfacial tension of the silicone oil-water+$FeCl_3$ system under the effect of a magnetic field. 168
11.3 The changes in interfacial tension of the decane-water system. 169

B.1 Main program GUI. 183
B.2 Main program GUI. 184
B.3 Interactive thresholding GUI. 185
B.4 Angle adjustment GUI. 186
B.5 Formatting for temperature files. 187
B.6 Template for Excel files. 188

List of Tables

2.1 Parameters of the normalised Young-Laplace equation, as presented by Boucher in 1987. 17

6.1 An example of how the starting estimates and initial coordinates are arranged during repeat analysis. 78

8.1 Fluid densities 110

8.2 Comparison of the results obtained using the new method and commercial pendant drop software for four water-oil systems. 112

8.3 The effect of sphere size on system geometry for the measurement of the oil-water interface. 116

10.1 Fitting parameters and physical properties 158

A.1 Melting point data 180

A.2 Solubility trends (all surfactants displayed the same general trends) . . 180

Chapter 1

Introduction

1.1 Overview

Interfacial tension is a widely used parameter that describes the excess energy at an interface.[9] As it can be directly measured, its physical relevance is well understood and far more accessible than mathematical constructs such as the Gibbs surface excess adsorption. Interfacial tension is a key parameter linked to the stability of fluid-fluid emulsions. Understanding the dynamic change in interfacial tension associated with various physical and chemical processes has the potential to offer a wealth of new information to a range of industrial processes, from food and pharmaceuticals though to flotation and oil-water separation. It is used to characterise the strength of surfactants and amphiphilic compounds by quantifying their tendency to enrich at an interface,[33] and describes the shapes of fluid drops and interfaces.[44]

A range of techniques have been developed to calculate interfacial tension, using both force- and shape- based methods. Of these, a series of drop shape techniques called Axisymmetric Drop Shape Analysis (ADSA) are capable of calculating interfacial tension with good accuracy from the shape of pendant or sessile drops. However, these techniques are designed for vibration-free surroundings in ideal laboratory environments. Use of these techniques for dynamic analysis is restricted by the fragility of pendant and sessile drops, and pumping restrictions or optical constraints can make even equilibrium measurements nigh-on impossible for some systems. Furthermore, pendant and sessile drops become approximately spherical as the size or density difference between the two phases is reduced, and ADSA fails in such a scenario.

It is clear that a more robust technique is required to facilitate interfacial tension analysis in dynamic settings. The holm technique was developed with an eye to providing continuous *in situ* measurement of complex, moving or otherwise physically constrained systems. A particular focus is placed on the anlaysis of *liquid-liquid* interfacial tension, as many techniques exist which are capable of accurately measuring surface tension.

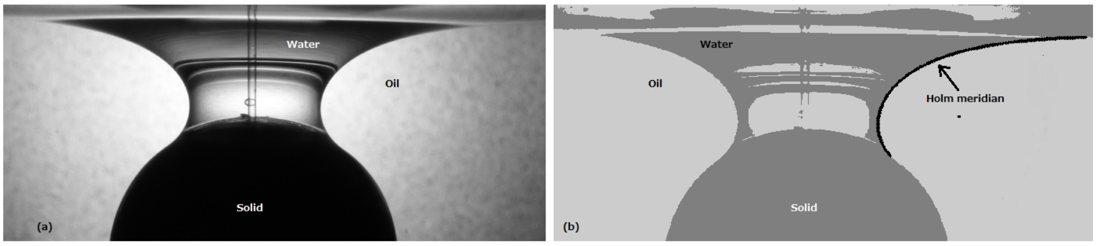

Figure 1.1: An example of the holm meridian, formed by deforming an oil-water interface around a Teflon sphere, showing (a) the original image and (b) a thresholded image highlighting the holm meridian in black.

The holm meridian is formed when the horizontal interface between two bulk phases is deformed by a solid object. A common example is the rising meniscus formed by the air-water interface as it approaches a glass wall. In the new technique, the holm is formed around a sphere of accurately known diameter submerged at the fluid interface, as shown in FIGURE 1.1. The key advantage of the holm meniscus is the added stability provided by a solid object. Where pendant drops are notoriously unstable and difficult to hold for long periods of time, the holm meridian is easily maintained for several days despite temperature fluctuations, vibrations from nearby machinery or even stirring of the sample. The technique lends itself to the measurement of the interfacial tension between two liquids, an area that can become quite complicated using other drop-shape or force-based techniques. Of key importance in the interests of mimicking industrial conditions, the entire cell and contents can be completely enclosed, as attested to by its use for the *in situ* measurement of alkane-water interfacial tension during microwave irradiation. This technique is then a stepping stone towards the measurement of interfacial tension under industrial conditions of high temperature, pressure, and movement.

This research provides a technique which eliminates the calibration requirements of force methods and overcomes the difficulties of pumping and stability associated with pendant and sessile drops, vastly simplifying the measurement of liquid-liquid interfacial tension. With external equipment – such as pumps or hanging plates – eliminated, and the basic equipment paired down to a simple cell and sphere, the novelty of this technique extends to the potential for use in a pressurized environment. Furthermore, as it is even possible to maintain the interface when subjected to external vibration sources, such as produced by a microwave reactor, it is conceivable these techniques could eventually lead to the direct, *in situ* measurement of interfacial tension within an operational processing plant.

The simplicity of the experimental component opens the doors to consider *in situ* measurement within the incredibly complex environments found in industry. While conventional measurement methods undeniably provide fast and accurate measurement of surface tension of liquid-gas systems, the niche for this new method focuses on liquid-liquid interfaces and the potential ability to mimic industrial conditions.

1.1.1 Novelty

The novel aspect of this work is the use of a widely used theoretical basis to a new physical situation. The holm meridian was identified as one of the four basic interfacial shapes long before the development of ADSA. Nonetheless, to the best of the author's knowledge, this is the first time that the methodology of drop-shape fitting has been applied to the holm meridian. The stability and simplicity of this technique provides a potential link between highly accurate measurements in controlled laboratory conditions and *in situ* measurements of complex industrial systems.

1.1.2 Structure of this thesis

The first part of this thesis provides background into interfacial tension and drop-shape analysis. The second part outlines a program developed in the MATLAB coding environment capable of calculating the interfacial tension from an image of the meniscus. In the final chapters, the method is applied to a range of scenarios that are difficult to measure using existing techniques, highlighting the functionality of the holm method.

Part I

An introduction to interfacial tension

PART I of this thesis provides a background on interfacial tension and interfacial phenomena. An overview of the existing methods used to measure interfacial tension is given, and the holm meridian is then discussed in more depth.

CHAPTER 2

A background on interfacial tension

2.1 The basis of interfacial tension

Picture a fluid drop in air. The rapid, dramatic change in physical properties at the interface of the drop exposes a molecule at the interface to an environment vastly different to those in the bulk. A molecule along the interface is bound to fewer molecules than one in the bulk phase, and the energy imbalance between two contacting substances results in net stresses at the boundaries of the fluids.[55,90] The differences in binding energies leads to a surface energy density,[90] and ultimately to the force known as the *interfacial tension*. In a solid-gas or liquid-gas system, the net effect of these stresses is often referred to as the *surface tension*.

Consider an interface with area A subjected to a tension force of γ mN/m. A change in area of $\mathrm{d}A$ requires external work equivalent to the surface energy of the additional area, or will in turn do work on the environment to release the energy associated with the area lost.[90]

$$\mathrm{d}W = \gamma \mathrm{d}A \tag{2.1}$$

It is clear, then, that expanding an interface against a positive interfacial tension requires work. By consequence, these systems will tend towards the smallest possible interfacial area. This is exemplified in soap bubbles: light enough to have negligible distortion due to gravity, a free-floating bubble will tend to a spherical shape to achieve the smallest interfacial area for a given volume. The contrasting case of a negative interfacial tension is not unusual in liquid-liquid systems either. As in this situation energy is released as the surface expands, the interface will tend to the largest area possible – in other words, complete mixing of the two fluids.[90] The present work is concerned with systems of imiscible fluids. In other words, systems where a positive interfacial tension causes the two phases to remain distinct.

2.2 Pressure discontinuity across the interface

A droplet or bubble with a positive interfacial tension will contract until the tension forces are balanced by the interior pressure. This leads to an equilibrium state where no net work is done:[90]

$$\mathrm{d}W = \gamma \mathrm{d}A - \Delta p \mathrm{d}V = 0 \tag{2.2}$$

As the interface applies an inwardly-directed force (a positive interfacial tension), the pressure inside the bubble or droplet will always be greater than the external pressure. This produces a pressure discontinuity over the interface, analogous to the pressure discontinuity over the interface between two bulk phases settled one over the top of the other as defined by Pascal's law.

2.2.1 The Young-Laplace equation

Thomas Young (1805) and Pierre-Simon Laplace (1806) are jointly credited with the realisation that an interface acts mechanically, as though under tension.[90] The resulting Young-Laplace equation describes the pressure discontinuity across the interface (Δp) as a function of the interfacial tension (γ) and the mean curvature of the surface.[24,124]

$$\Delta p = \gamma \left(\frac{1}{R_1} + \frac{1}{R_2} \right) \tag{2.3}$$

The two principal radii of curvature, R_1 and R_2, are any two orthogonal radii passing through the point in question, as exemplified in FIGURE 2.1.

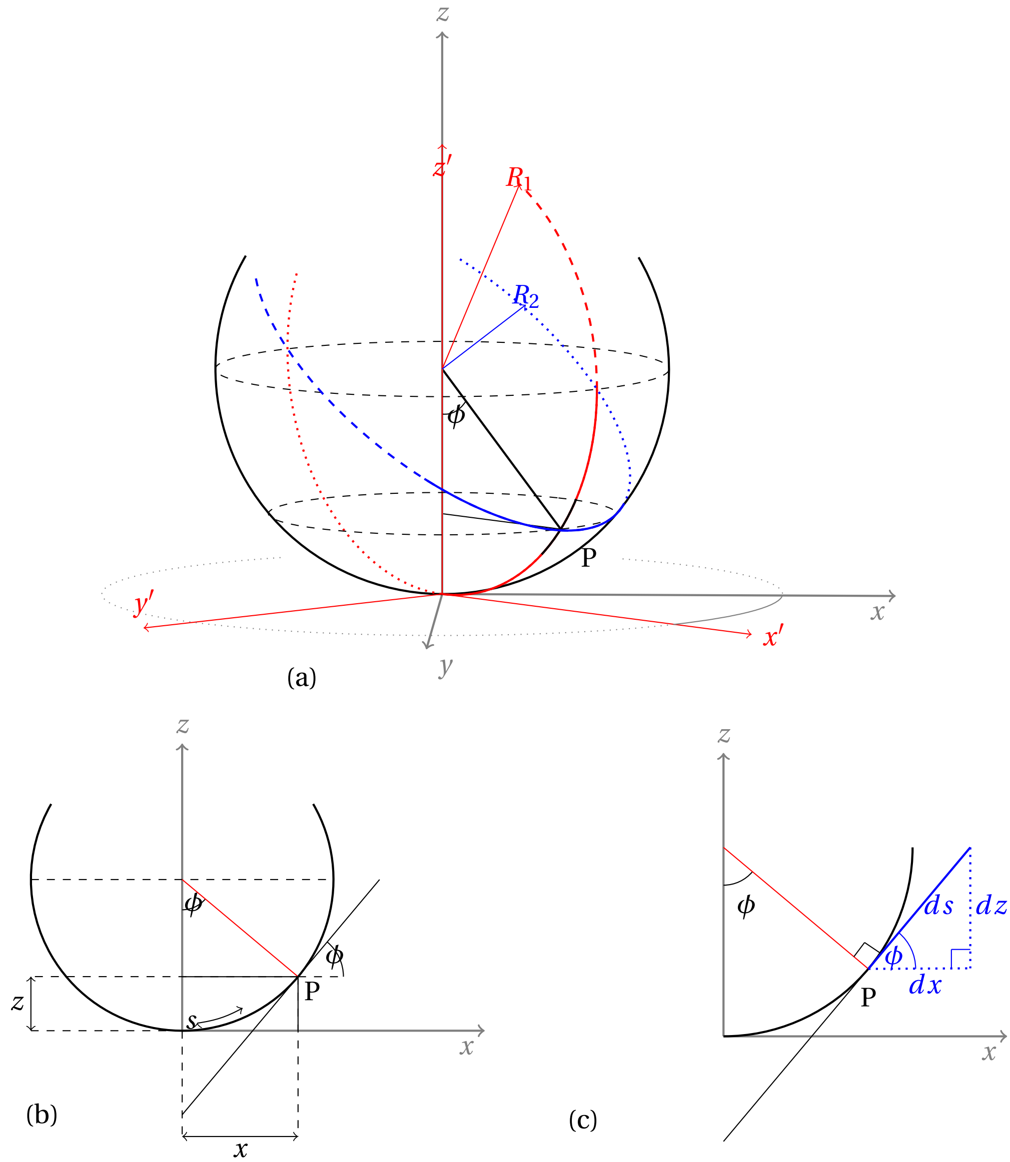

Figure 2.1: Defining axisymmetric menisci and their coordinate systems. (a) The two principal radii of curvature at a point P along the meniscus are shown. One of the principal directions, R_1, describes the osculating circle in the $x'z'$ plane (*red*), along the drop surface (*black*), thus defining the local rate of change of the angle, ϕ, with the distance, s, along the profile of the drop. The second principal direction, R_2 describes a circle (*blue*) in the plane orthogonal to R_1. In a two-dimensional drawing (b), this vector will be seen to come out of the page. (c) Geometric configuration leading to the two auxillary equations, (2.11b) and (2.11c). The tangent line at point P is extended a small distance ds, with a corresponding change in the width, dx, and height, dz, which are related by trignometric identities.

2.2.2 Excess pressure accross an interface

Pascal's law describes the change in hydrostatic pressure with a change in height through a constant gravitational field for a species of density ρ_i:

$$\frac{\delta p_i}{\delta z} = -\rho_i g \tag{2.4}$$

As drops and menisci, being three dimensional, necessarily involve a change in the vertical axis, z, it follows that the difference in pressure across the interface will change with the z axis unless both fluids have the same density. The Young-Laplace equation (2.3) is typically applied with Pascal's law to relate the hydrostatic pressure with the vertical height, yielding (2.11):

$$\gamma\left(\frac{1}{R_1} + \frac{1}{R_2}\right) = \Delta P_0 + (\Delta\rho) g z \tag{2.5a}$$

$$= \frac{2\gamma}{R_0} + (\Delta\rho) g z \tag{2.5b}$$

In (2.11) above, P_0 and R_0 are, respectively, the pressure and characteristic length at the datum point. For pendant and sessile drops, it is convenient to align the vertical axis with the centerline of the drop and take the z intercept (the drop apex) as the datum point. Both pendant and sessile drops are approximately spherical at the the tip of the drop, hence the two orthogonal radii of curvature are equal: $R_1' = R_2' = R_0$. Thus, the pressure difference P_0 can be written:

$$\Delta P_0 = \gamma(\frac{1}{R_1'} + \frac{1}{R_2'}) \tag{2.6}$$

$$= \frac{2\gamma}{R_0} \tag{2.7}$$

2.2.3 Capillary length and gravitational effects

The capillary length (2.8) defines a characteristic length of a system for which the hydrostatic pressure is negligible in comparison to the excess pressure.[90]

$$L_c = \sqrt{\frac{\gamma}{\Delta\rho g}} \tag{2.8}$$

Gravity distorts droplets, leading to the characteristic elongated shape of pendant drops and the flattened shape of the sessile drop. Bubbles and droplets significantly smaller than their capillary lengths are affected only negligibly by gravity, and can be estimated simply as spheres or spherical caps. In the case where $\Delta\rho = 0$, such as a thin film separating an air bubble from the atmosphere, gravitational effects are again negligible and the film shape will be a spherical cap. Thus a simple soap bubble, consisting of air separated from the bulk air phase by a thin film, is spherical unless acted on by external forces.

2.2.4 Distortion in a gravitational field: Axisymmetric menisci

Axisymmetric fluid bodies have an axis of symmetry around the vertical axis. Such profiles include the fluid-fluid interface of drops and bubbles, as well as the interface formed around an axially symmetric object such as a sphere or vertical cylinder, where the meniscus is not acted on by any force other than gravity. There are four generic forms of axisymmetric menisci, identified by Boucher in a series of papers.[21–24] Each of these have an inverted counterpart (reflected across the xy plane) depending on the relative densities of the bulk and droplet fluids. FIGURE 2.2 shows three of these basic shapes: pendant drop/emergent bubble, sessile drop/-captive bubble and the raised/submerged holm meniscus. The final shape is the heavy/light liquid bridge.

2.2.5 Describing the shape of axisymmetric menisci

FIGURE 2.1 shows the droplet profile changing with the angle ϕ from the z axis. The two principal directions of curvature and their radii are shown along with the arcs that they describe at point P on the droplet profile. The point P is shown in the $x'z'$ plane to lend clarity, and lies on the drop surface (*black*) rotated out from the full profile drawn in the xz plane.

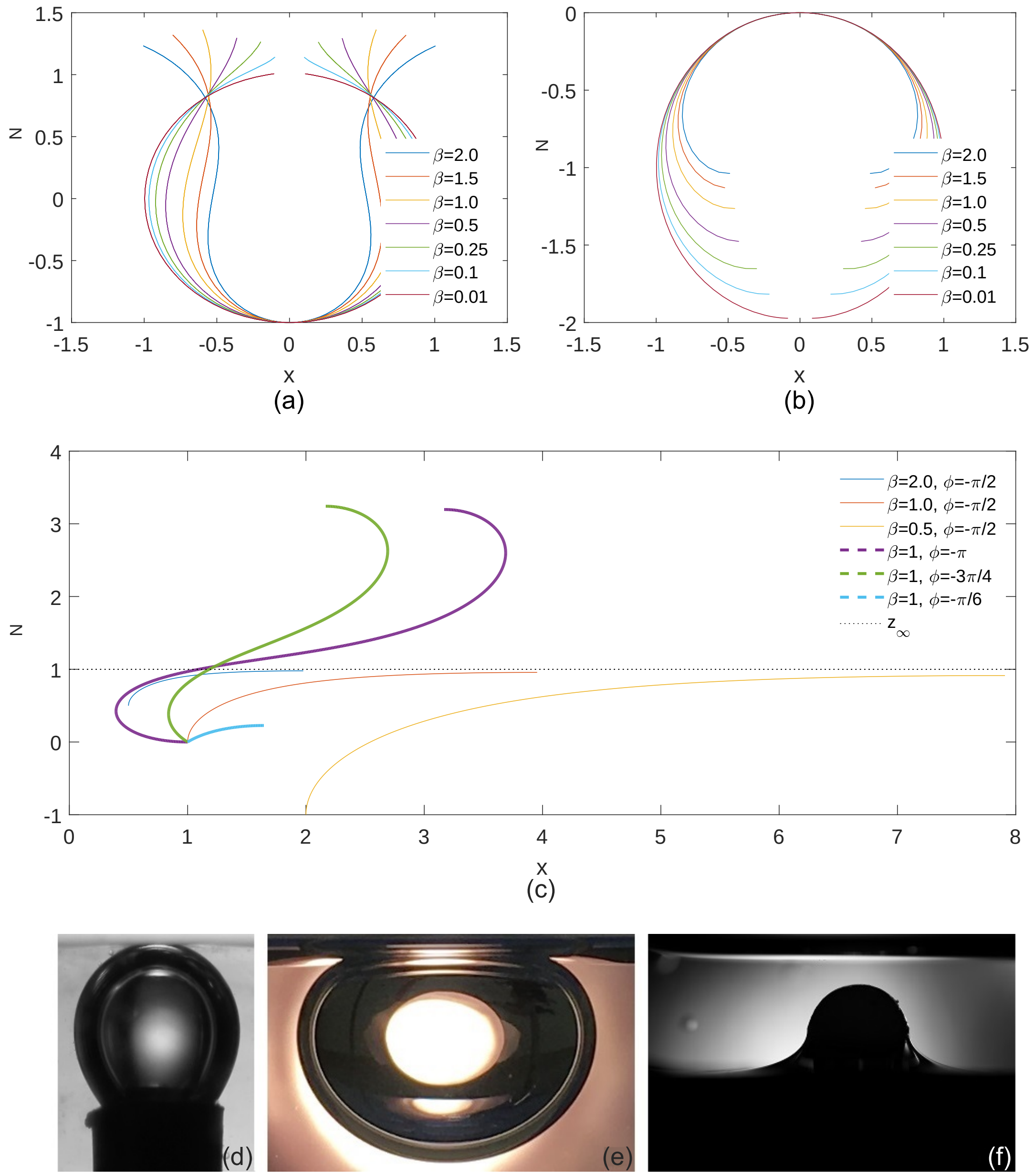

Figure 2.2: Theoretical profiles of (a) pendant drops, (b) sessile drops and (c) submerged holms: three of the four interface shapes defined by Boucher et. al.[24] Their inverted counterparts are shown in (d) (emergent bubble), (e) (captive bubble) and (f) (raised holm).

The two principal radii of curvature are orthogonal, and their associated radii describe circles in orthogonal planes passing through point P. The first direction, associated with R_1, describes the red circle in the $x'z'$ plane. For small changes in ϕ, the change in the length, s, along the drop profile is given in (2.9):

$$\mathrm{d}s = R_1 \mathrm{d}\phi \tag{2.9}$$

The second osculating circle (R_2, *blue*) rotates perpendicular to the first. On a two-dimensional representation of the droplet profile, R_2 will rotate out of the page. The radius can be related to the horizontal displacement, x, using simply geometry, to give (2.10):

$$\frac{1}{R_2} = \frac{\sin\phi}{x} \tag{2.10}$$

EQUATION 2.9 and (2.10) are combined with (2.5b) to give (2.11) below, and is solved through numerical intregration with the addition of (2.11b) and (2.11c), obtained by simple geometry.

$$\frac{\mathrm{d}\phi}{\mathrm{d}s} = \frac{2}{R_0} + \frac{(\Delta\rho)gz}{\gamma} - \frac{\sin\phi}{x} \tag{2.11a}$$

$$\frac{\mathrm{d}x}{\mathrm{d}s} = \cos\phi \tag{2.11b}$$

$$\frac{\mathrm{d}z}{\mathrm{d}s} = \sin\phi \tag{2.11c}$$

At the apex, where $x = z = s = 0$, (2.10) gives

$$\frac{\sin\phi}{x} = \frac{1}{R_0} \mathrm{for} R_2 = R_0|_{s=x=0} \tag{2.12}$$

hence avoiding the issue of dividing by $x = 0$. This method is used by del Rio and Neumann[48] to avoid use of l'Hôpital's rule to handle the case at the drop apex. (2.11a) then becomes

$$\frac{\mathrm{d}\phi}{\mathrm{d}s} = \begin{cases} \frac{1}{R_0}, & \mathrm{for} s = x = 0 \\ \frac{2}{R_0} + \frac{(\Delta\rho)gz}{\gamma} - \frac{\sin\phi}{x}, & \mathrm{else} \end{cases} \tag{2.13}$$

Equation (2.11a) describes the two-dimensional shape of axisymmetric menisci under the influence of gravity. This profile is then rotated by 360° around the vertical axis to describe the three-dimensional body. Consideration of the geometry of the droplet provides the simple definitions of volume (2.14) and surface area (2.15) of the three-dimensional solid:

$$\frac{\mathrm{d}V}{\mathrm{d}s} = \pi x^2 \sin\phi \tag{2.14}$$

$$\frac{\mathrm{d}A}{\mathrm{d}s} = 2\pi x \tag{2.15}$$

Boucher et al.[24] reports a version of the Young-Laplace equation with parameters H and λ that can be changed to represent the four types of menisci (2.16). The values for the two parameters and the computational ranges are shown in Table 2.1.

$$\frac{\mathrm{d}\phi}{\mathrm{d}S} + \frac{\sin\phi}{X} = 2(\lambda H - Z) \tag{2.16a}$$

$$\frac{\mathrm{d}X}{\mathrm{d}S} = \cos\phi \tag{2.16b}$$

$$\frac{\mathrm{d}Z}{\mathrm{d}S} = \sin\phi \tag{2.16c}$$

EQUATION 2.16 uses reduced coordinates X and Z where the capillary constant $a = \sqrt{2}L_c$ (2.18) is used as the reducing factor. The shape factor, H (2.19), is defined in more detail in the following section.

$$X = \frac{x}{a}, \; Z = \frac{z}{a} \text{ and } S = \frac{s}{a} \tag{2.17}$$

$$a = \sqrt{\frac{2\gamma}{\Delta\rho g}} \tag{2.18}$$

$$H = \frac{h}{a} \tag{2.19}$$

Table 2.1: Parameters of the normalised Young-Laplace equation, as presented by Boucher et al. [24] to compute the theoretical profiles. The inverted counterparts are reflections in the x axis.

Meridian type	λ	H	X	Z	Computational range
Pendant drop	+1	+	0	0	$0° < \phi < 180°$
Sessile drop	+1	+	0	0	$180° < \phi < 360°$
Heavy bridge	+1	±	X^o	0	$360° < \phi < -180°$
Submerged holm	—	0	X^*	Z^*	$0° < \phi < 179.5°$

2.2.6 Dimensionless systems and reducing factors

Reducing (2.11a) to dimensionless coordinates greatly simplifies computation of the curves. A range of reducing parameters are used in various papers, typically variations of the capillary number and Bond numbers. Boucher et al. [24] comments that the choice between the capillary length and capillary constant is arbitrary, provided that it is clear which is being used. Alternatives include using the characteristic length (the radius of curvature at the drop apex) as the reducing factor. [48]

There is also scope for using different shape factors. The shape factor, H (2.19), used by Boucher et al. [24] is defined as half of the pressure drop at the lowest point of the interface, ΔP_0, expressed as a hydrostatic height, thus:

$$h = \frac{\Delta P_0}{2\Delta\rho g} \tag{2.20}$$

The original tables produced by Bashforth and Adams [12] used a different shape factor, β:

$$\beta = \frac{\Delta g R_0^2}{\gamma} \tag{2.21}$$

giving the overall Young-Laplace equation the following form in reduced coordinates for pendant and sessile drops, which are distinguished by the sign of the second term:

$$J = 2 \pm \beta Z \tag{2.22}$$

In (2.22) above, J stands for the mean interfacial curvature.

Any of these methods can be used effectively, provided that they are applied consistently throughout the program. The reader will note that $H^2 = 2\beta$ and $H = a/R_0$.[24]

2.3 Surface chemistry: impurities and surface-active molecules

In SECTION 2.1 (pg. 9), the interfacial tension was related to changes in the bonding environment of a molecule at the interface. Disrupting these environments with surfactants or impurities can have startling effects on the interfacial tension. Surfactants (*surf*ace *act*ive age*nts*) are amphiphilic molecules where different parts of the molecule interact with phases on different sides of the interface.

2.3.1 "Real systems": surfactants and impurities in industrial processing

Surfactants are used extensively in industry throughout a range of applications. They have long been used in soaps and detergents, where their amphiphilic nature is the key behind solubilising oils in aqueous solutions. In recent times, surfactants have seen a wide-spread use in novel fields, including biology and medicine, where their stabilising[46] and carrier[37] capabilities are widely sought after. Surfactants are used in all stages of the petroleum industry to influence the properties of emulsions.[132] In mineral flotation, surfactants are used to stabilise froth and enhance the chemical affinity of certain materials to improve flotation efficiency.[96]

Even where compounds are not added intentionally, very few industrial systems can be considered "chemically pure". The purification steps undertaken for laboratory measurements are laborious and time consuming, as even trace amounts of some impurities can have measurable effects on the interfacial tension. Because of this, dynamic effects and the ability to predict the effect of impurities of interfacial tension is of significant practical use. The remainder of this chapter will discuss interfacial phenomena involving surfactants and impurities in more depth.

2.3.2 Impurities

In the vast majority of cases, the presence of impurities at the interface will reduce the interfacial tension. As the hydrogen bonding in water is unusually strong, the presence of almost any impurity will result in a weaker bond than would otherwise have been. The outcome of this is that the surface tension of pure water (72.8 mN/m) [89] is one of the highest known and frequently used as a reference value. Surfactants, particularly organic compounds, tend to disrupt this bonding, thus lowering the interfacial tension.

Impurities with negative surface excess (predominantly inorganic impurities like salts) will increase surface tension; those with a positive surface excess will reduce it. It is difficult to give firm rules as to the actions of a particular impurity or mixture of impurities in the interfacial layer, as the interactions between ions may be governed by energetic and entropic effects particular to the mixture. [17] Combining both organic and inorganic impurities can have interesting effects, as salts will tend to alter the hydrophobicity of organic compounds, lowering their solubility in the aqueous phase. For a more thorough discussion, the reader is directed to a recent review by Björneholm et al. [17].

2.3.3 Interfacial behaviour of surfactants

“Surface active” molecules are so named as they tend to concentrate at phase boundaries. By sitting in the interfacial zone, the surfactant molecules can minimise their energy through aligning "like" phases. This tends to concentrate the amphiphile at the phase boundary until the interfacial area is saturated, meaning that the subsurface concentration is significantly higher than the surfactant concentration in the bulk. As a result, surface active agents can significantly affect interfacial properties at even low bulk concentrations. Even small, freely miscible organic molecules such as ethanol are known to enrich at the water suface. [17]

Consider a simple air-water system with two bulk phases, into which a known quantity of surfactant is added in minute increments. If the surface tension were to be measured after each addition of surfactant, a curve much like FIGURE 2.3 would be observed. Initially, when $C_{surf} = 0$, the surface tension is that of the pure solvent. As surfactant is added, the interfacial tension decreases slowly throughout phase I.

The surfactant molecules will align themselves with their hydrophobic parts in the air and the hydrophillic parts interacting with the water, although the interface remains sparsely populated. The available published data suggests that the surface coverage reaches roughly 10% by the end of phase one, although this number is dependent on the surfactant type.[99] While surfactant molecules in the bulk migrate

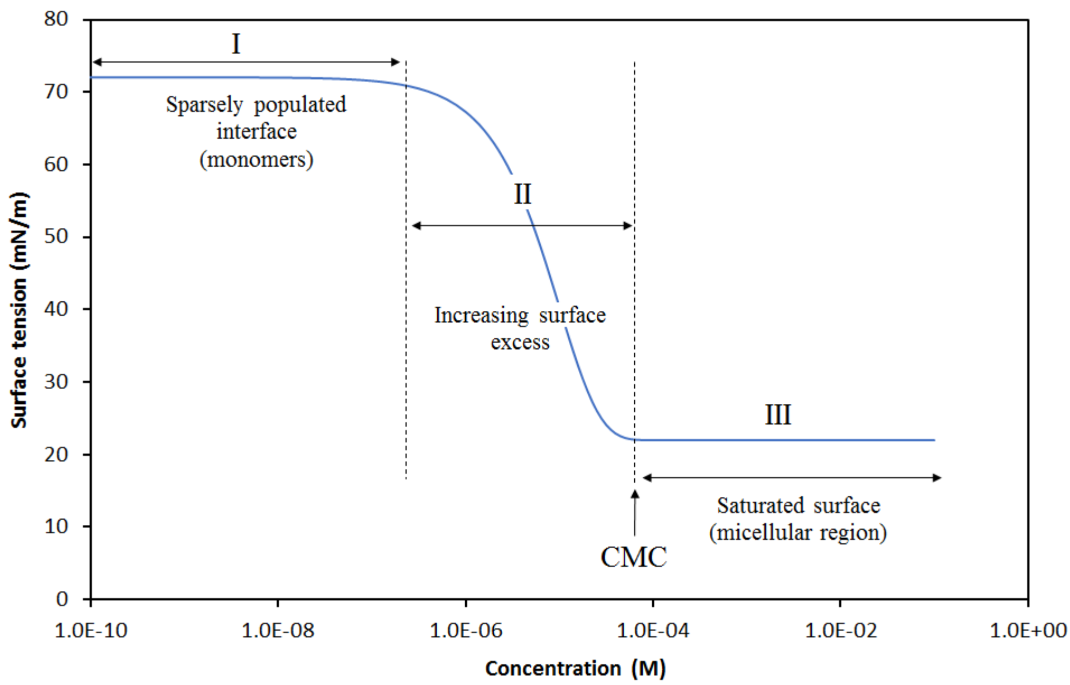

Figure 2.3: The sigmoid curve describing the relationship between interfacial tension and surfactant concentration.

towards the interface, there is also a movement of adsorbed surfactants to return to the bulk, creating a dynamic equilibrium. However, the net movement is of the surfactant to the interface. At a certain critical concentration, the adsorption process becomes co-operative and the interfacial tension begins to decrease rapidly throughout phase II, proportional to $\log\left(C_{surf}\right)$.

The Gibbs expression of the surface excess concentration, $\Gamma_2^{(1)}$, relates the bulk concentration of a species to the interfacial tension.[71] Notably, a factor referred to as the *activity coefficient* (α) is required to account for non-ideality of the species in solution, referring to the tendency of a molecule to enrich (or otherwise) at the interface.

$$\Gamma_2^{(1)} = -\frac{1}{nRT}\frac{\mathrm{d}\gamma}{\mathrm{d}\ln(\alpha C)} \tag{2.23}$$

The Gibbs surface excess is a description of the concentration in the interfacial layer *in excess* of the concentration in the bulk. The concentration is defined in terms of a mathematical construct known as the Gibbs dividing plane, which avoids uncertainty regarding the exact location of the interface. The distinction should be made between the *absolute* surface concentration, Γ and the (Gibbs) surface *excess* concentration, $\Gamma_2^{(1)}$, which are related by (2.24), where x is the molar fraction of the solute in the bulk.

$$\Gamma_2^{(1)} = \Gamma_1 - \frac{x}{1-x}\Gamma_2 \tag{2.24}$$

The surface excess is closely related to the partition constant (K), which is simply the ratio of the concentration of molecules between the two phases:[94] $K = \frac{C_{light}}{C_{dense}}$. The partition coefficient expresses the affinity of a molecule for the lighter phase.

$$\Gamma_2^{(1)} = \Gamma_{2,max}^{(1)} \frac{KC}{1+KC} \tag{2.25}$$

As the surfactant concentration increases, the critical micelle concentration, or CMC, marks the point where the surfactant has almost reached its maximum surface concentration. This point is characterised by the flattening curve in phase III of the sigmoid shown in FIGURE 2.3, as the surface concentration becomes essentially constant. Mukherjee et al.[99] report a typical surface coverage equal to roughly 90% of the available area. This is known as the limiting surface concentration, $\Gamma_{2,max}^{(1)}$ (relative) or Γ_{max} (absolute).

With the surface fully saturated, many surfactants will begin to self-assemble into micelles, shielding the non-polar tails and aligning their polar heads with the surrounding water molecules. This shields the alkyl chains from contact with the polar molecules, while allowing the polar heads to maintain the energetically favourable bonding networks with water, in a compromise known as the "hydrophobic effect".[132] Within a polar solvent, the volume inside the micelle becomes a hydrophobic environment capable of accepting suitable guest molecules, and hence of interest in a variety of applications such as drug carriers.

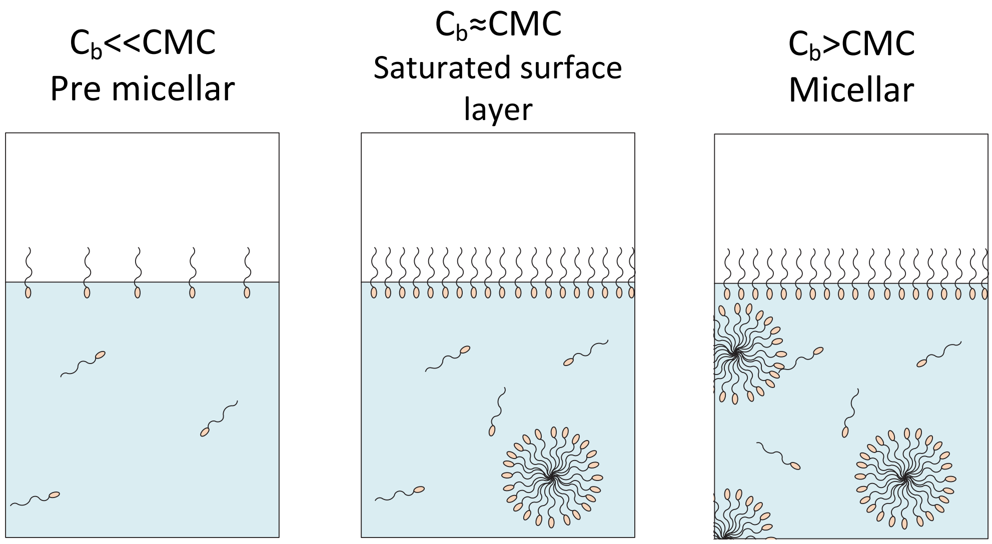

Figure 2.4: The effect of increasing surfactant concentration on interfacial and aggregation behaviour. At low concentrations (a), surfactants enrich at the interface until the interface reaches saturation. (b) Micelles begin to form once the bulk concentration exceeds the critical micelle concentration, which coincides with near-saturation of the interface. (c) The number of micelles increases as the bulk concentration increases, keeping the quantity of monomers essentially constant.

2.3.4 Surfactant use with emulsions

Surfactants are frequently used to stabilise multi-phase systems. Emulsion stability is improved with lower interfacial tensions and smaller drop sizes.[1] Reducing the surface tension reduces the amount of mechanical energy (agitation) required to break two continuous phases into small droplets, forming an emulsion. Thus, surfactants can both facilitate the formation of emulsions[47] and improve emulsion stability, reducing the tendency of the droplets to coalesce by forming charged barriers to repel other drops.[1] However, other surface active molecules (demulsifiers) are used to increase the speed of demulsification.[85,92] Although the exact mechanism is not well understood, it is thought to be independent of the ability of the surfactant to reduce interfacial tension.[62] Emulsions significantly increase the interfacial area available for surfactant adsorption.

2.3.5 Equilibration and dynamic systems

The presence of interfacially active species introduces an element of dynamism into the system. As these species will migrate towards the interface, there necessitates a certain amount of time for this transfer to take place, during which the interfacial tension will decrease from the interfacial tension of the pure solvent to the equilibrium interfacial tension. This length of time is associated with the rate of diffusion of the surfactant from the bulk to the interface. For small, strong surfactants, this can be a relatively short period of time. For large, complex molecules, such as proteins, this time can be of the order of hours or even days.[15] Dynamic effects can also result from chemical reactions occurring at the interface, or due to surfactants reacting to external stimuli.[31]

A significant length of time may be required for a system to come to equilibrium. This is particularly true in ever-changing industrial environments. As a result, knowledge of the dynamic interfacial tension is often of more physical and predictive value than equilibrium measurements. The next chapter will detail existing methods used to measure surface and interfacial tension. It will become clear that dynamic measurements on liquid-liquid systems is particularly complex, and that many situations exist that cannot be adequately measured using the current methods.

CHAPTER 3

Existing methods for the measurement of interfacial tension

3.1 Chapter overview

A range of methods exist for the measurement of surface and interfacial tension in laboratory environments. For the most part, the methods can be divided into two main groups: force measurement techniques and shape fitting methods. In addition, several other commonly used techniques have been developed for specific experimental conditions, such as the measurement of dynamic interfacial tension on very short time scales using the maximum bubble method. In this chapter, we will discuss the existing methods for the measurement of interfacial tension and the types of measurement to which they are suited.

3.2 Force-based methods

Force-based methods measure the force required to oppose the interfacial tension. For example, the force required to draw a submerged object through the interface or the maximum weight of a drop that can be supported by the interfacial tension. Measurements are typically done at the point of failure – i.e., the maximum drop weight is just before the drop pulls free of its support.

3.2.1 The Wihelmy Plate Method

The Wihelmy Plate method, first published in 1863,[152] has since become one of the most frequently used force-based techniques for the measurement of surface tension. A thin metal plate, typically made of platinum or Pt-Ir, with an accurately known surface area and wetted perimeter (P_w), is submerged in a fluid before being drawn back up through the interface. The rise of the wetted plate pulls the interface upwards, expanding the interfacial area. Accordingly, the plate's movement is opposed by the interfacial tension. The force required to withdraw the plate is given in (3.1):[39]

$$F = \gamma P_w \sin\theta \tag{3.1}$$

The surface tension is typically measured at the point where the force required to raise the plate reaches its maximum, just as the plate pulls free. If the plate is fully wetted, the estimation $\theta = 0^o$ is generally acceptable.[39] Alternatively, provided that the surface tension is known, it is possible to reverse the technique to calculate the contact angle.[63] The method is illustrated in FIGURE 3.1(a).

3.2.2 The Du Nuöy Ring

The basic approach to measurement using the Du Nuöy ring is identical to the Wihelmy plate method, the difference being solely the shape of the suspended solid. Surface tension is determined from the force, F, required to draw the ring through the interface, as illustrated in FIGURE 3.1. The fluid attached to the ring is drawn from the bulk in a roughly cylindrical shape, giving, for a ring of radius R:[53]

$$\gamma = \frac{F}{4\pi R} \tag{3.2}$$

The maximum force is measured just before the fluid (of volume V) detaches from the ring, corresponding to a downwards force due to the liquid weight:

$$F = \Delta\rho g V \tag{3.3}$$

In practice, a correction factor is required as the raised fluid is not perfectly cylindrical in shape due to the curved holm.

As the technique expands the interface at the point of detachment, leading to a deficiency of surfactant at the surface, the du Nuöy ring technique is often found to report a high value of interfacial tension unless the surfactant has a very short equilibration time.[108]

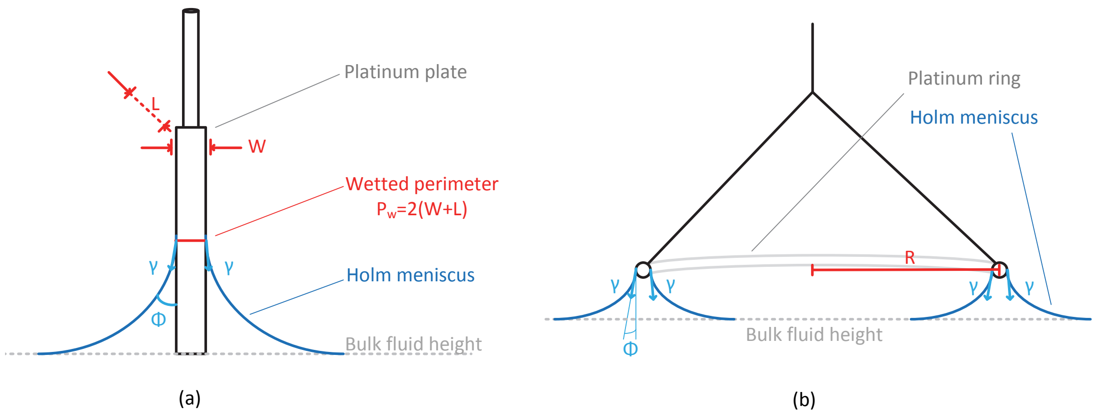

Figure 3.1: Force-based measurement methods for interfacial tension. (a) Wilhelmy plate method, (b) du Nuöy ring method. As the plate are pulled through the interface, their passage is opposed by the tension force.

3.2.3 Calibration and modifications for liquid-liquid measurement (Du Nuöy Ring and Wilhelmy Plate methods)

Both the Wilhelmy plate and Du Nuöy ring methods are best suited to the measurement of surface tension. They are typically calibrated using the water-air surface tension and hence conversion/calibration factors are required for the measurement of the interfacial tension between two liquids. These conversion factors account for the substantial change in the density differences between liquid-fluid and liquid-liquid interfaces. While it is certainly true that these techniques provide accurate and repeatable measurements if set up and used correctly, nevertheless much care is required in their calibration and use.[9]

3.2.4 The drop-volume/drop-weight methods

An alternative force-based method measures the maximum weight or volume that can be supported by the tension force.[53] By controlling the drop volume, V, using an automated capillary, the force on the drop (the drop weight) can be accurately determined. The point at which drop detaches is typically monitored by a light sensor.

The interfacial tension is related to the drop weight by

$$\gamma = \frac{V \Delta \rho g}{2\pi r_{cap}} f \tag{3.4}$$

The correction factor, f, accounts for the drop detaching at the necking point rather than the capillary tip, and r_{cap} is the capillary radius.

3.2.5 Limitations of the force methods

The force methods are capable of accurate measurement provided sufficient care and attention is taken in the experimental details. The condition of the plate or ring is of fundamental importance in obtaining accurate measurements. Typically, a fine platinum ring or plate on a platinum wire is used. The metal is heated to a glowing orange prior to use, in order to ensure that impurities are burnt off and the plate or ring is totally clean, as even minute traces of impurities will affect the measurement.

The plates/rings are easily deformed, such that it no longer hangs evenly in the device. This results in error in the force measurement as the plate does not pull cleanly from the interface. In a similar vein, the measurement can be affected by movement in the bulk fluid causing the plate to swing on its wire. Unfortunately, as pulling the plate/ring through the interface disturbs the surface, it is often very difficult to minimise movement in the sample if multiple measurements are to be taken quickly.

Issues arise when the metal is wetted by both phases, or otherwise incompletely wetted by one fluid – a particular issue with liquid-liquid measurement. Heertjes et al.[68] refers to issues with incomplete wetting when measuring the interfacial tension between water and large alcohols, attributed to the similarities in surface affinities between the two phases, and comments that coatings (i.e. black platinum) can be used to circumvent this issue. Li et al.[91] found that surface tension measurements using the Wihelmy and du Nuöy methods systematically underestimated the limiting surface coverage of cationic surfactants, attributed to incomplete wetting of the plate or ring due to the negatively charged metal surfaces. Consistent differences were noted between CMC measurements of cationic surfactants using ADSA, plate and ring methods, with the onset at or around the CMC.[91]

As the drop-volume method, the du Nuöy method and the Wihelmy plate method all involve increasing the interfacial area (as the drop or lifted volume increases until the point of failure), these methods are generally ill-suited to the measurement of dynamic interfacial tension and indeed should be limited to the measurement of equilibrium interfacial tensions of rapidly equilibrating systems,[39] although some dynamic measurements are possible with the plate method. Measurement of liquid-liquid systems requires the use of correction factors and careful experimental work, as one phase must be completely wetted and the balance zeroed at the interface.[68]

Apparatus for force-based measurements are typically large, rendering them unsuitable for containment and measurements under pressurised conditions. More sample is required than for pendant drops. However, as measurements are made on the bulk solution, they are appropriate choices for measuring interfacial tension isotherms using an auto-dilutor to alter the surfactant concentration. The reader should note that all of the force-based measurements listed here require direct and repeated contact with the sample.

3.3 Shape based methods

As discussed earlier, the Young-Laplace equation relates the shape of a drop to the density difference between the phases and the interfacial tension. In other words, the shape of an interface can be predicted for a known γ and $\Delta\rho$. Shape fitting techniques make good use of this predictive ability. With the ready availability of fast computing power, a suite of techniques have been developed to utilise the Young-Laplace equation with numerical optimisation to calculate the interfacial tension from the shape of (images of) pendant and sessile drops, a technique known as Axisymmetric Drop Shape Analysis (ADSA). Recently, the ADSA algorithm was even adapted for use on a smartphone.[40] Whereas ADSA considers only the coordinates on the profile of an interface, a later technique known as Theoretical Image Fitting Analysis (TIFA) applies similar principles to fit the entire (2D) image.

3.3.1 Axisymmetric drop shape analysis (ADSA)

ADSA is a method where interfacial tension is estimated based on the shape of curved interfaces. The method relates the Young-Laplace equation to the curvature of an interface extracted from images of drops and bubbles. These techniques use a reduced, parameterized form of the Young-Laplace equation (2.16) which can be solved using numerical integration. Provided that the density difference between the two phases is known and there is some scale available to relate the image back to its real world size, the program then optimises the surface tension and characteristic lengths of the system until the theoretical profile that is the best match for the drop image has been found.

A very good summary of the development of drop-shape analysis is given by Hoorfar and Neumann[72], and the reader is directed there for more information. The earliest fundamental work was the creation of tables by Bashforth and Adams[12] in 1883 which tabulate the profile of sessile drops for a given combination of γ and R_t. The surface tension of a drop could be estimated by interpolating the tables. While others contributed to expanding these tables, the procedure known as ADSA was originally developed by Rotenberg et al.[124] in 1983, requiring manual edge detection. Cheng et al.[42] developed a methodology to computerize the edge detection component an 1990, and ADSA has since expanded to a variety of specific drop and bubble shapes: ADSA-CB (captive bubble)[118]; ADSA-CSD (constrained sessile drop)[127]; ADSA-NA (no apex)[83]; ADSA-EF (electric field)[13]. ADSA is now one of the most frequently used techniques for the measurement of surface tension, requiring minimal solution volumes and providing highly accurate results.

3.3.2 The pendant drop technique (ADSA-P)

On a high-level overview, all techniques in the ADSA family follow the same methodology, outlined in FIGURE 3.2. High-resolution photographs of pendant or sessile drops are analysed by edge detection algorithms to determine the coordinates of the interface. (Typically gradient edge detection such as Canny[36], Sobel[52] or SUSAN[139] are used, although recent additions such as entropic edge detection[9,70] have been proposed for noisy images.) The user must supply the accurately known densities, or at least the density difference between the two phases, and the length of a single pixel in real-world terms. Typically, the latter is done by taking a photograph of an object of known size – such as the width of the capillary – at the same magnification as the drop, thus relating a certain number of pixels with an equivalent length in millimeters. A shape factor is then estimated from an assumed interfacial tension. The program then estimates the theoretical drop profile for a given shape factor (in this case, the capillary constant, c) and drop length, R_0, and calculates the error with the detected edge. The best fit is determined by optimising c and R_0 for the minimum fitting error, and the interfacial tension can then be back calculated from the optimised parameters through the equation:

$$\gamma = \frac{\Delta\rho g}{c} \tag{3.5}$$

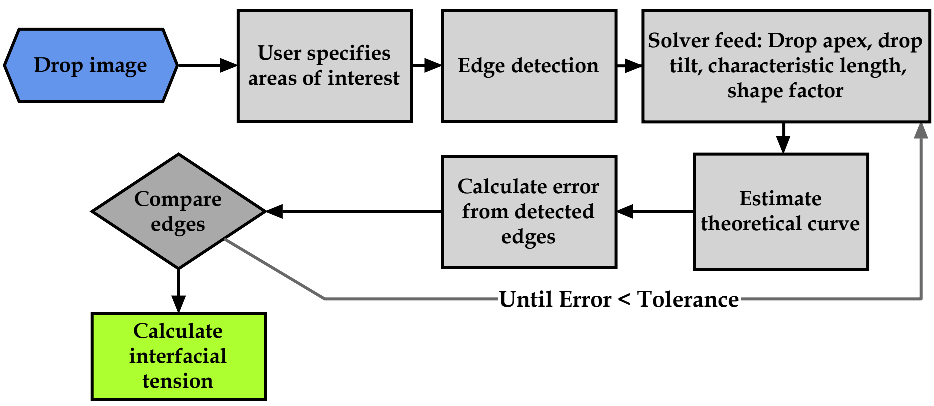

Figure 3.2: An overview of the standard ADSA program.

Pendant drop methods are eminently suitable to the measurement of the surface tension of liquids. The phase-inverted counterpart – an emergent bubble – is also applicable to this technique. The measurement of liquid-liquid interfacial tension is also possible, however difficulties begin to arise when pumping viscous fluids. The main issue associated with the pendant drop is stability. Firstly, movement of the drop introduces additional parameters into the force balance that may distort the axial symmetry. Furthermore, vibrations often result in significant blurring of the drop image, leading to issues with edge detection and drop fitting. Secondly, ADSA and other drop shape techniques rely on the drop deformation of the fluid interface. The extent of deformation depends on the relative strengths of (1) the elongating gravitational force and (2) the interfacial tension attempting to minimise surface area by pulling the drop into a spherical shape. The precision of ADSA-P is improved as the deformation of the droplet (i.e. deformation from the zero-gravity spherical shape) increases, meaning that the largest (and hence least stable) drop is desired for fitting. In consequence, the experimental setup must be kept very still to allow accurate and continued measurement on the same drop.

It is widely reported in the literature [73,126,128] that the error associated with drop-shape analysis will increase significantly for near-spherical drops as large changes in interfacial tension incur only small changes in drop shape. As early as 1991, Cheng and Neumann [41] had observed that data points at the neck of the drop are critical for accurate analysis as it is here that the most significant deformation can be seen. Some techniques have discussed weighting the error associated with different coordinates based on their location on the drop profile. [82]

There have been attempts in the literature to identify critical ranges where (pendant) drop techniques will yield reliable results. It is widely accepted that fewer, more accurate coordinates will yield better results than a large number of poor coordinates, and consequently, image resolution is a fundamental aspect of the technique. Hoorfar and Neumann [72] applied random perturbations to theoretical (perfect) drop profiles. By steadily reducing the proportion of the drop profile available for fitting, they were able to determine cut-off points showing were the ADSA algorithms would fail for drops with different shape factors. This investigation clearly showed that the points around the drop neck were critical for accurate analysis of otherwise spherical pendant drops.

It is desirable to maintain the drop needles and solid substrates properly vertical (or horizontal) in order to ensure that the interface remains axisymmetric. In the older versions of ADSA, it was necessary to adjust for tilt in the camera, typically by including an image of a plumb line in the drop image and rotating the image until the line was properly vertical. In the new generation ADSA [35], vertical axis tilt was included as an additional fitting parameter.

3.3.3 Modifications to ADSA for specific applications

The ADSA methodology was applied to sessile drops (dense phase sitting on a surface) and captive bubbles (light phase trapped by a surface). The principles of measurement are the same as for the pendant drop in ADSA-P. The measurement of surface and interfacial tension using sessile drops will typically incur more error than for pendant drops, in part due to uncertainty around the contact line, but simultaneous measurement of contact angles with the solid becomes possible. The issues of a drop falling or bursting is reduced with the sessile drop technique as the drop is well supported by the substrate. However, the drops are not fixed to the surface and are still capable of coming free. Image analysis is often more complex for the sessile drop, as it is can be difficult to obtain good contrast with the solid substrate and identify the true contact circle. [101] The program may fail for flat sessile drops as the curvature at the drop apex tends to infinity and the algorithm is unable to converge. [118]

ADSA-CB (captive bubble) was developed primarily for the measurement of low surface tension liquids with the potential to suffer from film leakage. ADSA-CSD (constrained sessile drop) was developed for generating film balances to measure the collapse pressure of insoluble monolayers. [127] Shape-fitting techniques are uniquely disposed to measuring the collapse pressure (marked by a change in the slope of the surface tension - area per molecule isotherm) as both surface tension and the drop's surface area can be measured simultaneously. ADSA-EF (electric field) was developed to account for the force of an electric field on the shape of a conductive drop. [13] ADSA-NA (no apex) determines the surface tension of liquid bridges and sessile drops formed around a capillary. [83] For a more detailed summary, the reader is referred to a review by Saad and Neumann. [129]

3.3.4 Theoretical Image Fitting Analysis (TIFA)

Theoretical Image Fitting Analysis (TIFA)[34] is an alternative approach to the method of drop fitting. Rather than extracting edge coordinates directly, the TIFA approach attempts to match the entire image. The theoretical profile derived from the Young-Laplace equation is converted into a binary image,[83] and then a gradient image. The pixel-by-pixel error between the theoretical and experimental gradient images is minimised. TIFA is not a modular program – edge detection is undertaken as part of the optimisation procedure, meaning that the detected edges are themselves constrained by the Young-Laplace equation.[129]

The TIFA methodology was adapted for use with fluid bodies without an apex (lenses and liquid bridges) in a technique known as TIFA-IA.[35] Curiously, even with the inclusion of liquid bridges, neither this method nor regular ADSA has been adapted for use with the final interfacial shape: the holm meridian.

3.3.5 Issues with drop-shape methods

Both force measurements and drop-shape measurements are typically undertaken on anti-vibration benches to minimise the effect of external forces. Evaporation is a concern for lengthy measurements, particularly with the large surface area-to-volume ratios of pendant drops, and the drops may consequently be enclosed in some way to control humidity. Pumps and associated paraphernalia are often employed to control drop volume over longer periods.

3.4 Other methods for specific applications

3.4.1 Systems with low Bond numbers

Peters and Arabali[112] proposed a drop-shape method to calculate interfacial tension without use of the Young-Laplace equation. By replacing the Young-Laplace equation with a force balance across the bubble cap, an explicit equation for interfacial tension is derived. In this way, issues with low Bond number systems (where the drop becomes essentially spherical) are avoided. However, a highly accurate pressure measurement is required in its place. Note that this method still suffers from the same uncertainties as other optical analysis techniques, and pumping is still required to form the drop.

An adaptation of ADSA-P uses a compound pendant drop to increase the deformation of low bond number systems.[102] By attaching a small scilica sphere to a pendant drop, the increased deformation allowed low surface tensions ($Bo \approx 0$) to be measured using the Young-Laplace equation for a liquid bridge.

In low-Bond axisymmetric drop shape analysis (LBADSA), a small-perturbation solution for the sessile drop profile is optimised in much the same way as ADSA. However, rather than fitting directly to a set of detected coordinates, an image energy approach using the complete pixel information is applied for improved fitting of noisy images.[140]

3.4.2 The spinning drop method

The Young-Laplace equation of capillarity, which forms the basis of most drop-shape tensiometric techniques, describes the curvature of an interface under a constant gravitational field.[24] An additional term extends (2.11) to account for the effects of a centrifugal field:[149]

$$\gamma\left(\frac{1}{R_1}+\frac{1}{R_2}\right)=\gamma\frac{1}{R_0}+\Delta\rho g z+\Delta\omega^2\lambda^2 \tag{3.6}$$

where λ denotes the potential distance across the bubble ($R_0 - R_2$) and ω is the angular velocity of the centrifugal rotation.

In 1942, Bernard Vonnegut proposed a drop-shape technique where a light fluid drop is maintained in a dense fluid in the center of a rapidly rotating glass tube. At sufficiently high rotational speeds, the centrifugal acceleration $\omega^2\lambda$ dwarfs the gravitational acceleration g, leading to an elongated drop centered around the horizontal axis of rotation.[149] The spinning drop method is widely used for analysis of low Bond number systems.

3.4.3 The Capillary rise method

A rise of height h of a liquid inside of a cylinder of radius r making contact with the wall at an angle ϕ is related to the interfacial tension by (3.7),[95] where the term $\frac{r}{3}$ is an approximation to account for non-spherical menisci.

$$\gamma = \left(\frac{rg}{2\cos\phi}\right)\left(h + \frac{r}{3}\right)(\rho_1 - \rho_2) \tag{3.7}$$

3.4.4 The maximum bubble pressure method

The maximum bubble pressure method (MBPM) allows measurements of dynamic or equilibrium surface tension for well-defined surface ages.[100] The pressure inside of a bubble being blown from a fine capillary of radius r_{cap} passes through a maximum (P) when the bubble is hemispherical. Thus, by correcting for the height of the capillary, the surface tension can be obtained as

$$\gamma = \frac{pr_{cap}}{2} \tag{3.8}$$

Each fresh bubble constitues a new air-liquid surface. Hence, the dynamic effects of surfactant sorption can be characterised by changing the time required to reach the maximum pressure, in turn achieved by adjusting the rate at which the bubbles are blown, effectively altering the age of the interface.

While it is possible to use the same method with liquid-liquid systems, the experimental difficulties (higher densities, viscosities, pumping pressures etc.) are prohibitive.

3.4.5 Analysis of the capillary rise profile around a cylinder (ACRPAC)

This final method is mentioned not as a method to measure surface or interfacial tension, but as it is one of the very few techniques which uses a shape-fitting methodology with the holm meridian. ACRPAC is a technique to determine the contact angle of a fluid with known surface tension against an axisymmetric solid (cylinders or cones) by measuring the capillary rise around the outside of the rod.[63] Provided that the sample container is significantly larger than the rod, the bulk fluid will return to the “undisturbed” (or reference) height. Thus, the theoretical profiles can be generated from the Young-Laplace equation for the holm. With the reference height and surface tension known, the contact angle is taken as a fitting parameter and used to optimise the fit of the first-order nonlinear ordinary differential equations.

3.5 Where the niche is

As shown by this chapter, a wide variety of techniques exist to measure surface and interfacial tension. Measurement of surface tension is significantly easier than the measurement of interfacial tension. Liquid-liquid systems typically suffer from difficulties with image analysis due to noisy or poor-contrast images, affecting drop-shape methodologies, or require complex correction factors and careful experimental work. Viscous fluids provide pumping difficulties and drop stability can be a serious problem. The need for vibration-free environments for the two most common methods – the Wihelmy plate and ADSA – precludes measurements outside of an ideal laboratory environment, and both techniques typically use additional bulky equipment to improve accuracy. Clearly, there is room for a technique that is robust against vibrations and sample movement, and that can be contained inside of a relatively small space. Such a technique could form the basis of interfacial tension measurements inside less ideal environments: constrained spaces, external vibration and long experiment times.

Chapter 4

The holm meniscus

4.1 Overview

The term “holm” was coined by Boucher and Kent [26] in 1977 in the third of their extensive works on capillary phenomena. The word described axisymmetric fluid bodies formed when a planar horizontal interface is deformed by an axisymmetric object, so chosen as one of its meanings is “island”. In systems of unbounded extent, these interfaces are characterized by the horizontal asymptote as the radial coordinate tends to infinity, and the three-phase contact line describes a circle parallel to the plane of the bulk fluid. [76] Holms have only one bounding phase, unlike liquid bridges, which have two, and the two principal radii of curvature are always of opposite sign.

While the holm meridian is one of the four basic shapes proposed by Boucher [24] (see Figure 2.2(c)), so far it is the only one left to be adapted to some form of interfacial tension measurement. The closest is the ACRPAC contact angle method developed by Gu et al. [63] discussed in the previous chapter. This omission is unsurprising in many respects, as the method for the numerical computation of the holm differs somewhat from those used to produce theoretical curves for pendant and sessile drops. Firstly, holms have no shape factor. Or rather, by definition, the shape factor $2H$ is the curvature at the point where $Z = 0$, $X \to \infty$ and $\phi = 180^o$. Being a plane, the curvature at this point is zero. Consequently, the Young-Laplace equation describing the holm meridian becomes:

$$\frac{\mathrm{d}\phi}{\mathrm{d}S} + \frac{\sin(\phi)}{X} + 2Z = 0 \tag{4.1}$$

As the shape factor cannot be used, a reducing factor that involving γ (i.e. the Bond number, capillary number or capillary length) must be used in order to relate the reduced Young-Laplace equation to the interfacial tension.

The holm meridian "suffers the considerable complication of being a two-point boundary-value problem with one of the 'points' at infinity".[76] The horizontal asymptote means that the solution to the Young-Laplace equation (4.1) is in fact the solution to the family of meridians sharing a certain bulk fluid height. Where this chapter refers to an "unbounded fluid interface", it is understood that the interface extends far enough to become sensibly flat, and thus the deformations at the three-phase contact line are not impacted by any other boundary.

This chapter delves into the holm meridian in more depth, providing the core theory underlying the holm measurement technique. There is a particular focus on the form of the Young-Laplace equation that describes the interface and on the mathematical techniques used to solve it. The final section outlines the new technique.

4.2 Early work

4.2.1 Tabulated solutions

A way to define the shape of the holm meniscus was needed in the 1950s and 60s for early studies of bubbles or spheres at deformable interfaces. For small bubble, the deformation of the bulk interface was often ignored. However, for solid spheres and larger bubbles or drops, the deformation of the bulk interface needed to be taken into account.[65,116] The interface was assumed to deform according to the Young-Laplace equation. As the holm equation has no analytical solution, the shape of the bulk interface was deformed according to the tables published by Bashforth and Adams[12], and later added to by Huh and Scriven[76] and Padday and Pitt[107]. However, use of these tables requires knowledge of the initial values at the three-phase contact line, shown in red in FIGURE 4.1.

4.2.2 Early attempts at integration

Princen[116], who at this time was using computers for integration, circumvented the issue of not knowing this initial point by assuming a contact angle (ϕ_c) and using the Bashworth and Adams tables to determine the corresponding x, z coordinates. From this starting point, the Young-Laplace equation in reduced coordinates (4.3) was integrated numerically,

$$\frac{1}{R_1} + \frac{1}{R_2} = c(z - z_\infty) \tag{4.2}$$

using the boundary conditions:

$$\frac{dz}{dx} = \tan(\phi_c) \text{ at } (x_c, z_c) \tag{4.3}$$

and

$$z = z_\infty \text{ at } x \to \infty \tag{4.4}$$

where

$$c = \frac{\Delta\rho g}{\gamma} \text{ i.e. } c = \frac{1}{L_c^2} \tag{4.5}$$

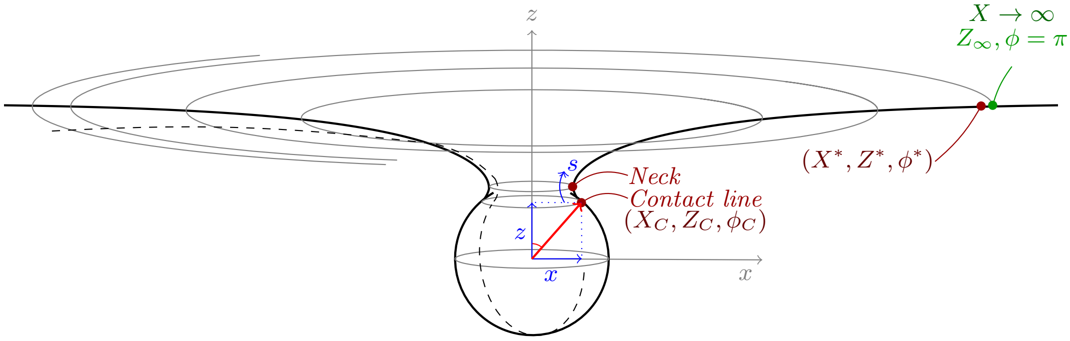

Figure 4.1: The coordinate system used for the holm meridian. In the past, integration has been started from an estimated point just before the horizontal asymptote (ϕ^*, X^*, Z^*). The new technique proposes using image analysis to determine the contact point ($\phi_{CL}, X_{CL}, Z_{CL}$) and integrate from the solid surface, eliminating part of the uncertainty in the boundary conditions. The angle ϕ is defined as $\phi = \text{atan}(\text{d}Y/\text{d}X)$.

The bulk fluid height, z_∞, was determined by relating the pressure drop across the combined drop and interface and was thus related to the system geometry. Princen[116] found the integrated profile to have three distinct shapes. At the correct contact angle, the classic holm shape would appear, with the interface tending to a horizontal asymptote. However, at low contact angles the interface would curve back in on itself, and at high contact angles, would pass through an inflection point and continue rising. The new contact angle was chosen with a modified bisection method depending on which type of holm presented.

4.2.3 Integrating using Bessel functions

Several years later, Huh and Scriven[76] proposed to solve the issue of the unknown boundary condition by commencing integration from the far end – just before the asymptote is reached. By specifying a position where the angle of the interface, ϕ^*, is 179.5^o, Huh and Scriven[76] estimated the starting position $(x^*, z^*)_{\phi^*=179.5^o}$, thus defining the curve by a single parameter, x^*. Hence, the initial conditions for integration become

$$z = z^* \text{ and } \frac{\mathrm{d}z}{\mathrm{d}x} = \tan(\phi^*) \text{ at } x = x^* \tag{4.6}$$

This point is shown in green FIGURE 4.1. Note that the vertical reference point is shifted to the height of the bulk interface.

$$z \to 0 \text{ as } x \to \infty \tag{4.7}$$

Also, as the deforming solid was a cylinder of known diameter (as opposed to the drop of unknown size considered by Princen[116]), the radial coordinate of the contact line is known:

$$x_0 = r_c \tag{4.8}$$

Various approximations were developed to avoid numerical integration of the Young-Laplace equation, with varying success.[76] One such, a modified Bessel function (K) was widely used after implementation by Huh and Scriven[76]:

$$\frac{d^2z}{dx^2} + \frac{1}{x}\frac{dz}{dx} - z = 0 \tag{4.9a}$$

$$\frac{dz}{dx} = \tan(\phi^*) \text{ at } x = x^* \tag{4.9b}$$

$$\frac{dz}{dx} \to 0 \text{ as } x = \infty \tag{4.9c}$$

Huh and Scriven[76] eventually published a series of modified Bessel functions to provide the starting height Z_0, thus tabulating results to the corresponding family of equations tending to the same asymptote. Burrill and Woods[32] provided an approach using Bessel functions in the same year. The solution to system (4.9) is given below, where K_0 and K_1 are the Bessel functions of orders zero and one, respectively.

$$Z = -\frac{K_0(x)}{K_1 x^*}\tan\Phi^* \tag{4.10}$$

The approach proposed by Huh and Scriven[76] involved producing the curves for the approximating angle ϕ^* and successive values of x^*, and then interpolating through the tables to find the profile appropriate to the system, an approach still used by Huh and Mason[75] in 1973.

4.3 Axisymmetric fluid bodies with one asymptote

It was not until 1977 that the holm was given a thorough treatment on its own. Boucher and Kent[26] first considered the shape of the unbounded interface deforming around an axially symmetric object, and then dealt explicitly with the flotation of spheres[23] and rods[25], and issues with holm and liquid bridges in finite solutions (i.e. considering wall effects) in 1978 and 1979.[27] In 1980, Boucher[21] listed the holm meridian as one of the four main types of axisymmetric menisci: pendant drop/emergent bubble, sessile drop/captive bubble, heavy and light liquid bridges, and raised and submerged holms.

Note that due to the different reducing factor used by Boucher and Kent[26], the Bessel approximation comes to:

$$\frac{\mathrm{d}^2 X}{\mathrm{d}X^2} + \frac{1}{X}\frac{\mathrm{d}Z}{\mathrm{d}X} - 2Z = 0 \tag{4.11}$$

and

$$X = BK_0(\sqrt{2}X) \tag{4.12}$$

where

$$B = -\frac{\tan(\phi^*)}{\sqrt{2}K_1(\sqrt{2}X^*)} \tag{4.13}$$

and

$$Z = -\frac{\tan(\phi^*)K_0(\sqrt{2}X)}{\sqrt{2}K_1(\sqrt{2}X^*)} \tag{4.14}$$

4.4 Modeling the meniscus

4.4.1 Development of an initial boundary problem

In all of Boucher's papers, Bessel functions are used to integrate the holm meridian as an initial value problem starting just before the asymptote is reached. This addressed the issue of not knowing the location of the three phase confluence. However, image analysis can provide another way around this issue.

During shape-fitting techniques, theoretical profiles determined from the Young-Laplace equation are fitted to the coordinates of a real drop. Hence, the coordinates of the detected edge can provide an excellent estimate as a starting point for the initial value problem, allowing the holm to be computed from the three-phase contact line. In fact, it is significantly easier to obtain the three-point contact line from an image than to determine the limiting height of the fluid, as the latter presupposes that it is possible to obtain the entire holm with good clarity in a single image. The advances in computing power mean that the holm meridians can be solved using numerical integration, without using the Bessel functions as an approximation. Integration can be started at any point along the meridian, and it is possible to obtain a good fit using only part of the interface. During integration, the distance along the meridian, S, will outwards towards the asymptote, through the range $X_C \rightarrow X^*$.

4.4.2 A new shape fitting technique

With this in mind, a new technique for the measurement of interfacial tension is proposed. The bulk interface is deformed by a solid sphere at an adjustable height. Use of a fixed solid provides stability that cannot be achieved with sessile or pendant drop experiments. Images of the drop are analysed to determine the coordinates of the meniscus and the circular profile of the sphere. As the sphere is of an accurately known size, the images are scaled by fitting a circle of radius R to the coordinates of the sphere.

The curvature of the sphere provides a smooth transition from the solid to the fluid interface. This region can be approximated by a polynomial fit to the detected coordinates. The polynomial relates the coordinates (ϕ_c, X_c, Z_c) at the three-phase contact line. This point, where the holm diverges from the sphere, is chosen as the starting point for numerical integration of the Young Laplace equationand the need to assume a starting point for integration at the horizontal asymptote is removed. The value z_∞ is estimated from the highest point among the detected coordinates. While the holm interface, strictly speaking, has a shape factor of zero, the detected coordinates are reduced using a factor $\beta = \frac{1}{L_c^2} = \frac{\Delta\rho g}{\gamma}$ as used in the original Bashworth and Adams tables, allowing the curve to be related to the interfacial tension.

A theoretical curve is generated for the initial conditions (ϕ_c, X_c, Z_c) with the assumed limiting value Z_∞ and a given reducing factor (β), determined from a user-supplied estimate of the interfacial tension. (As the function converges well, this estimate need not be in any way exact.) The error between the theoretical profile and the detected coordinates is determined, and the parameters ϕ_c, z_∞, R_0 and β are optimised to reduce this error. The interfacial tension is calculated from the vales of R_0 and β of the best fitting solution:

$$\gamma = \frac{\Delta\rho g}{\beta} \tag{4.15}$$

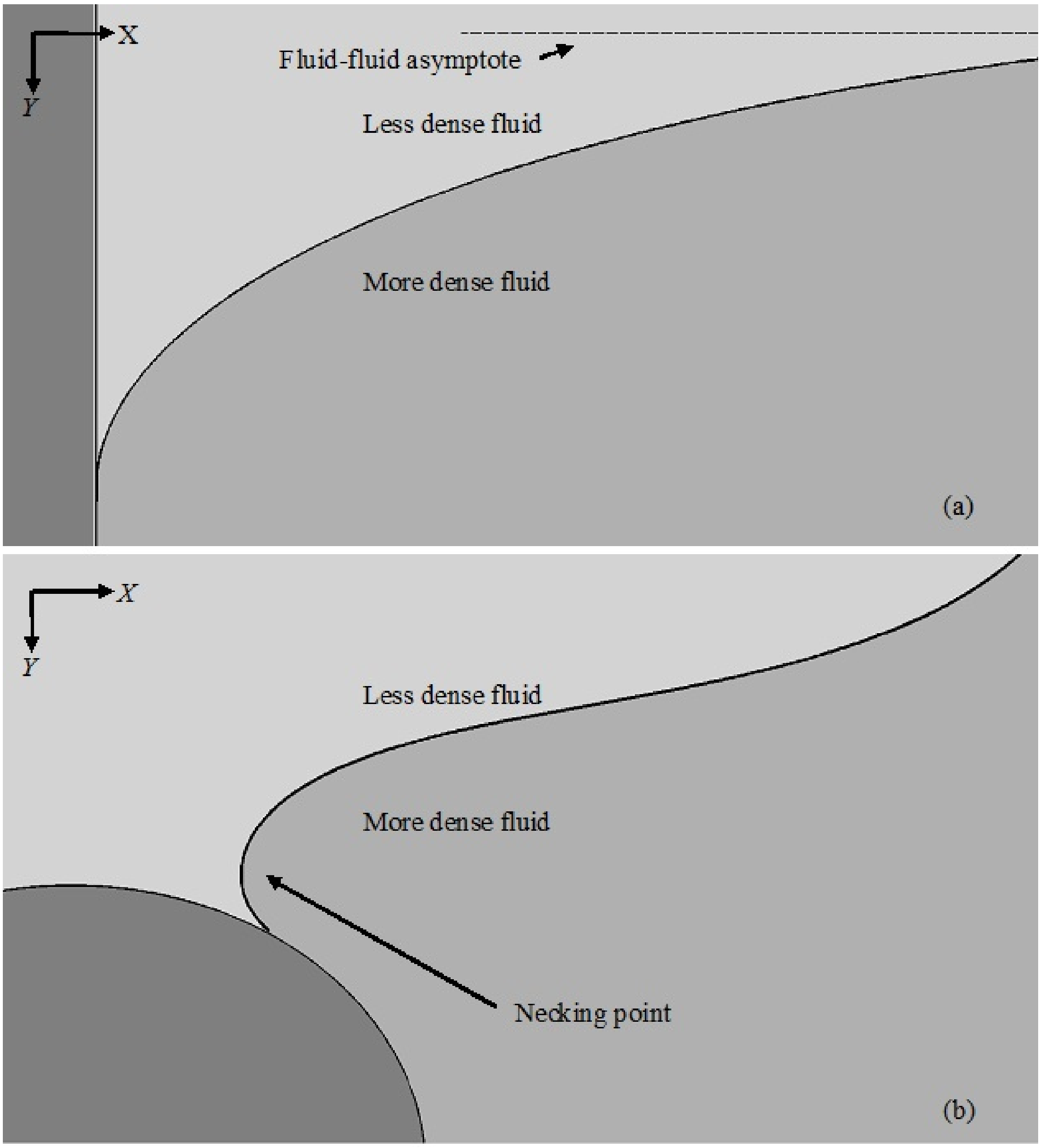

Figure 4.2: Comparison of the holm meridian formed around (a) a cylinder and (b) a sphere. A much wider range of contact angles are available using the sphere, making it possible to achieve good curvature for fitting. *Originally published in Hyde et al.*[79]

4.4.3 Choice of the submerged solid

There are three key reasons for using a submerged sphere to deform the holm:

- The size of the sphere is accurately known and is convenient for scaling. A circle can be easily fitted to the coordinates of the image to determine the width of the sphere in pixels, for scaling.
- As a sphere is symmetrical along any axis, the issues regarding vertical alignment that plagues pendant and sessile drops is removed. However, if a cylinder, cone or rod were chosen, it would be necessary to carefully align the vertical axis.
- The curvature of the sphere produces a smooth transition from the solid to the interface and allows for the maximum curvature of the holm to be reached.

The increase interfacial curvature made possible using a submerged sphere is shown in FIGURE 4.2.

4.5 Chapter summary

This chapter details the literature specific to the holm meniscus. The Young-Laplace equation has typically been solved from the asymptotic side through the use of Bessel functions, providing an approximate solution accurate to up to five decimal places.[21] With the advances in computing capabilities, it is now possible to integrate the Young-Laplace equation directly, without resorting to the Bessel approximation. A measurement technique was proposed that uses image analysis to estimate the coordinates of the contact line and hence fit the detected coordinates in a variant of drop shape analysis. The technique uses a submerged sphere to produce a stable holm meridian for fitting. The technique will measure the interfacial tension of liquid-liquid interfaces by applying the principals of drop-shape analysis to the holm meridian.

Part II

Coding and technique development

The primary focus of this Doctorate research is the development of a method to measure the interfacial tension of liquid-liquid systems by applying shape-fitting methodologies to the holm meridian.

PART II consists of three chapters. The first details the experimental work undertaken in a wet laboratory and details key points for the acquisition of images. The second chapter outlines the program methodology for the calculation of interfacial tension. This chapter encompasses the details of methodology that stand alone from the coding platform (MATLAB) and that would form the basis of the technique on any platform. The specific functions and their parameters as implemented on the MATLAB version developed in this doctorate are also detailed. The third chapter deals specifically with the difficulties of digital image analysis in the complex image and is not platform specific. The methodology implemented in the MATLAB code is equally applicable to other coding languages. The key considerations for the use of MATLAB in this doctorate were (1) the availability of code libraries for image analysis, and (2) MATLAB's powerful graphics display.

CHAPTER 5

Image acquisition

5.1 Overview

This chapter is concerned with the front-end image acquisition prior to computerised analysis and is the only part of the technique to be undertaken in a wet laboratory. Subsequent chapters detail the program and the image analysis techniques required to infer the interfacial tension from the drop image.

This thesis describes a shape-fitting method. Like all of the drop-shape techniques, the final result is strongly reliant on the accuracy and reliability of the images and system parameters (density and scaling).

5.2 Experimental technique

5.2.1 General setup

The holm meridian is formed around a solid sphere suspended at the interface. A horizontal interface is first formed between two imiscible fluids. For example, between oil and water. The fluids are contained within a transparent cell, either glass or an appropriate plastic. A solid sphere of accurately known diameter is then lowered below the height of the interface, causing the meridian to curve down to meet the solid. The extent of curvature can be adjusted by adjusting the penetration depth or by altering the chemical affinities of the fluids or solid, as will be discussed below.

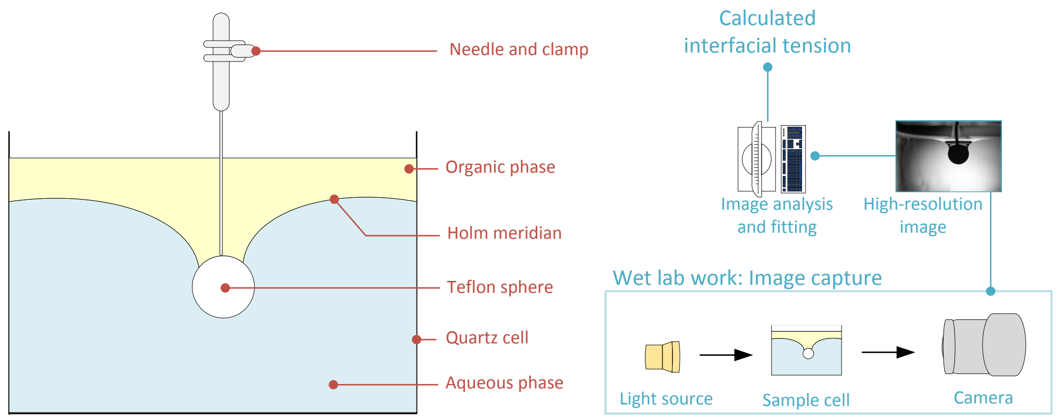

Figure 5.1: Schematic of the experimental setup and a brief overview of the methodology.

5.2.2 Scaling

The analysis program uses dimensionless numbers to reduce the computing requirements. The reduced coordinates are related to the image dimensions using the reducing factor β, which is in turn related to the interfacial tension. The images are scaled to relate a real-world length to each pixel. Any object can be used for this purpose, provided that its size is accurately known and can be detected in the image. For example, the diameters of the spheres used here are known to a tolerance of 0.1 mm.

Using the solid sphere has several benefits:

- The solid sphere shows up with good contrast in the images, allowing the edge to be easily detected.
- A circle can be fitted to the profile of the sphere in the image, allowing the number of pixels across the diameter to be accurately detected. There is no concern regarding orientation as the sphere is symmetrical in all directions.
- No additional apparatus is required – the sphere is part of the main setup.

5.2.3 Additional experimental parameters

Accurate knowledge of the fluid densities or density difference is required. If the intention is to calculate the fluid densities from empirical equations, accurate knowledge of the temperature will also be required. As it is quite possible to undertake measurements on quite large interfaces, there is plenty of room available to include additional sensors, such as pH or conductivity, on the bulk solutions. Examples of this are given in CHAPTER 9 and CHAPTER 10. FIGURE 5.2 gives an example of an electrochemical cell set up around the holm meridian, where a number of electrodes were required.

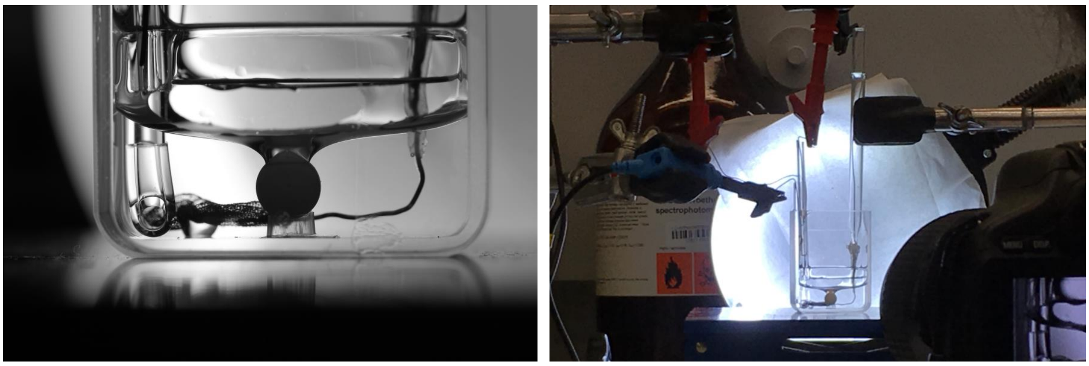

Figure 5.2: An example of an electrochemical cell, where the large surface area of the holm interface provides sufficient space for additional electrodes.

5.3 Choice of solid

5.3.1 Solid shape

The holm meridian can be formed around any axisymmetric object, including deformable objects such as floating sessile drops. SECTION 4.4.3 (pg. 47) discusses the difference between using the sphere compared to cylindrical objects. It is thought that the additional curvature offered by the sphere improves the accuracy of the technique. Nevertheless, with minimal alterations to the scaling procedure, the code could be adapted to fit any holm provided that the region of interest is clearly defined in the photograph.

5.3.2 Sphere size

Altering the size of the sphere alters the Bond number of the system. From a practical sense, it may be necessary to adjust the size of the sphere either to allow sufficient distance between the sphere and the wall, or to achieve a better ratio between the number of pixels of the holm and of the sphere. In CHAPTER 8, it is shown that the size of the sphere has no appreciable effect on the measured interfacial tension.

5.3.3 Sphere material

It is possible to adjust the chemical affinities between the fluids and the sphere by altering the sphere material. As an example, most of the work described in this thesis makes use of Teflon spheres for measurements with aqueous solutions. The strong hydrophobicity of the Teflon coating results in good curvature of the holm in these systems. Alternative materials (coatings, ceramics, metals) can be used to improve fitting with particular systems. Alternatively, they can be used to produce a raised, rather than submerged, holm. In general, it is desirable to use a material with low affinity for the phase which (mostly) surrounds the sphere. For example, when measuring the oil-water interface using a submerged holm, a hydrophobic material is used and the sphere pushed down into the aqueous phase. However, it is also possible to use a hydrophilic material and measure using the raised holm, as will be discussed in SECTION 5.7.1 (pg. 63).

5.4 Choice of cell

5.4.1 Cell shape

Theoretically speaking, the holm meridian describes the shape of an interface surrounding an axisymmetric object in an unbounded fluid. One could argue, then, that the cell in which the experiment is undertaken should also be axisymmetric to eliminate the effects of non-axisymmetric interfaces formed around the walls, particularly at the corners of a square cell. However, the optical distortion produced by cylindrical cells is prohibitive. Use of a square or rectangular cell is perfectly adequate, provided that the cell is sufficiently large to simulate an unbounded fluid. Later sections will show that even quite small cells can be used if enough of the interface is available for fitting.

5.4.2 Cell size

The effect of the sphere size relative to the cross-sectional area of the cell will be discussed in more detail in CHAPTER 8 with a case study on the alkane-water interface. Sufficient distance between the sphere and the wall will allow the holm to reach its horizontal asymptote, simulating an unbounded fluid. However, as will be shown, the measurement will work in smaller cells, provided enough length and curvature of the holm remains to achieve a unique fit. In general, a rectangular cell with sides of 2 – 5 cm should provide adequate space for measurement using spheres of a range of sizes (5 mm – 12 mm), although measurements with very small spheres inside 1 cm^2 cuvettes have been achieved.

The user should note, however, that smaller cells can be more susceptible to edge effects (where the interface curves to meet the wall, obscuring the section desired for measurement) than larger cells. Hartland [65] discussed issues with cell size in early work on the flotation of spheres, noting that relative distortion from the dissimilar fluids increased when the cell size became unduly large. Additionally, the focal depth of the camera may result in increased blurring along larger interfaces, although a larger aperature may be enough correct this.

In terms of depth, the only requirement is to sufficient height that the sphere can be depressed sufficiently for good curvature. The analysis itself is independent of the quantity of fluid involved.

5.4.3 Cell material

The choice of material for the cell is left to the discretion of the user, however good optical properties are necessary for this technique. The cell should not be affected by the solutions used inside it. For example, plastic cells susceptible to leeching would be a poor choice for measuring strong organic solvents.

It is generally desirable that cell material (hydrophobic/hydrophillic) be chosen such that the bulk interface meeting the wall curves *away* from the holm around the sphere. This will ensure that the interface is not obscured by curvature at the wall. This issue is highlighted in FIGURE 5.4.

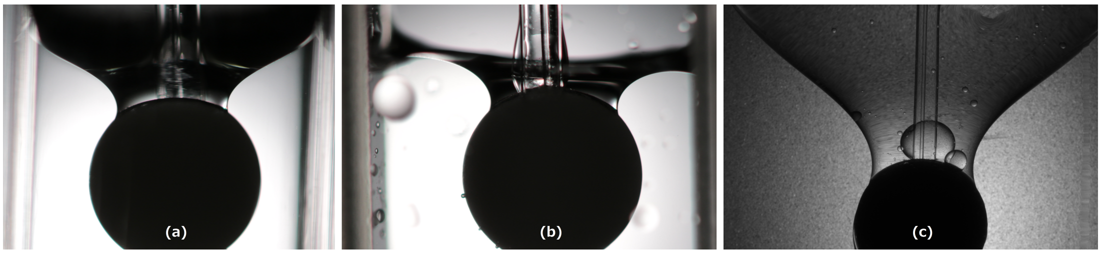

Figure 5.3: (a) DTAB/water and (b) DTAB-Gd/water interfaces measured using a 5.5 mm Teflon sphere in a 1 cm^2 cuvette. (c) Silicone/water+$FeCl_3$ around a 9.33 mm ball in a 3 cm × 3 cm cell. Due to the lower surface tension of the magnetic surfactant DTAB-Gd (b), the interface reaches the horizontal asymptote, compared to (a). In contrast, the extremely small density difference in (c) results in a large capillary length, meaning that the interface does not approach the horizontal asymptote even in the larger cell.

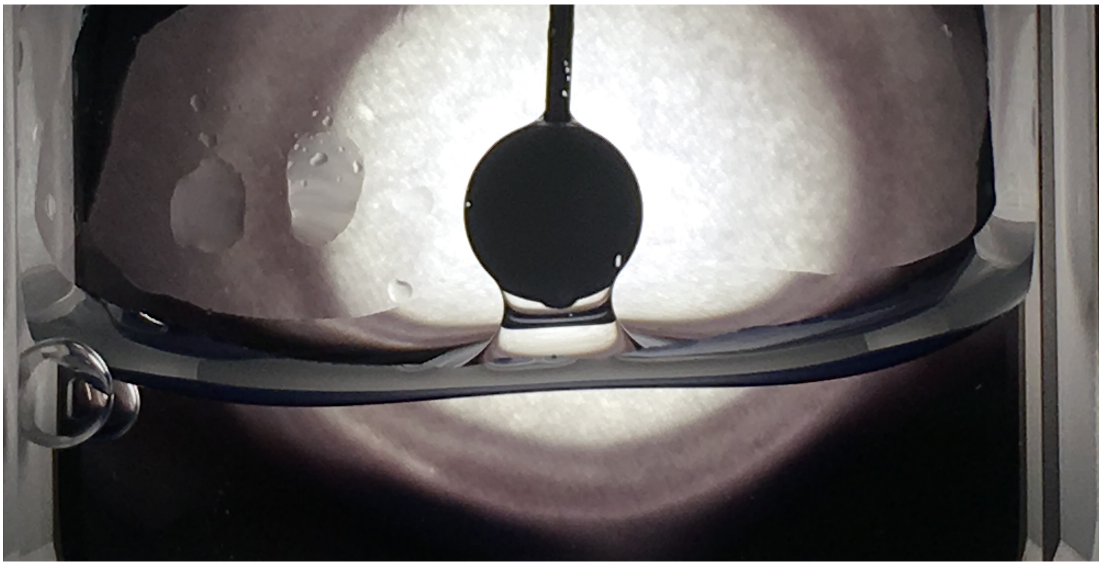

Figure 5.4: Measurement of an aqueous and dense organic phase in a silanised (hydrophillic) glass cell. The raised holm is obscured by the fluids rising towards the wall.

5.5 Improving the measurement accuracy

5.5.1 Curvature of the interface

Clear images with strong fore- and background differentiation are crucial for successful analysis. Further, it is important to have a clear edge visible over as long a length as possible. Where at all possible, the sphere should be lowered until the dense fluid rises above it, creating a “neck". In the same way that larger, more deformed pendant drops improve the accuracy of ADSA, the greater deformation at the interface improves the accuracy of this technique. The use of the highly hydrophobic Teflon sphere maximises this effect in the oil-water system, and appropriate hydrophobic or hydrophillic solids should be used depending on the fluid system and the desired direction of curvature.

5.5.2 Adjusting the penetration depth

The extent of curvature can be adjusted simply by altering the depth of the sphere. This is one area where the sphere is vastly superior to a long cylindrical object, as the interface is pinned to the top of the solid. The height can be adjusted by one of several methods:

- Moving the string or tube to which the ball is affixed,
- Fixing the sphere and moving the cell up and down (shown in FIGURE 5.5),
- Altering the volume of the dense phase to raise or lower the interface.

Depending on the relative densities of the fluids, the sphere (or other solid) may float, making it difficult to adjust the height. This can be overcome by use of a suitable spacer, threading the sphere directly onto a rigid object (such as a needle or tube) or by weighting the end of the thread.

The upper fluid will detach from the sphere if the holm is depressed too far. In this instance, the holm can be recovered by reducing the penetration depth until the holm reforms. The depth can then be increased again until the desired curvature is reached.

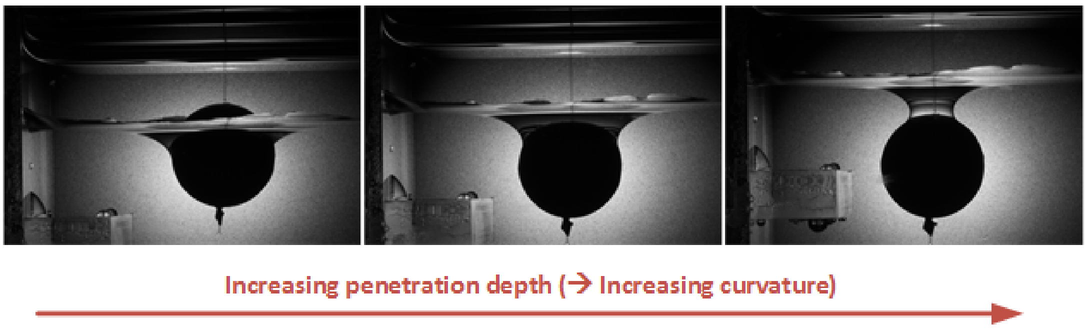

Figure 5.5: The effect of fluid height on the curvature of the bulk interface.

5.5.3 Impurities

It is important to note that interfacial tension is highly susceptible to the presence of impurities, and hence all equipment should be cleaned carefully before use. Soaking equipment in alcohol for 15 minutes to remove traces of organic components is recommended.

5.6 Image quality

5.6.1 Lighting

One of the key differences between pendant drops and the holm interface should be noted at this point. Namely, the complexity of the image and the subsequent difficulties in analysis. When analysing a pendant drop, it is quite possible to obtain a strong contrast between the foreground (the drop) and the background, particularly in the calculation of surface tension where the background is air. Accordingly, it is fairly straightforward to develop a program to distinguish the drop from the background and successfully determine the edges of the drop. The solid substrate adds and extra dimension of difficulty for the analysis of sessile drops, particularly around the contact line. This issue is typically dealt with in the same way as identifying the needle in pendant drops – by specifying the areas of interest to limit the program's search area.

The holm meridian introduces a few additional issues, predominantly associated with the nature of curvature of the interface. The depression of the meridian causes much of the light to be trapped within the conical shape of the interface. Accordingly, it is possible to direct the back light such that very little light escapes from the "light" fluid to reach the camera. The result is that while both fluids may be clear and transparent in bulk, the upper "light" fluid appears dark or black in the photograph. The sphere, being a solid and opaque object, will also appear as a black region in the image.

However, reflections are a common issue with images of the holm, causing the edge to be broken by blurring or new “edges” produced by rapid changes in pixel intensity. While these issues will be discussed in depth in the following chapter, along with code modules designed to address them, suffice to say that images with strong contrast between the fore- and background, and without reflections or blurring along the edge of the holm, will provide for the simplest and most robust analysis. The user may find that a diffuser placed between the cell and the light will significantly improve the quality of the images.

5.6.2 Image resolution

Good resolution is paramount for accurate analysis. The greater the number of pixels around the holm, the more accurately the edge can be located in the image. The number of coordinate points used for fitting can be adjusted in the program: reducing this will reduce the time required for the program to run. An image with good resolution may have 500 - 800 edge coordinates per side, and processing all of the points will require significant computing time. Using the default parameters, MATLAB will fit the edge using 10 random selections of 50 points.

5.6.3 Length of the holm

Ideally, the image will capture the entire length of the holm, up to the point where the horizontal asymptote is reached. This may not always be possible, and the program will attempt to fit any length of the holm. However, if the captured edge is too small, the interface will resemble the arc of a circle. Much like a spherical bubble, this arc has minimal distinguishing features, and several combinations of the interfacial tension and characteristic length may provide reasonable fitting, leading to error. If the image captures a larger area, the areas of interest can be explicitly defined.

In contrast, there is no need to capture the entire sphere in the photograph. Only just over half of the edge is required for the program to make an accurate fit to the circular profile.

5.7 Adaption for specific applications

5.7.1 Coloured or opaque solutions

As with pendant drops, a clear image of the interface is required. An inspection of the holm interface will reveal that one phase extends into the other. If it should extend into a dark or opaque solution, the interface will no longer be visible. This is illustrated in FIGURE 5.6(a), where the small submerged holm is obscured by the dark fluid. Measuring coloured systems requires careful selection between use of the raised or submerged holms, to ensure that the continuous phase is clear. In FIGURE 5.6(b), a raised holm has been formed, causing the dark fluid to extend into the clear alkane layer. A clean fit is easily obtained.

As stated above, changing the chemical affinities of the sphere can facilitate formation of raised or submerged holms.

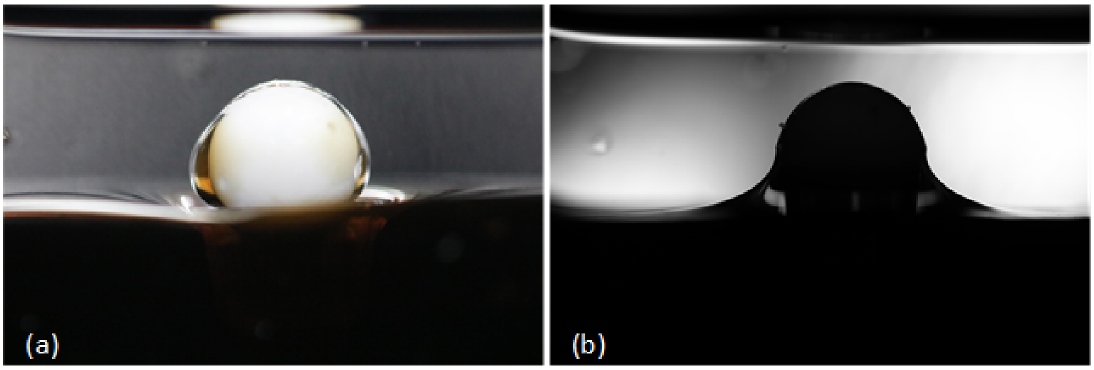

Figure 5.6: The (a) submerged and (b) raised holms formed between a dense, coloured ferro-fluid (10% in water) and decane around a ceramic sphere.

5.7.2 Dynamic (multiframe) analysis

Dynamic analysis can be achieved using movies or sequences of images. The program can accept movies as inputs and will proceed to break them down into sequence of images. The sequences can be analysed at any regular interval (i.e. every 10^{th} image.) No further adaption is required for dynamic analysis. The program will use the same inputs for each image, including the areas defined for the location of the interface. Consequently, it may be necessary to adjust these inputs partway through the run if the interface moves substantially. The program will also accept a file containing temperature readings, which can be used to calculate fluid densities if desired.

While there is no difference in the running of the program between images fed as a movie and as a sequence of images, the user may wish to remember that a sequence of images will likely offer better resolution, and this may impact their choice of medium.

5.8 Chapter summary

This chapter describes the experimental component required by this technique. Aside from a description of the technique itself, this section details some specific points to be taken into account to improve the images and thus facilitate image analysis.

A significant benefit of this technique over the pendant drop method is the stability offered by deforming the bulk interface. Later chapters detail how this method has been used to successfully measure the dynamic interfacial tension in moving or vibrating samples and for measurements over long periods of time (up to several days).

CHAPTER 6

Applying the principals of drop-shape analysis

6.1 Chapter overview

This chapter details the program methodology that analyses the experimental data (images and temperature readings) for the calculation of interfacial tension. The edge detection methodology is explored in detail in explained in CHAPTER 7, and hence only an overview is given here. Similarly, aspects relating to the experimental methods were detailed in CHAPTER 5. The fitting methodology encompassed by the MATLAB program and the Excel macros used for filtering and visualisation after dynamic analysis are explained in this chapter. The code to which this chapter refers is available in full in the extended appendices.

6.2 Program overview

The main program is written for the MATLAB programming environment, the output of which is a data table containing image information, the temperatures and densities used for the frame, shape factor, fitting errors and, of course, the interfacial tension. While the text file output from MATLAB is an unwieldy method to present the data, it ensures that a record of the user's inputs and version information for the program is maintained in a small file format along with the original output of the analysis. For further analysis, the data is exported to a macro-enhanced Excel spreadsheet which allows the user much more freedom in interacting with and visualising the data. The spreadsheet also doubles as the secondary and interactive filtering mechanism, allowing the user to easily identify and discard unacceptable fittings from MATLAB.

An overview of the process is given in FIGURE 6.1.

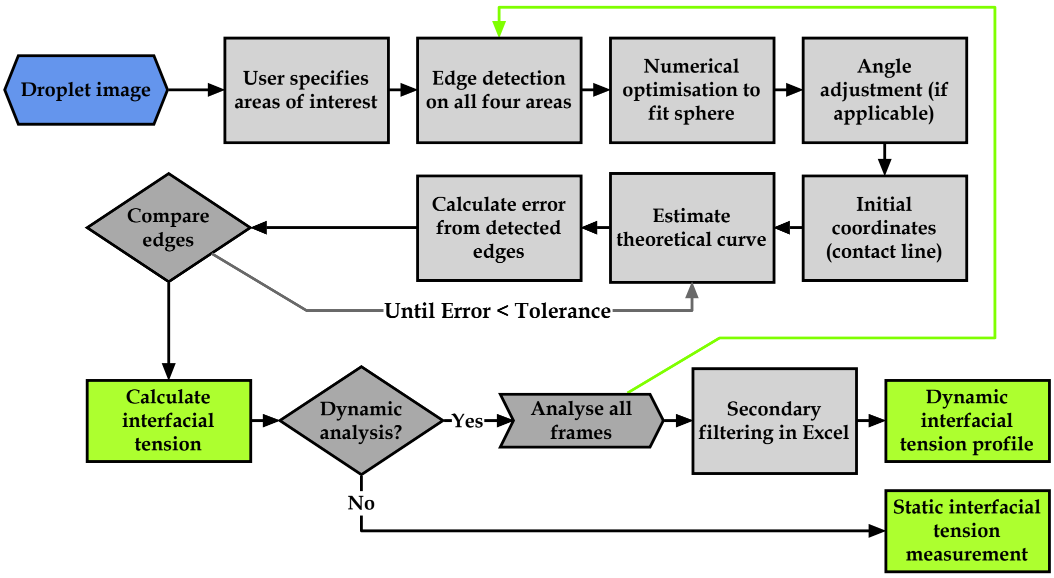

Figure 6.1: An overview of the program and its basic methodology.

The MATLAB program comprises of a series of graphical user interfaces (GUIs) which are linked as described below. The names here correspond to the file names in the MATLAB code.

- *HolmMainGUI*: The main (outer) GUI. This GUI should be called first – all of the other GUIs can be called from it. This GUI allows the user to specify the image and temperature files being used, define the fitting areas, fluid densities and scaling, control the optimisation and which frames to analysis, and start the analysis.

- *Timeline*: This GUI was built to handle temperature data from the AMOTH FL-2000 optical themometer and convert it into a file type readable by the program. If the temperature data already exists in the folder in the correct form, there is no need to call this GUI again. As the program is modular, a different program could be written to handle data specific to the user's experiment. Alternatively, the run can be analysed at a constant temperature.

- *ThreshGUI*: An interactive way to adjust the parameters for edge detection and image analysis. The GUI generates an output file that stores the specific thresholding data that is later called by the main program.

- *AngleGUI*: This GUI can be used to calculate the image tilt based on the detected coordinates or to enter an adjustment angle manually.

An *identifer* is a unique name chosen by the user to save the data for a particular run. Data files corresponding to that identifier are used to save inputs, thresholding values *etc.* and are called for each frame in the subsequent analysis. It is also possible to reuse this information for a later run if desired. Each new analysis will create a new folder named with the *identifier* and a unique time/date string.

6.3 Machine requirements

The MATLAB programming environment was chosen for its extensive libraries as well as optimization and image processing abilities. The MATLAB version of the program can be run with a standard 2014b installment or later, and requires the **Image Analysis** and **Curve Fitting** toolboxes. A stand-alone version produced using the MATLAB **Compiler** allows the program to be deployed on Windows machines without MATLAB being installed. The secondary filtering macros were written for Excel 2007+.

6.4 Program feeds

The experimental data fed to the program is minimal. Images can be fed as either still frames or movies. Movies are broken down into individual frames using MATLAB **videoreader** and the analysis proceeds on a frame-by-frame basis. The first and last frames as well as the sample interval can be specified through *HolmMainGUI*. This is done in terms of frame numbers, which the code will then convert into a time-stamp based on the defined frame rate. The fitting produces a dimensionless shape factor which is converted to practical units (mN/m) through:

- *The density difference between the two fluids*
  The density difference is a key-parameter of the Young-Lapalace equation relating the curvature and interfacial tension. The user can define either static densities or a density function to calculate the density based on the defined temperature. In the latter case, a static or dynamic temperature profile is also required.

- *The scaling or length-conversion factor*
  The default scaling factor is determined from the user defined width of the sphere (in mm) and the equivalent width in pixels calculated by the program. The reasoning behind the choice of the sphere was outlined previously in SECTION 5.3 (pg. 56). However, from a purely theoretical basis, the only requirement is the ability to convert the image from pixels to standard units of length.

6.5 Defining areas of interest

As the holm image is significantly more complex than a single pendant drop in a uniform field, areas of interest in the image are defined by the user at the beginning of the application. For example, parts of the image outside of the lit area can be discarded, and the left and right portions of the holm and sphere are separately defined. An interactive GUI allows the user to define these areas on a single frame, and the same information is then used on subsequent frames. Example images are shown in FIGURE 6.2.

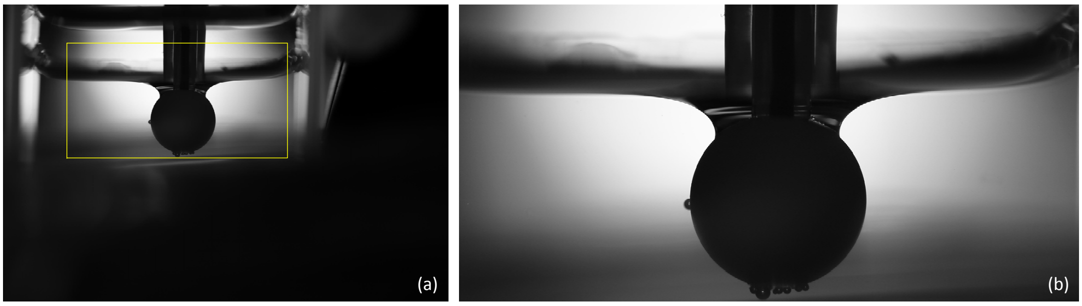

Figure 6.2: An example of the raw image file, (a) before and (b) after cropping. Only the lit area is required.

Once the unwanted areas have been cropped, the user is prompted to define four *search areas*. These consist of the holm meridian on the left and right side, as well as the left and right sides of the solid sphere, shown as blue boxes in FIGURE 6.3. The areas are defined once at the beginning of the analysis, and the same regions will be used for any subsequent frames using the same identifier.

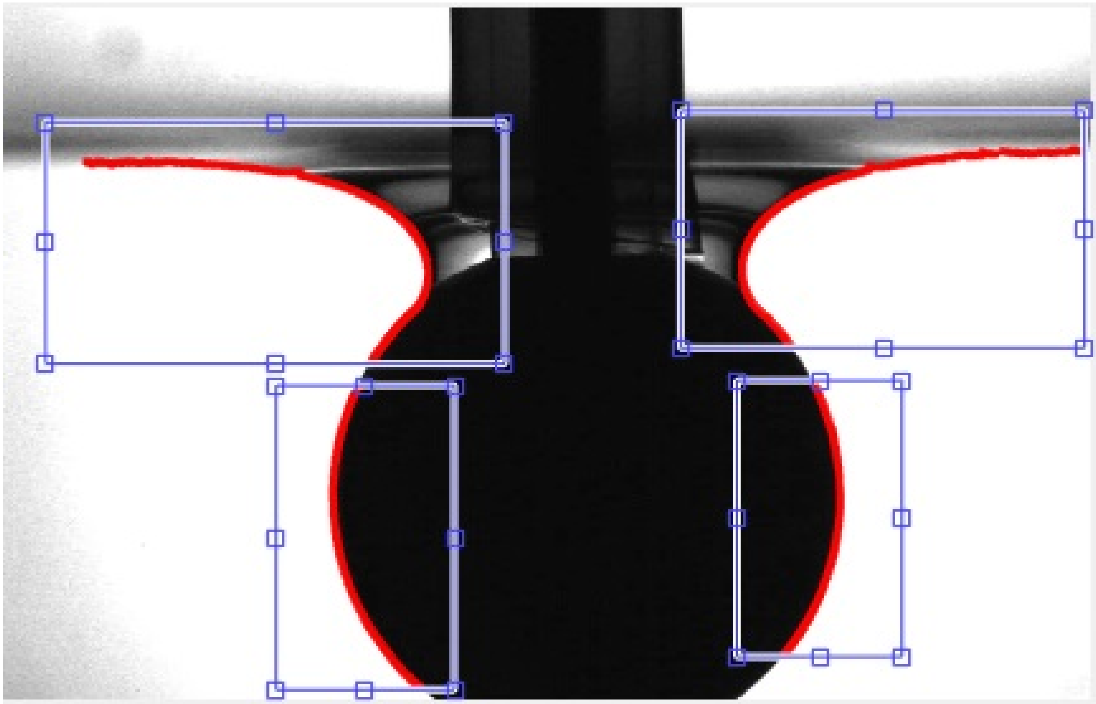

Figure 6.3: An image, with the detected edges overlaid in red, showing the user-defined regions to be used for analysis.

There are several advantages to explicitly defining the areas of interest in the images. To wit:

- processing time is reduced as edge detection is only required on small sections of the image,
- the requirement for the program to identify specific parts of a complex image is removed, allowing the algorithm to function correctly on a wider variety of images,
- to some extent, bubbles, reflections or other areas of the image that may affect the calculations can be discarded by the user, reducing the load on the image analysis algorithms.

6.6 Edge detection and image analysis

The specifics of the edge detection algorithms developed for this program will be discussed at greater length in CHAPTER 7. This section aims only to provide an overview so that the place of image analysis within the overall program can be understood.

Edge detection is undertaken only within the four defined areas, allowing to some extent the exclusion of such artifacts – such as reflections or bubbles – as may cause the analysis to fail. The inbuilt MATLAB **Edge** function, using the `Canny` parameter, returns the image edges to pixel resolution. Further filtering and masking is applied to remove unwanted edges from the result, as discussed in CHAPTER 7.

6.7 The spherical profile and contact line

While the theory behind the Young-Laplace equation equation holds firm for any submerged object, the program produced as a part of this doctorate is hard-coded to calculate the interfacial tension from a fluid meridian formed around a sphere, using the sphere as the basis of the scaling factor used to convert the meridian shape into the interfacial tension and to determine the starting point of the holm. There are several benefits from using a sphere rather than any other solid object: Firstly, the solid sphere provides a reliable, high-contrast centre to the the image. Secondly, the sphere's diameter as calculated by the program is unaffected by camera tilt, and furthermore it is not necessary to have the ball hanging ‘vertically’ for the same reason. Thirdly, and crucially for the technique, the sphere allows a contact angle greater than 90^o at the start of the holm meridian, increasing the length of the holm that can be fitted and improving the technique’s accuracy.

6.7.1 Fitting the spherical profile

The sphere is shown in the images as a two-dimensional projection – a circle. It is a simple matter to determine the circle’s width (in pixels) and centre point (x_0, y_0) by means of numerical optimisation using MATLAB’s constrained optimisation function **fmincon** with the condition that R be non-zero. The objective function for optimisation is

$$(x_i - x_0)^2 + (y_i - y_0)^2 = R^2 \tag{6.1}$$

where (x_i, y_i) are the detected points along the circle’s edge.

By this method, the radius and centre point of the circle that best satisfies the detected coordinates is determined. The radius is needed to determine the scaling factor for the image (discussed in SECTION 5.2.2 (pg. 54)). The center of the circle (x_0) marks the center of the holm. The coordinate system for the holm has its origin at (X_0, Z_∞), where Z_∞ is the height of the unbounded fluid. To differentiate the two sets of coordinates, henceforth the sphere will have coordinates (x, y); and the holm shall have coordinates (x, z), and (X, Z) in reduced form.

6.7.2 Adjusting the image angle

Where necessary, angle adjustment can be applied to the image to compensate for camera tilt. As the adjustment results in blurring and pixelation, the image itself is rotated only for display purposes. The actual adjustment is made on the holm coordinates after they have been extracted from the original image, ensuring maximum accuracy. Details for calculating the angle are given in SECTION 6.10.1 (pg. 78).

If the image is rotated α degrees around the image center (x_m, y_m), a coordinate (x, y) is rotated along an arc of radius R_r:

$$R_r^2 = (x - x_m)^2 + (y - y_m)^2 \tag{6.2}$$

Thus, rotating any coordinate (x, y) to the adjusted coordinate (x_a, y_a) or (x, z) to (x_a, z_a) gives:

$$\phi = \begin{cases} \frac{\pi}{2}, & \text{for } x = x_m \\ \arctan\left(\frac{y-y_m}{x-x_m}\right), & \text{otherwise} \end{cases} \tag{6.3a}$$

$$x_a = \begin{cases} x_m - R_r \cos(\phi + \alpha), & \text{for} < x_m \\ x_m + R_r \cos(\phi + \alpha), & \text{for} \geq x_m \end{cases} \tag{6.3b}$$

$$y_a = \begin{cases} y_m - R_r \sin(\phi + \alpha), & \text{for } y < y_m \\ y_m + R_r \sin(\phi + \alpha), & \text{for } y \geq y_m \end{cases} \tag{6.3c}$$

These adjustments are made on all detected coordinates and on the center of the sphere: $(x_0, y_0) \to (x_{0a}, y_{0a})$.

From this point forth it will be assumed that all coordinate points have been adjusted appropriately. Accordingly, the subscript '*a*' will be dropped.

6.7.3 Identifying the contact line

As described in CHAPTER 2, the Young-Laplace equation is a set of three differential equations. However, unlike the pendant and sessile drop scenarios, there is no common starting point for integration – the curve is strongly dependent on the location of the origin. In the past, the accepted method to solve the integration issue was to use the point X^*, just before the horizontal asymptote,[76] as described in SECTION 4.2.3 (pg. 42). However, with the improved computing and digital image processing techniques available today, it is possible to obtain a good estimate of the contact point that can be used as the starting point for integration.

As was shown in FIGURE 4.1, the contact *line* is the circle in the horizontal plane where the liquid-liquid meridian (the holm) comes into contact with the solid sphere. At the contact line, the position of both the sphere's circular profile and the holm's two dimensional profile are the same. In any given image, being two dimensional, two contact *points* are visible, one at each of the left and right extremes of the circle in the horizontal plane. Consequently, these points provide the initial coordinates from which to build the meridian profile.

6.7.4 Initial coordinates (contact point)

The contact point was originally determined by a uni-directional search from the circle side which continued until the detected coordinates of the holm deviated by more than a set tolerance from the the circle's descriptive equation. However, this method was found to fail frequently when bubbles or reflections introduced erroneous "edge" pixels around the neck. To avoid this issue, the program determines the coordinates with heights corresponding to the origin ($z = y_0$) and the top of the sphere ($z = y_0 + R$).

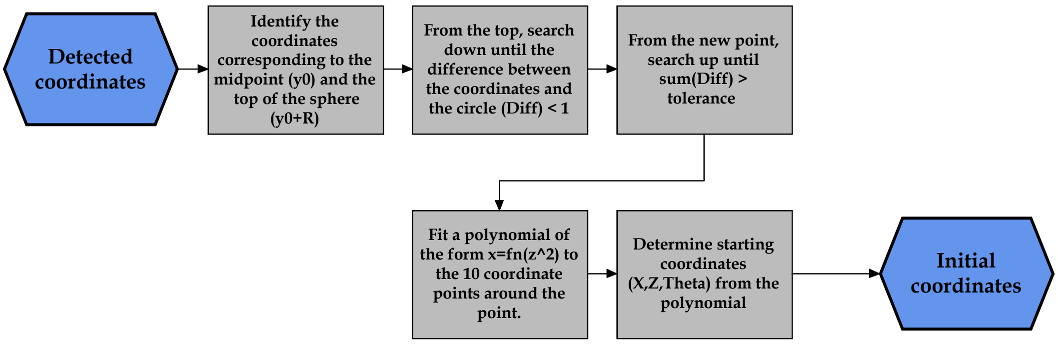

Figure 6.4: Flowchart showing the method to determine the initial coordinates to be used for integration of the differential Young-Laplace equation.

For a given edge coordinate (x, z), the x-coordinate of a point on the sphere profile (x_s) with the same height is given as:

$$x_s = \sqrt{|R^2 - (z - y_0)^2|} + x_0 \tag{6.4}$$

The difference between the sphere profile and an edge coordinate is then defined simply as:

$$\Delta_{\text{Diff}} = x - x_s \tag{6.5}$$

Note that the absolute value is not taken – the direction of the difference is important. A positive difference means that the edge coordinate lies on the outside of the sphere. Any point on the inside implies an issue with the detected edges and should not be considered as a viable starting point.

As outlined in FIGURE 6.4, the program searches along the detected coordinates between $z = y_0 + R$ and $z = y_0$. Searching down from $z = y_0 + R$, x_s and Δ_{Diff} are evaluated for each coordinated until $\Delta_{\text{Diff}} < 1$ pixel. The search direction is then reversed, and x_s and Δ_{Diff} are evaluated again at each point until $\sum \Delta_{\text{Diff}}$ exceeds the user-defined tolerance. A default value of 2 pixels is used, as the detected coordinates have only pixel-level resolution. The point so determined is considered as the first coordinate of the holm. Any coordinates below this point are removed.

The progression from the sphere to the holm is smooth. A second order polynomial of the form $x = \texttt{fn}(z, z^2)$ is fitted to n points around the detected coordinate (default is 5 above and 5 below). The initial coordinates at the contact line, $(x, z, \phi)_{CL}$, are determined by evaluating the polynomial at the height of the mid point, where the contact angle is given by

$$\tan(\phi) = \frac{\mathrm{d}z}{\mathrm{d}x} \tag{6.6}$$

X and Z are expressed in the reduced coordinate system by the relationships:

$$x = \frac{X}{\beta} + x_0 \tag{6.7a}$$

$$z = \frac{Z}{\beta} + z_\infty \tag{6.7b}$$

where reduced coordinates (X, Z) are scaled using the factor β to the image coordinates, (x, z) (in pixels). z_∞ is the height of the unbounded fluid and x_0 is the horizontal coordinate of the vertical axis running through the center of the sphere.

6.8 Theoretical profiles

6.8.1 Integrating the Young-Laplace equation

The three differential equations describe the changes in angle and the normal directions X and Z in terms of the distance along the meridian profile, S. Given $\lambda = 0$ for the holm meridian, (2.16) is simplified to (6.8). The case handling $X = 0$ (6.8a) is derived from l'Hopital's rule.

$$\frac{d\phi}{dS} = \begin{cases} Z - \frac{\sin\phi}{X}, & \text{for } X \neq 0 \\ -Z, & \text{for } X = 0 \end{cases} \tag{6.8a}$$

$$\frac{dX}{dS} = \cos\phi \tag{6.8b}$$

$$\frac{dZ}{dS} = \sin\phi \tag{6.8c}$$

EQUATION 6.8 describes the theoretical shape of a fluid interface without reference to a shape factor, unlike the case of pendant or sessile drops where the shape factor is included explicitly in the reduced form of the equation. In the case of the holm, however, the parameter β is included implicitly, as it relates the reduced coordinates to the image coordinates through (6.7).

EQUATION 6.8 is integrated numerically through MATLAB's **ODE45**, which solves systems of non-stiff first-order differential equations. (6.8) is integrated from the initial coordinates determined in SECTION 6.7.4 (pg. 73), over the span $0 \rightarrow 3\pi$.

6.8.2 Trimming the theoretical curve to the feasible region

The theoretical profile determined from the integration extends past the feasible region as the curve doubles back on itself several times. Using the MATLAB function **findpeaks**, the maxima of the curve can be determined. The theoretical profile is trimmed at the first such peak, which is the extent of the feasible region. The reduced coordinates (X, Z, θ) are scaled to the image coordinate system (x, z, θ) (in pixels) as defined in (6.7).

6.9 Numerical optimisation

The program uses constrained numerical optimisation to determine the best parameters to match the theoretical equation to the detected edge. To avoid confusion, let the coordinates of the theoretical interfacial shape calculated from (6.8) be referred to as (x_n, z_n), and the coordinates of edge detected from the image be (x_i, z_i). Both sets of coordinates correspond to the image, and have units of pixels.

6.9.1 Calculating the fitting error

The fitting error (6.9) describing the difference between any point on the theoretical curve (x_n, z_n) and the detected edge (x_i, z_i) is given as:

$$E^2 = (x_n - x_i)^2 + (z_n - z_i)^2 \tag{6.9}$$

However, the detected edge is a series of coordinates detected from an image and the theoretical profile is obtained by numerical integration of the Young-Laplace equation. Consequently, both curves are a set of discrete points: (x_i, z_i) and (x_n, z_n) where the two numerical indices, i and n, do not necessarily correspond to equivalent points on the curve.

This issue is addressed by the use of MATLAB's cubic spline interpolation function, **spline**. A cubic spline is applied to the coordinates of the theoretical profile (x_n, z_n) and evaluated at vertical coordinates matching the detected edge:

$$(x_{si}, z_i) = \texttt{spline}(x_n, x_n) \text{ evaluated for } z = z_i \tag{6.10}$$

The result is the original curve interpolated to produce points matching the vertical position of the detected coordinates. The error function across all of the coordinates then simplifies to:

$$E_{\text{fit}}^2 = \sum_i (x_{si} - x_i)^2 \tag{6.11}$$

This error quantifies the difference between the theoretical curve, obtained by integrating the Young-Laplace equation using estimated initial conditions, and the actual edge detected from the image.

6.9.2 Optimisation constraints

The error calculated in (6.11) is minimised through MATLAB's constrained optimisation function **fmincon** using the following variables and constraints:

- β: $0.1\beta < \beta < 5\beta$
- θ_{CL}: $0.95\theta_{CL} < \theta_{CL} < 1.05\theta_{CL}$

- z_∞: $-100z_\infty < z_\infty < 1.5z_\infty$ *

Note that while θ_{CL} is determined from the image, it is allowed to vary $\pm 5^{o}$ to improve the fit. The large bound on z_∞ is to mange the rising tail.

* Note that, by convention, coordinate (1,1) in an image is at the top left corner. Hence, all of the z-coordinates are actually negative when written in the image coordinate system in the MATLAB code. Consequently, the lower and upper bounds of z_∞ are reversed, and $-100z_\infty$ is actually above z_∞ if plotted over the image.

6.9.3 Calculation of interfacial tension

The interfacial tension (γ) is calculated from the reducing factor (β) returned from the optimisation procedure, related to the interfacial tension by:

$$\beta = \frac{\Delta\rho g R^2}{\gamma} \tag{6.12}$$

The characteristic length, R, is the length of the submerged sphere.

6.9.4 Random coordinate selection and repeat analysis

A typical drop profile contains several hundred coordinate points. The computation time can be significantly reduced by taking only a small subsection of this set and repeating the analysis multiple times.[155] A selection of 20 points repeated 10 times was recommended by Cheng and Neumann,[41] although it was later suggested[48] that 50 points would provide improved accuracy. The program's default setting is a random selection of 50 points, although this value can be adjusted by the user. However, the number of repeats is handled differently.

Unlike sessile and pendant drops, the starting point of the holm is not fixed. In order to improve the fitting, the analysis is repeated from a number of different coordinates, meaning that a slightly different initial boundary is specified every N repeats for SL points. In addition, to ensure that the fitting is not biased by the estimated interfacial tension, the initial guess for β (calculated from the assumed interfacial tension) is varied between $\frac{1}{2}\beta$, β and 2β. An example of how the initial coordinates changes with each repeat is given in TABLE 6.1.

Table 6.1: An example of how the starting estimates and initial coordinates are arranged during repeat analysis. In this example, $N = 2$ (two repetitions at each coordinate), and the total number of repetitions will be $N \times SL$.

Repeat	1	2	3	4	5	6	7	...
β	$\frac{1}{2}\beta$	β	2β	$\frac{1}{2}\beta$	β	2β	$\frac{1}{2}\beta$	...
(X,Z)	$(X,Z)_1$	$(X,Z)_1$	$(X,Z)_2$	$(X,Z)_2$	$(X,Z)_3$	$(X,Z)_3$	$(X,Z)_4$	...

6.9.5 Initial filtering and error analysis

The fitting errors (E_{fit}), optimised parameters (β and z_∞) and interfacial tension from each repetition are stored in the output matrix. Once the analysis is complete, the fitting with the lowest overall error is deemed the best fit. Any fittings meeting the filter criteria $E_{\text{fit}} < 2E_{\text{fit, min}}$ are kept and used to calculate the average interfacial tension and 90% confidence interval. The final output of the program includes the fitting errors, optimised parameters and interfacial tension if the best fit; and the average fitting error and interfacial tension of the repeated runs.

6.10 Angle adjustment

6.10.1 Determining the tilt angle

Rather than determining the angle as an optimisation parameter, the angle is calculated from certain geometric considerations applied directly to the image. For a holm interface that is properly symmetrical and has sufficient curvature to produce a neck, the two closest points on the left and right sides of the profile should be horizontal. This provides an efficient method for checking the alignment of the camera and adjusting images if required. The user is also able to make manual adjustments if desired. This will be required if the interface was not completely axisymmetric (see FIGURE 6.7) or the holm did not show a clear necking region, or if the the edge in that region is obscured. In both of these instances, the nearest neighbour search will not return an appropriate answer.

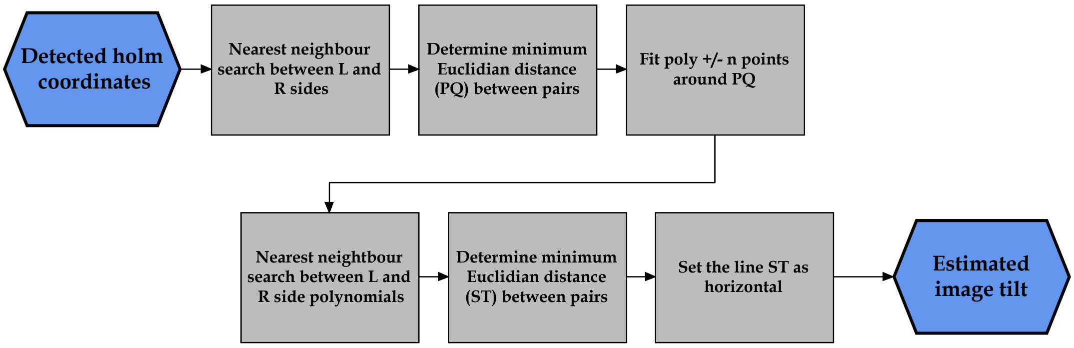

Figure 6.5: Schematic of the process used to estimate the image tilt.

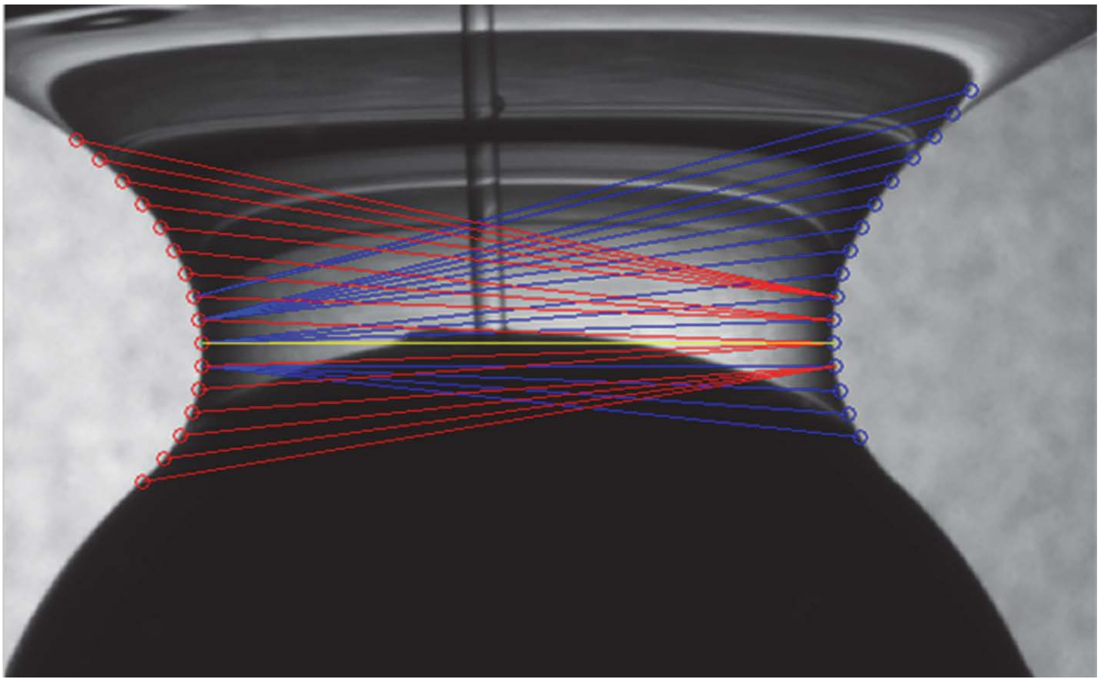

Figure 6.6: Result of the nearest neighbour search between polynomials fitted to the left and right sides of the holm.

6.10.2 Left- and right-side fitting

It is an assumption of axisymmetric systems that both sides of the two-dimensional profile are identical. In a key difference to pendant and sessile drops, the left and right "sides" are separated in a two-dimensional image. In practice, however, tilted images or the placement of the sphere off-center in the cell can mean that the left- and right sides of the image are not perfectly symmetrical, requiring different starting conditions for the two sides. The program fits the left and right sides separately, by the simple expedience of flipping the image and running the right-side analysis twice. The interfacial tension calculated from the two sides should be in good agreement. By capitalizing on the separated profiles of the two sides and analysing each side separately, the program uses the difference in the calculated interfacial tension as an internal check to ensure that experimental conditions and angle adjustments are properly taken into account.

Post processing on the analysis output flags frames where the difference between the interfacial tension calculated from the two sides exceeds a user-specified limit, prompting the user to check the edge detection or angle adjustment on certain frames. This is discussed further in SECTION 6.11.2 (pg. 81) as part of the multiframe filtering. If the program appears to be providing a good fit but the calculated interfacial tensions differ significantly, the user is advised to check that the image tilt has been properly accounted for and to make adjustments if required.

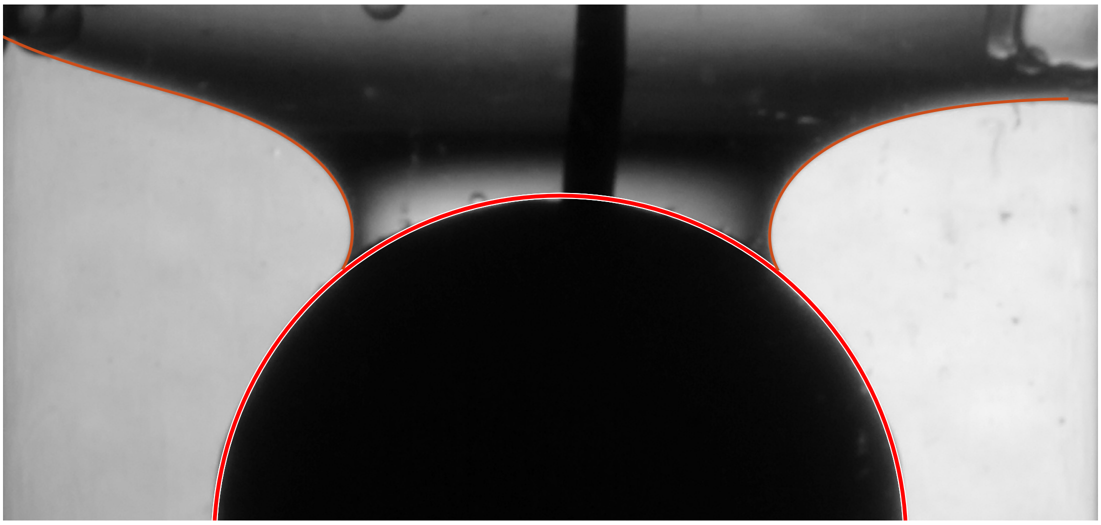

Figure 6.7: An example of fitting to an asymmetric image. Note that while the initial coordinates change, an effective fit is managed from both sides.

6.11 Adaption for dynamic (multiframe) analysis

6.11.1 Code adjustments

The program described in this research was developed with dynamic analysis in mind. As the camera is untouched for the duration of analysis and the interface is formed around a solid object, the vertical alignment can be considered constant throughout the experiment. Accordingly, computing time can be significantly reduced by determining the image tilt from a small subset of images and using this adjustment for all frames. The angle is then adjusted as described in SECTION 6.7.2 (pg. 71). This is a significant reduction on processing time compared to including the tilt angle as an optimisation parameter as is done with many drop-fitting programs.

Images for multiframe analysis can be provided as a sequence of images or as a MATLAB-supported movie. Thresholding parameters and search regions are also kept constant within a run. Once the most appropriate parameters are determined on a single frame, these parameters are automatically applied to each frame in the series. In some instances (i.e. if the contrast of the images changes), the user may wish to break a long run into shorter sections, using different thresholding parameters on different groups of frames. The user is freely able to choose the starting and finishing frames and the analysis interval.

Multiframe analysis can be conducted at a constant temperature (i.e. constant densities) or by importing a `.mat` file containing temperature data. The GUI *Timeline* was written to convert temperature data returned from the AMOTH FL-2000 fibre-optic thermometer into the appropriate file format. The temperature file in use can be visualised by selecting the `temperature` option on *HolmMainGUI*.

6.11.2 Multiframe filtering (Excel)

The text file produced by the MATLAB program from the multi-frame analysis can be exported into a macro-enable Excel file which provides efficient post-processing capabilities. In particular, the Excel file facilitates multi-level filtering, allowing failed or poorly fitting frames to be flagged and removed from the final set of data. The user can experiment with different filters regarding acceptable fitting error on the holm and the sphere, acceptable difference between the left and right sides, and whether or not to include independent sides when only one fitting is available.

6.12 Chapter summary

This chapter outlined the methodology and program underlying the holm measurement technique developed in this thesis. In a clear distinction between the holm method and ADSA on pendant and sessile drops, the Young-Laplace equation cannot be integrated from the origin. Instead, the starting point at the holm contact line is determined from the edge coordinates detected directly from the image. Computing time for the holm method is longer than required for the pendant drop, as the edge detection and parameter optimisation is more complex. The holm method was developed for multiframe (dynamic) analysis, and computing time is reduced by saving certain parameters (thresholding parameters, angle adjustments *etc.*) and applying them to all frames in the analysis rather than calculating the angle with each frame. Unlike ADSA-P, which uses up to five optimisation parameters,[72] and ADSA-NA, which uses seven,[83] the holm technique varies three parameters within a constrained numerical optimisation module, and varies the starting coordinate manually by repeating the analysis from several coordinates on a small subset of randomly chosen coordinates. By reducing the number of varying parameters, MATLAB's ability to determine the best fitting solution is improved. A final analysis on the various repeats produces a final, best-fitting solution which is returned as the interfacial tension of the frame. The program capitalises on the separate left/right profiles by fitting the two separately, providing an internal check to confirm that the image adjustment is appropriate. Additional filtering in a macro-enabled worksheet provides rapid flagging of poorly-fitting frames in multi-frame analysis and can be used to handle dynamic systems where bubbles or reflections may make some frames impossible to fit successfully.

CHAPTER 7

Image Analysis

7.1 Introductory remarks

It is widely accepted in the literature that all drop-shape techniques and their derivatives, from the original ADSA [42,124] and spinning drop [117,134] techniques through to new approaches such as TIFA [34] and contour methods, [112] are limited by the accuracy of the image analysis intrinsic to their methodology.

As early as 1990, when Cheng et al. [42] started the ongoing process to optimise the ADSA program to harness the steadily increasing computational power, it was known that drop-shape techniques rely more on accurate data points than on having a large number of points. Cheng et al. [42] proposed that as few as 20 highly accurate coordinate points would be sufficient for the ADSA program to successfully find the best-fitting Laplacian curve. Over twenty years later, Kalantarian et al. [84] identified edge detection over numerical integration and optimisation strategy as the key issue affecting accuracy in drop shape techniques.

In its earliest form, the ADSA program developed by Rotenberg et al. [124] required manual digitization, a labour-intensive and error-prone method of manually picking edge coordinates from a backlit page used in 1883 by Bashforth and Adams [12] in their pioneering work on the shape of drops. ADSA approach made one key distinction from previous methodologies, in that any random selection of edge points could be used, and no point along the interface had any special significance. This is in stark contrast to earlier approaches which required specific points, such as the drop apex or the maximum width, to be determined.[124] While there are obvious limitations to the accuracy of manual edge detection,[42] not to mention being completely unsuitable to the bulk treatment of large numbers of images or frames for use in dynamic analysis, the inarguable benefit of user-directed edge detection is the lack of uncertainty as to the location of the drop amongst all of the objects within the image. As digital edge detection methods progress, becoming increasingly accurate and sensitive, the shear amount of data available during image analysis begins to swamp a program's ability to identify the appropriate edge from within a complex image. The issue of edge detection is thus a key factor limiting the use of drop shape techniques in complex, realistic situations.

7.2 Edge detection - evolution from thresholding to gradient methods

7.2.1 Image thresholding

In the first iteration of the automated ADSA program, edge detection was carried out using simple black-white thresholding.[124] Black-white thresholding modifies a grayscale image by the simple expedient of returning all pixels below a certain value as 0 (foreground) and those in excess as 1 (background). Choosing the threshold can be a subjective matter, and the value will be different for each image. For a high-contrast image, where the fore- and backgrounds are strongly differentiated, the intensity histogram shows two strong peaks, light and dark, leaving a sparsely populated 'valley' of mid-range pixels. BW thresholding is a particular concern with holm images as there will typically be three major colour groups – the white background, black sphere and mid-range pixels around the holm interface itself. Pixels in the grey region may be categorised as fore or background objects depending on the threshold value.

A technique known as "Otsu's method" is generally regarded as a reliable approach to thresholding. This is the default method used by MATLAB. Otsu's method [106] provides a rapid thresholding technique that is fairly robust on images where the fore- and background are strongly differentiated by intensity. However, the pixels affected by changes to the threshold are more commonly found around the edge regions where intensity graduations exist. The effect of changing the threshold value is explored in FIGURE 7.1

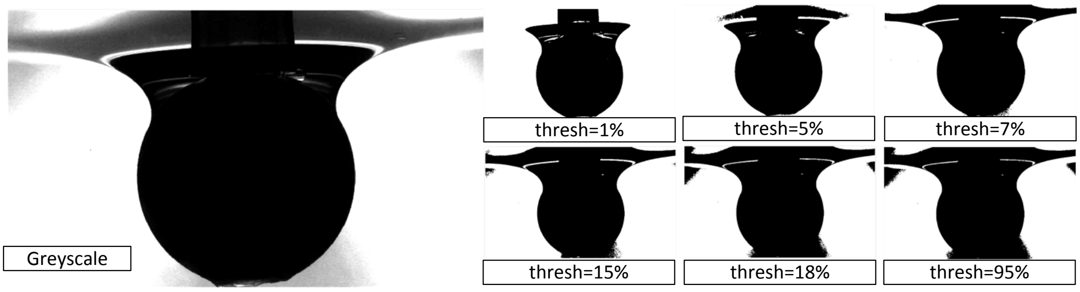

Figure 7.1: A greyscale image with clear contrast differences between the solid and upper fluid phases, and the effect of BW thresholding on such an image.

Subsequent work on on ADSA found simple BW-thresholding to be far too sensitive to lighting conditions [42] and hence unsuitable for accurate analysis. The position of an edge could easily move several pixels depending on the threshold value, with the effect increasing as the intensity change across the edge becomes less pronounced. Accordingly, most subsequent generations of ADSA programs switched to gradient edge detection methods.

7.2.2 Gradient edge detection

Unlike thresholding methods, gradient edge detection considers the intensity gradient across an image and labels as 'edges' any where the gradient exceeds a stated minimum value. This could be considered as a threshold method acting on the first differential of the intensity values in an image. The gradient under consideration is the gradient of a plane fitted to the pixels of an $n \times n$ sub-array of the greyscale image. [42] While a sub-array of any size can be used, Cheng et al. [42] chose a 3×3 array to compromise between accuracy and the increased computing time required for a larger array. The gradient of the sub-array is assigned to the central pixel, the process repeated for every pixel in the image, and pixels with sufficiently high gradients are identified as edges. Cheng et al.[42] found the Sobel method, which applies a weighted average to the least-squares regression used to fit the plane,[52] to be effective at identifying diagonal edges. Other 3×3 methods exist, such as the Prewitt operator[115], which is well-suited to detecting vertical edges. However, as diagonal edges far exceed both vertical and horizontal edges in number in most curved interfaces, the Sobel method was chosen as the most appropriate method.[42]

Later, the Sobel operator was replaced with the Canny method[36] for improved edge detection in noisy images. The key advantage of the Canny algorithm is the use of two parameters to determine whether or not a true edge has been detected. Firstly, gradient edge detection is done using the higher threshold. Edge detection then is conducted a second time, but at a lower threshold. Of the edges detected in the second run, only those that join an edge detected using the higher parameter are considered true edges. The remainder are discarded. In this way, the Canny algorithm is designed to minimise false edges, while still being capable of detecting weak edges using the lower threshold.[36]

7.3 Complex and noisy images

7.3.1 Determining the desired edge

While manual edge detection is a time consuming, highly inaccurate method to determine the drop edge coordinates, it has one distinct advantage in the ability to employ some of the most efficient pattern recognition software available – its human operator. This is particularly relevant to the current work as the nature of the holm interface implies a far more complex image than pendant and sessile drops, with a need to isolate two sides of the fluid-fluid (holm) and solid-fluid (sphere) interfaces from the midst of reflections and other noise. In order to be effective for multi-frame analysis, the program must be capable of discarding noise automatically or with minimal user interference. This is of particular importance in dynamic analysis where many frames are to be analysed.

Poor lighting, cloudy solutions or the presence of suspended particles or bubbles in the bulk solution adds additional complexity to automating edge detection in terms of noise and low contrast. Edge detection for the holm system developed in this research is significantly more complex than pendant or sessile drops as a simple foreground-background distinction is rarely present, and reflections occurring at the interface and intensity variations in the background interfere with detecting the correct edge in an image.

Consider a typical greyscale image, shown here in FIGURE 7.2 below. Applying Canny edge detection to FIGURE 7.2 results in a plethora of edges, as shown in FIGURE 7.3. While the patterns in FIGURE 7.3 are clear to the human eye, the challenge is to make them apparent to an algorithm with minimal operator input.

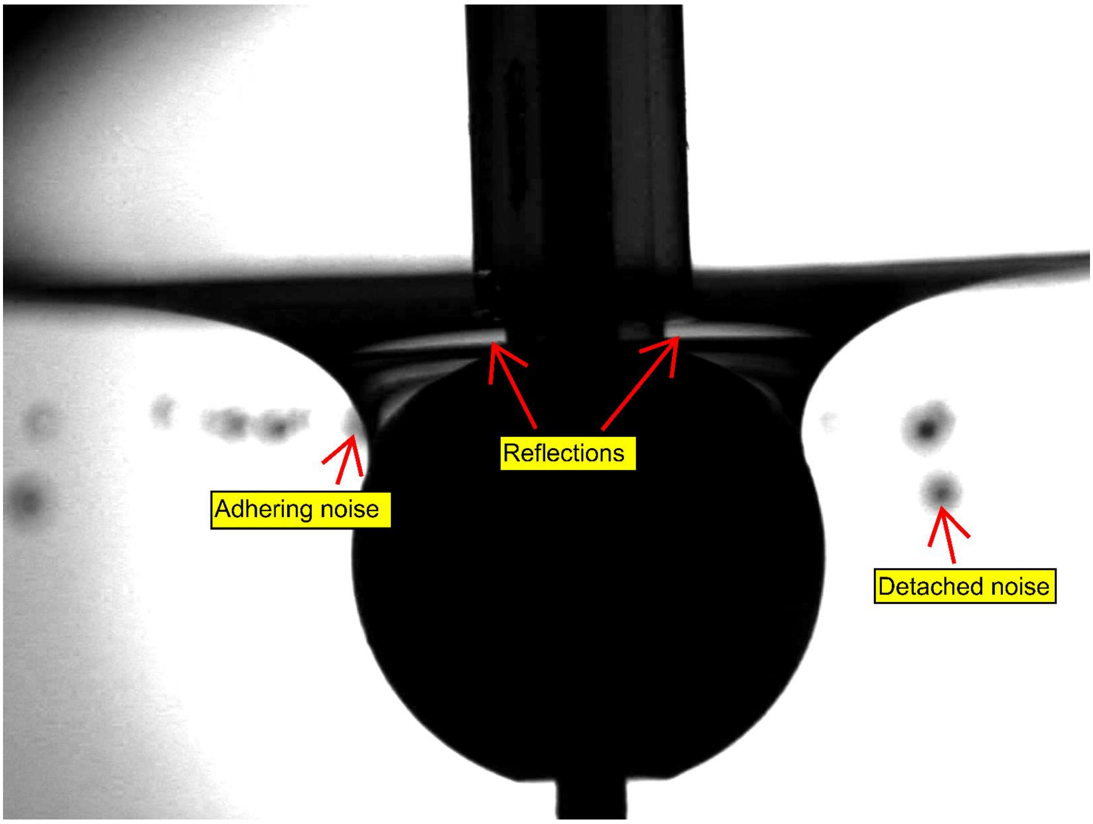

Figure 7.2: Original greyscale image showing reflections and both adhering and detached noise.

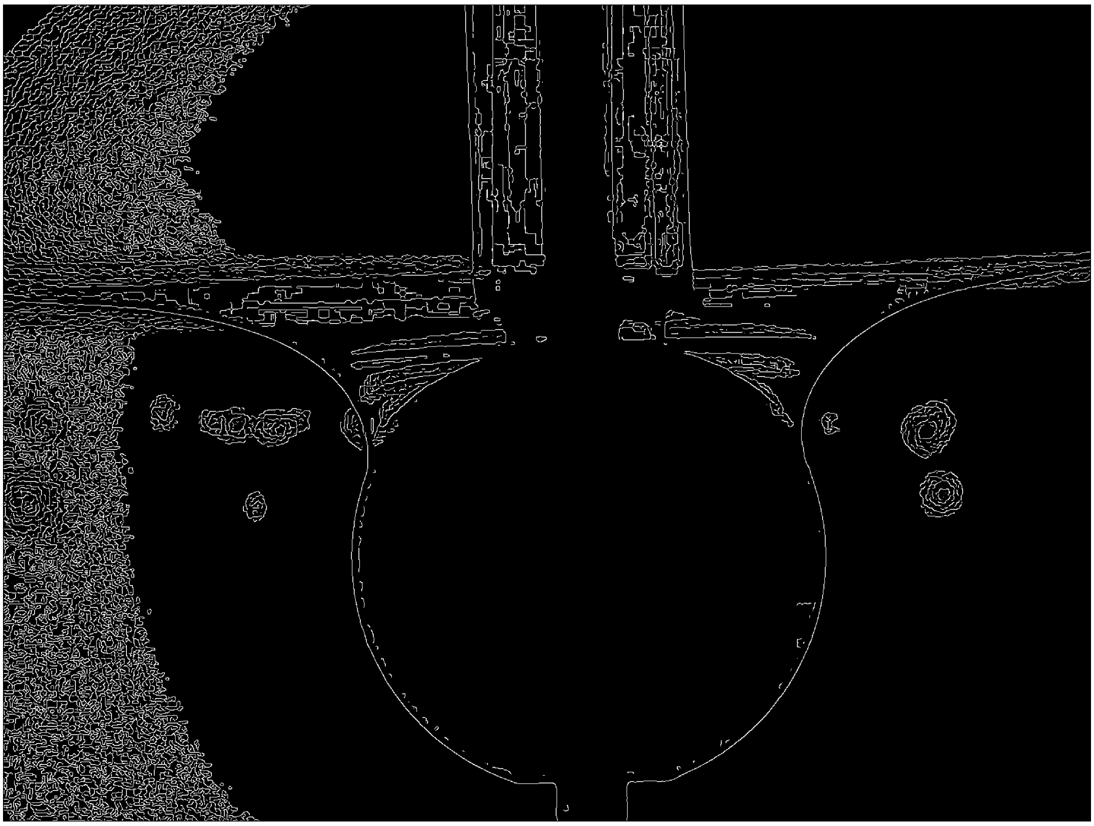

Figure 7.3: Results of Canny edge detection of the image in FIGURE 7.2 using the default thresholds. 3906 individual edge sections are shown in this image, the largest consisting of 1394 pixels and the smallest of only a single pixel.

While a glance at FIGURE 7.3 would suggest that the desired edge is in fact the longest continuous line, this is often not the case. Consider the edge matrix corresponding to FIGURE 7.1, shown in FIGURE 7.4, where the longest continuous line is highlighted in red. The abrupt contrast change due to the pair of horizontal reflections produces erroneous edges, and is a common issue in images with otherwise perfectly distinct edges. While the code offers the option to choose the longest line method as the edge detection algorithm, clearly an alternative method to determine the correct edge, one that does not require the edge to be a single continuous line, is required.

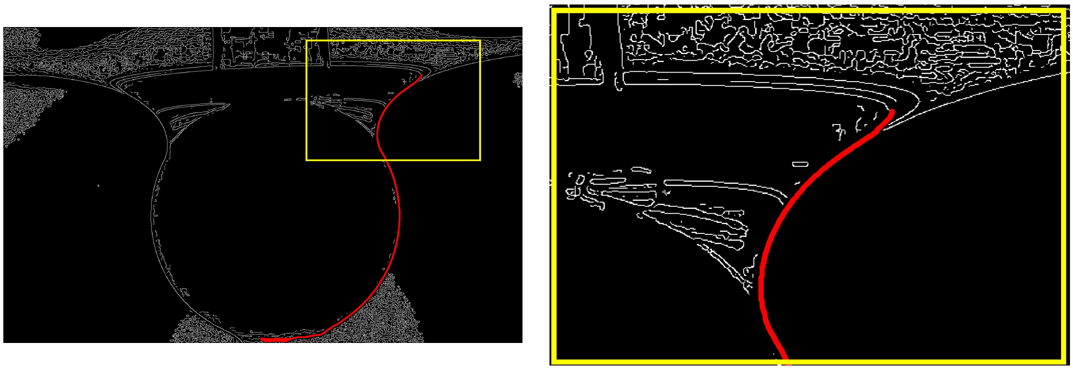

Figure 7.4: The result of Canny edge detection on an image where the interface is broken by strong reflections. The longest continuous edge segment is shown in red.

7.3.2 Categorizing noise

In their extensive work on interfacial studies with bovine lung fluids, Zuo et al.[156] encountered significant difficulties with murky or otherwise noisy images. Their work regarded sessile drops in turbid solutions, and the clarity or otherwise of the bulk fluid was a significant factor affecting the performance of ADSA in this situation. Three main types of noise were identified. Firstly, adhering noise that touches the drop edge, giving false or unrelated edges. This is further broken into two categories: bubbles or particles that are physically attached to the drop edge and those that are in the surrounding solution but appear connected in the image. Secondly, detached foreground objects, such as bubbles or large particles within the image frame. Lastly, reflections that induce large intensity gradients inside the drop image. All three types of noise were shown in FIGURE 7.2.

This chapter will show that noisy images can be effectively handled by successive use of masking layers, essentially telling the program which of the edges in FIGURE 7.3 and FIGURE 7.4 can be ignored. A major benefit of such a method is that, unlike making changes to the image itself (i.e. altering pixel intensity[156]), modifications are made only to the edge matrix, avoiding direct manipulation of the image. A combination of positive and negative masks are used. A "positive mask" refers to a mask which treats as true any edge pixels inside the defined region. *i.e.* :

```
{if(mask(i,j)==true, then accept edge pixel}
```

A "negative mask" defines a mask which will exclude any pixel inside the defined region. *i.e.* :

```
{if(mask(i,j)==true, then reject edge pixel}
```

"BW" will be used as a shorthand for binary or "black-white" images.

7.4 Identifying the appropriate edge within a complex image

All inputs to the functions in this section are managed by the graphical user interface *ThreshGUI*, which was created as part of the main program. *ThreshGUI* first creates a set of inputs based on the default parameters for the image. The user is then able to change the parameters interactively and monitor the effects on the detected edge. The parameters are saved and applied to all other frames in the dynamic analysis. Further input from the user is not required.

7.4.1 Defining regions of interest

One of the simplest methods of removing extraneous items from an image is cropping. In current versions of ADSA,[156] for example, the user specifies the location of the pendant drop to separate it from the needle. When the image is first loaded, the user is given the option to crop the image, if desired, and will then be prompted to define four regions of interest: the holm interface (right and left sides) and the sphere profile (right and left sides).

The sphere regions should contain only the sphere's edge, as all of the edge pixels in this region will then be used to determine the size of the sphere. However, the holm region can extend down to include part of the sphere's edge. The starting point for the holm will be calculated based on its deviation from the sphere, as will be described in the next chapter. The regions defined in this section are saved and are automatically used for the remaining frames in the analysis.

7.4.2 Edge detection

MATLAB boasts an inbuilt function, **edge**, for detecting edges. Optional commands allow specific methods to be chosen, such as the Sobel, Prewitt or Canny methods. The sensitivity of the functions is defined by additional thresholding parameters. The default parameters typically provide a very good results. However, it is possible for the user to specify their preferred parameters, and view their effects, in *ThreshGUI*.

The matrix showing the detected edges (such as FIGURE 7.3) is modified using the function **bwmorph** to thin the edges to a single line thickness

```
{imsk = bwmorph(Edges, 'thin', Inf);}
```

and then remove the branching points

```
{bpoints = bwmorph(imsk, 'branchpoints', 1);}
{imsk(bpoints)=0;}
```

7.4.3 Whole image mask (WIM) and perimeter mask

The first mask to be applied is dubbed the *whole image mask*, or WIM, from which is produced the *perimeter mask*. This mask is based on BW thresholding, and produces a binary mask of adjustable width that follows the perimeter of the foreground objects. Any edges outside of this region are discarded.

7.4.3.1 Black-white thresholding

The greyscale image is first converted in to a BW image by simple thresholding (**graythresh**). The default threshold is calculated by Otsu's method (SECTION 7.2.1 (pg. 84)). The user can change the parameter *BWadj* to alter the threshold as a percentage of the default value. (BWadj = 1.2 is 120% of the default threshold.) The choice of BWadj remains somewhat subjective, and the user is advised to alter the parameter through the interactive *ThreshGUI* to view the effects of the change. As a general rule, the user should set BWadj such that the holm becomes part of the foreground stretching across the length of the cropped image. This allows flood-filling operations (SECTION 7.4.3.3 (pg. 93)) to be used in the masking process. The thresholded image corresponds to the black region in FIGURE 7.5.

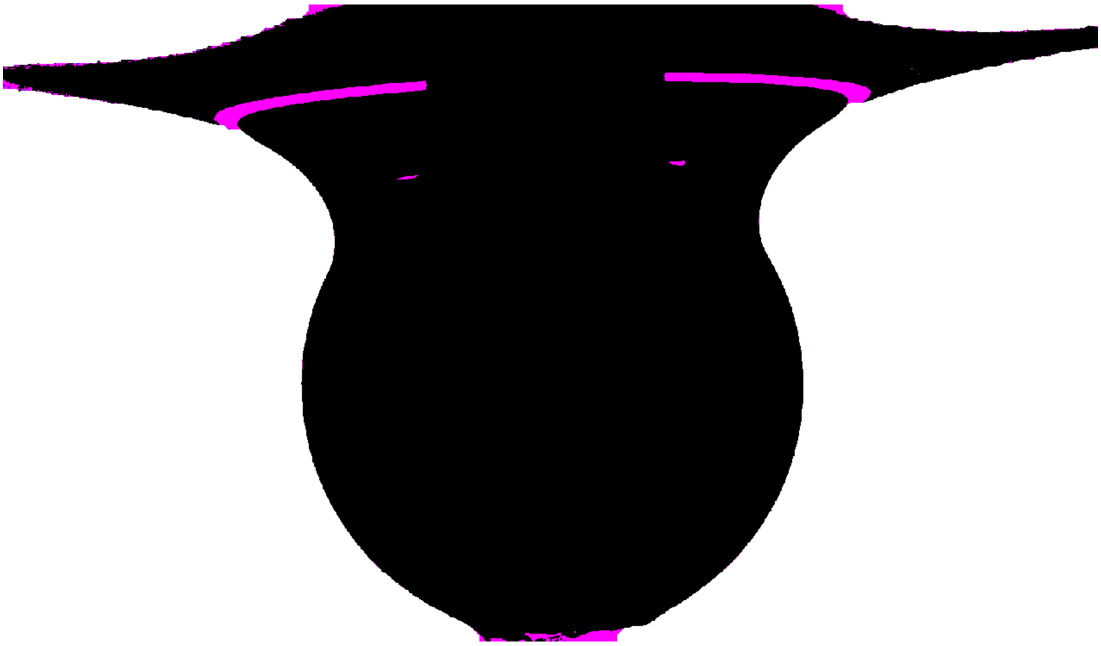

Figure 7.5: The initial binary image produced from thresholding FIGURE 7.1 (black) and the effects of the morphological "close" operation applied to the BW image, using a disk-shaped structuring element with a 15 pixel radius. The highlighted regions (magenta) were closed as a result of the manipulation.

7.4.3.2 Morphological operations

A common issue found with holm images is a bright reflection part-way along the interface, as shown in FIGURE 7.1. This reflection breaks up the continuous edge forming the holm, and is sufficiently common to render the simplest approach to edge detection – returning the longest single line identified by Canny edge detection – ineffective. These regions are isolated in the BW image by using the 'close' option in the **bwmorph** function. Image 'closing' is the end result of two morphological operations - dilation and erosion. The image is first dilated using the nominated structuring element. Dilation serves to extend components slightly in all directions, meaning that components that are discrete but quite close to each other can be made to overlap. With a sufficiently large structuring element, this can join the two edges into a smooth, continuous component. The second part of 'closing' – 'erosion' – returns the enlarged components to their original size (using the same structuring element) while leaving the 'bridge' formed by the dilation. This has the effect of 'closing' small gaps in the image. The user is able to define both the shape and size of the structuring element, but a disk-shaped element of 15 to 45 pixels is recommended. The effect of the 'close' operation is shown in FIGURE 7.5.

7.4.3.3 Holes and flood fill

The MATLAB function **imfill** is applied to fill in holes in the image. 'Holes' are defined as any background areas which are wholly surrounded by a foreground object. For our purposes, these 'holes' are typically either reflections or the light area above the top fluid. The user is also able to define starting points for flood fill. This can be used to fill white areas that extend to the top or sides but are bounded completely by the holm on the bottom. The 'fill' command passes over these areas as they are not bounded on all sides by a foreground object. Note, however, that this function is not applicable unless the holm edge is continuous on the lower side, hence a combination of image cropping and adjusting 'BWadj' should be used to ensure this. The binary image matrix corresponding to FIGURE 7.2 is shown in FIGURE 7.6, with the holes detected by MATLAB and points for flood-filling highlighted.

7.4.3.4 Component labelling

Discrete foreground objects are then assigned a unique numerical label in a process known as "component labelling". Component labeling is the process of assigning a unique numerical label to each component of an image,[87] where a "component" is a region of connected pixels. Component labeling is typically done on a BW image. In a background of white pixels, discreet groups of black pixels will be considered as a single component and assigned the same label (MATLAB function **bwconncomp**). A *label matrix* is a matrix of the same size as the image, where the value of each cell is the numerical label of the component to which the pixel belongs. The labels are generated with 8-point connectivity, which considers two pixels with touching faces or corners to be part of the same component. Generating a label matrix (MATLAB function **bwlabel**) gives access to MATLAB's **regionprops** function, which will re-

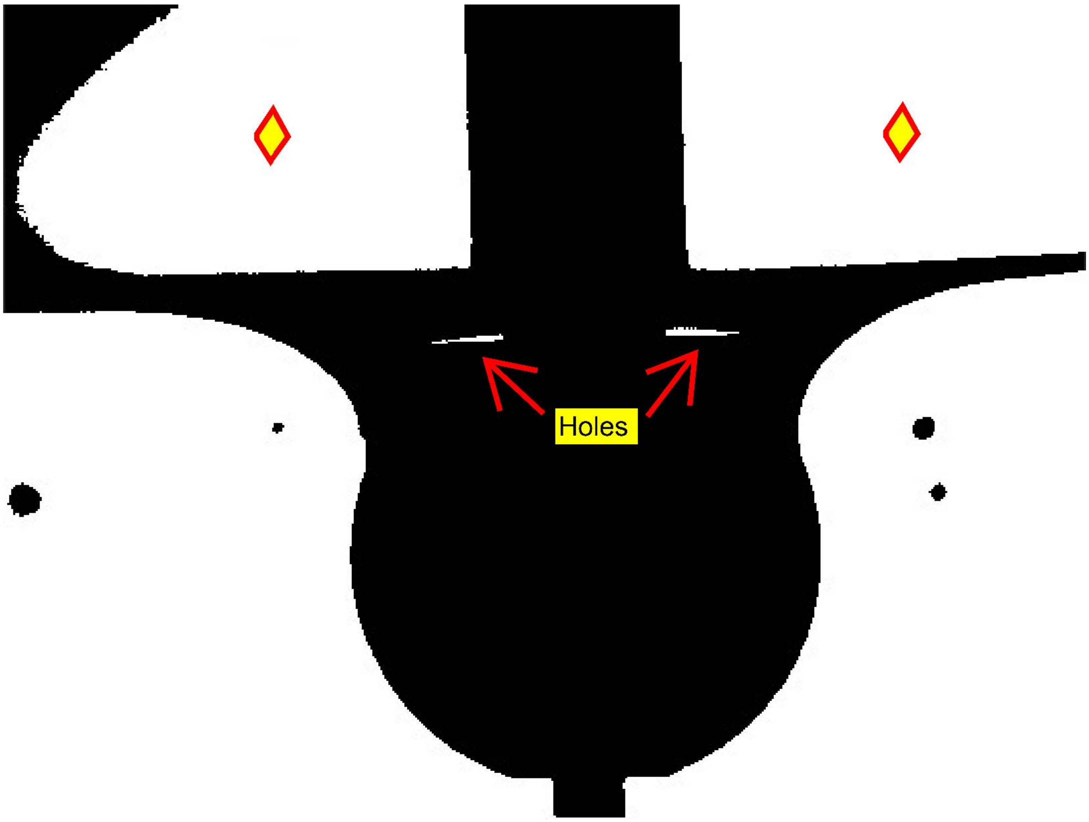

Figure 7.6: The black and white image showing basic thresholding. The diamonds show the points to be used for flood filling. Two holes, which are detected and filled by MATLAB's **imfill** function are also shown.

trieve information on a range of properties of each component. In this instance, the area (number of pixels) is required. The components of the image mask in FIGURE 7.6 are shown in FIGURE 7.7 after filling the relevant sections to produce a single main foreground object. Each foreground component is shown with a different colour.

7.4.3.5 Separating regions

It is a common feature of the images in question that the combined sphere+holm is the largest foreground component in the image. (By default, the background is assigned the label '0' and is considered an absence of, rather than a discrete, object, and may not be continuous.) In a typical image, bubbles or other particles will have significantly smaller areas than the main object. In this way, particulates and small, unattached bubbles can be identified and discarded.[156] This is highlighted

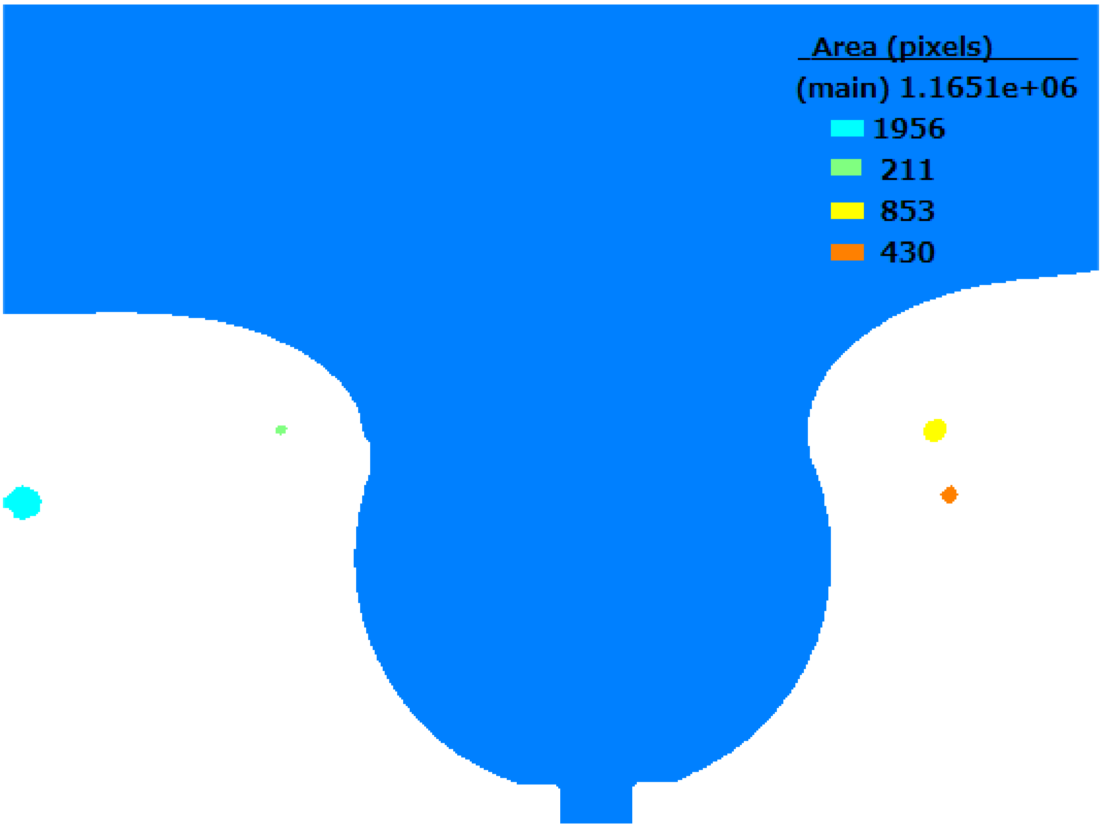

Figure 7.7: The BW mask after closing and filling FIGURE 7.6. The separate components, as found by MATLAB's algorithm, are each coloured differently. The areas of each component are shown.

by FIGURE 7.7. As uneven light conditions may result in large areas of dark background that will appear to the BW image as foreground objects, careful definition of the regions of interest must be used to minimise this issue. The final BW mask associated with FIGURE 7.6 will contain only the largest foreground object – shown in FIGURE 7.7 as the dark blue main drop.

7.4.3.6 Perimeter mask

The final phase of the "WIM" mask comprises solely the single largest foreground object (the sphere and holm interface) and ideally extends from the left- to right-most extremes and from the interface to the upper limit of the image. This should effectively divide the mask into two sections: the lighter fluid and sphere together as the foreground objects, and the background beneath them.

This matrix undergoes two more morphological operations. First, the perimeter between the fore- and background is found. Secondly, the one-pixel wide perimeter is enlarged by a structuring element of user-defined width *wd*. (A different size can be specified to that used for the earlier close operation.) Typically, a disk-shaped element of a few pixels is appropriate, forming a thin line of *2*wd* pixels' width straddling the perimeter of the BW image. The resulting array is a positive mask that will be used to identify a region from which edge pixels should be taken. Any points outside of this region are deleted.

Figure 7.8: (a) the "WIM" mask and (b) corresponding perimeter mask. The perimeter was enlarged using a disk-shaped structuring element with a radius of 5 pixels.

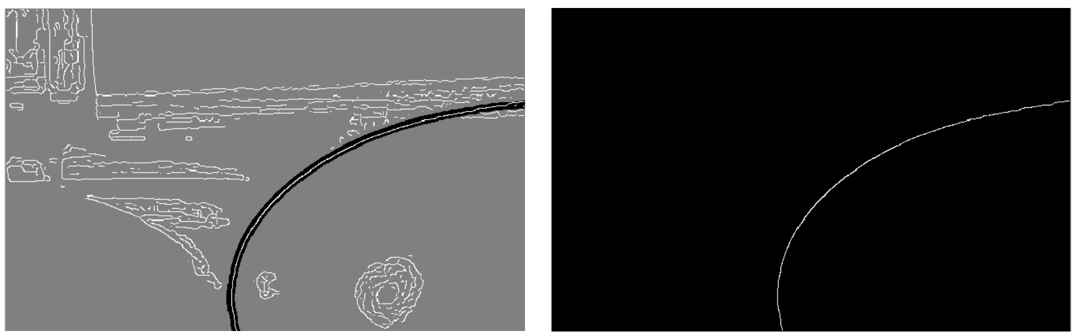

Figure 7.9: (a) A closeup overlaying the perimeter mask over the detected edges on the right-hand holm. The edge matrix is shown in white and grey. Any edges outside of the black region (the perimeter mask) are discarded, leaving the clean edge shown in (b).

7.4.4 White mask

The *white mask* identifies isolated bright regions in the image that are not connected to the true background. The true background is defined as the largest component in the inverted thresholded image. These regions are then simply dilated by a set fraction to cover any edges within that region. The purpose of the *white mask* is to create a masking region over bright reflections which may produce erroneous edges. While many of these regions will be filled in automatically by the **imclose** and **imfill** functions, *white mask* is a simple backup to cover for when the reflections sit on the edge itself, within the region defined by the *perimeter mask*. An example is given in FIGURE 7.11(c)(pg.101).

7.4.5 Bubble mask

While detached bubbles are excluded by removing all but the single largest component in the image, bubbles or other particulates attached to the interface, so-called "adhering noise", will be regarded by MATLAB as part of the main component. This was of particular concern when analysing images during and after microwave irradiation, where rapid temperature increases often resulted in bubble formation. While the size, position and pixel intensity of this attached noise varies greatly, the trait common to almost all instances is their roughly circular shape.

MATLAB is capable of identifying circles and parts of circles in an image (MATLAB function **imfindcircles**). The circular components identified are enlarged by a specified percentage to remove corners that may have been smoothed over by the structuring element during the **close** operation. The resulting array is a negative mask that will be used to remove edges from the identified areas. These bubbles are far smaller than the Teflon sphere around which the holm is formed. As it is possible to search for circles of radii within a particular range, there is no concern that the sphere itself will be mistakenly identified as noise. By setting {`'ObjectPolarity'`, `'dark'`}, bright circular reflections inside of a bubble are ignored.

An image with an unusually large number of bubbles is shown in FIGURE 7.10. On the greyscale image, circular objects detected by **imfindcircles** (with radii enlarged by 50%) are highlighted. Of particular concern are the four adhering bubbles, whose edges are not part of the true interface but will be considered part of the main foreground object in *perimeter mask*. In part (b) of the figure, the *bubble mask* is shown overlaying the *perimeter mask*. Where the two masks overlay, that portion of the perimeter mask is deleted. Consequently, only edges within the fully black region are returned.

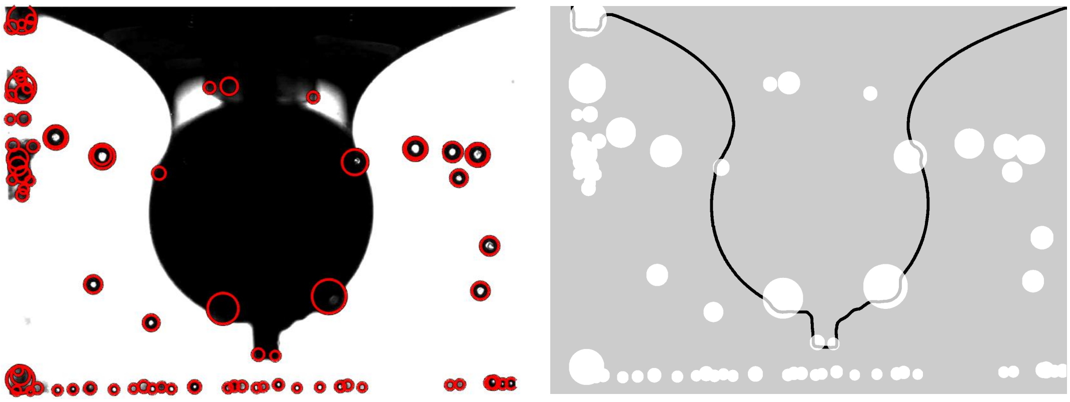

Figure 7.10: (a) The original greyscale image showing circular objects detected by MATLAB's **imfindcircles** (radii extended by 50%). (b) The perimeter mask (black) with the bubble mask (white) overlayed. Only edges lying within the fully black region will be kept.

In order to minimise computing time, the *bubble mask* is computed only on the four search regions, rather than on the image in its entirety. In this way, the host of small bubbles around the bottom of the image can be avoided. Note that a fairly low thresholding factor is required to identify adhering bubbles where only small parts of the circle remain.

7.5 Applying the masks

The masks are applied to the results of the Canny edge detection as follows:

- *Perimeter mask* (positive): {`Edges(Mask==0)=0`} removes any edges outside of the mask region.
- *White mask* (negative): {`Edges(White==1)=0`} removes any edges within the mask region.
- *Bubble mask* (negative): {`Edges(BubMask==1)=0`} removes any edges within the mask region.

7.6 Final edge filtering

After the final edge matrix is created (FIGURE 7.10(b)), component labelling is used one final time to determine the number of pixels in each edge segment. True edges are most likely to long, fairly continuous lines. Accordingly, any segment comprising of less than a certain minimum number of pixels, as specified by the user, is discarded. While this function will have no apparent effect on smooth, high contrast images, which tend to produce strong and continuous edges, it can be used to manage noise (such as seen around the sphere in FIGURE 7.3) that may be included if a large structuring element is used to create the *perimeter mask*.

7.7 Alternative algorithms

While the method described above is recommended, in some instances other algorithms can be an appropriate choice to reduce computational time. *ThreshGUI* contains options to switch the edge detection algorithm to choose the longest continuous line in each region (by setting {`Use BW mask==false`}) or to search for the two longest lines, one from each end of the search area (by setting {`search from both ends == true`}). It is also possible to switch on or off *bubble mask* and *white mask* individually if the user so desires. As an example, the user may wish to turn off *bubble mask* for images with grainy backgrounds, as **imfindcircles** will be very slow.

7.8 Improving edge accuracy

7.8.1 General remarks

Image analysis is arguably the most crucial aspect of successful analysis. Prior to any efforts from the computational side, it is imperative that high-quality images be taken and cropped appropriately. The ideal image will have a near-uniform light background with the sphere and top phase showing as a dark foreground image. The image should be cropped so that the dark foreground object stretches from the left to right side.

7.8.2 Subpixel resolution

It is well accepted that a certain error will be associated with the location of an edge in an image. Indeed, even in a black and white image, an error of at least ±1 pixel is implied simply by choosing whether the light or dark pixels represent the true edge.[42,84] This issue is particularly true of images with poor focus. Sub-pixel resolution was introduced to ADSA to reduce this inherent error.[42] By fitting a natural cubic spline along the principal direction (vertical, horizontal or diagonal) that sat closest to perpendicular to the detected edge. The 'true edge' was then identified as the point where the pixel intensity reaches the mid point between the high and low plateau of the spline. The inclusion of sub-pixel resolution has been maintained in subsequent improvements to ADSA in its various forms, however it was found to have minimal effects on images with moderate contrast.[72] In the present method, we conclude that the impressive pixel resolution afforded by modern, high-definition cameras improves the accuracy of image analysis sufficiently that sub-pixel resolution is not really required, and it was omitted here in favour of improved computing speeds.

7.8.3 Image angle adjustment

Maintaining the camera's perfect horizontal alignment can be quite difficult, even with the use of leveling devices. Drop fitting programs typically have methods to adjust for drop/interface tilt. For example, many versions of ADSA used plumb-lines to provide a vertical reference to adjust the image,[35,83] and some later ADSA variants[72,82] include an additional optimisation parameter, α, thus including the adjustment angle within the optimisation module for each image or frame.

Rather than determining the angle as an optimisation parameter and recalculating for each image, the angle is calculated from certain geometric considerations that are described in the following chapter. Note that rotations applied directly to the image are for display purposes only, as the operation can subtly alter the location of the edge. The coordinates are determined from the original image and mathematically rotated around the center point of the sphere.

7.9 Flowcharts

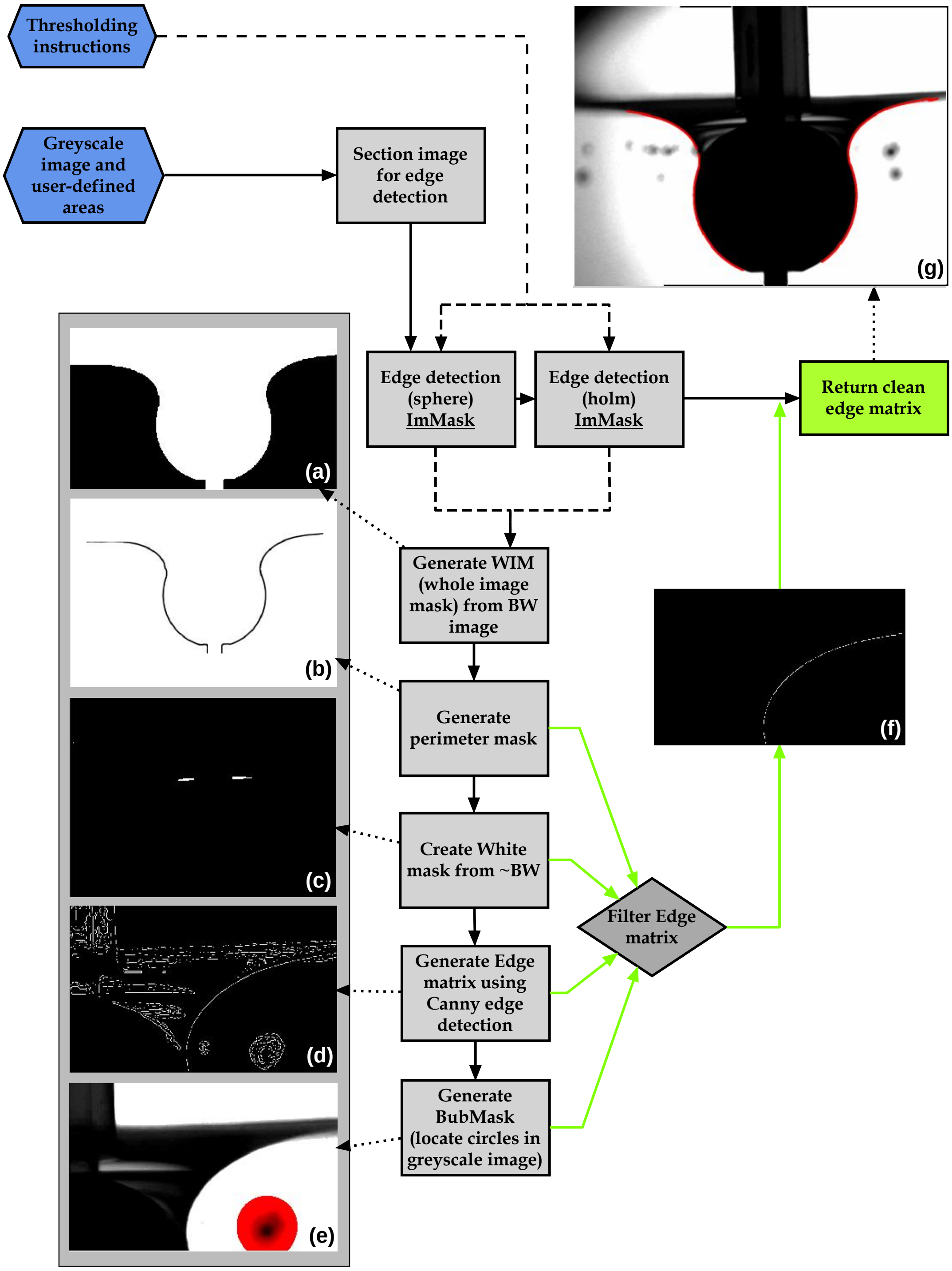

Figure 7.11: An overview of the image analysis algorithm, showing images of the key steps.

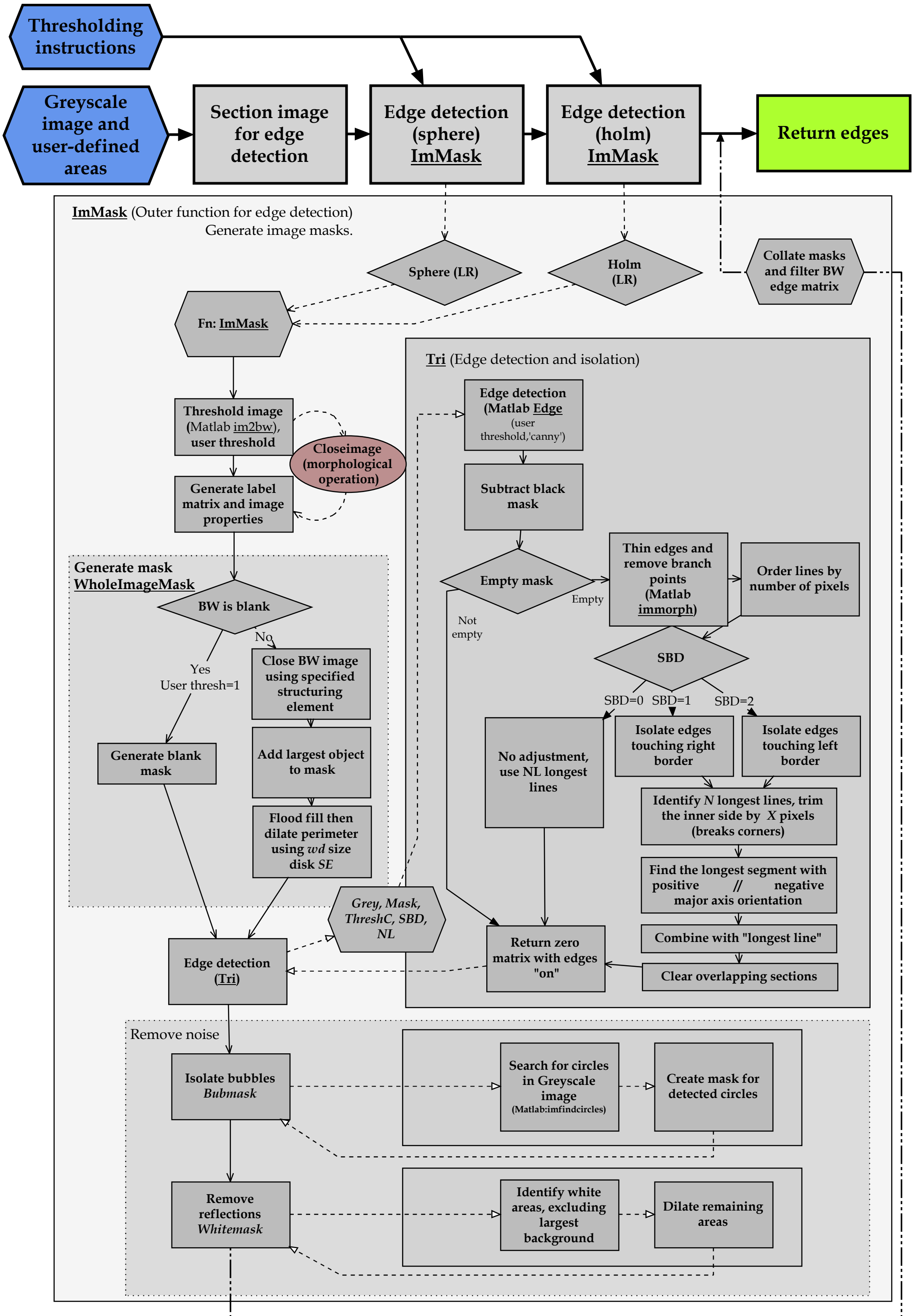

Figure 7.12: Flowchart describing the image analysis algorithm. This flowchart describes the same processes as FIGURE 7.11 on a function level.

7.10 Chapter summary

Determination of the interfacial pixels is a crucial aspect of any optical analysis technique. This chapter details the method of handing edge detection in complex systems, specific to the holm analysis technique. The problems surrounding edge detection are approached by the use of multiple masks which filter off undesired edges from the image. Components with small areas can be reasonably attributed to detached particulates or bubbles, or some other turbidity in the bulk.[156] Component labeling allows these components to be identified and flagged in a black-and-white (thresholded) version of the image, making it an integral part of producing the image mask. The characteristic circular shape of a bubble makes it possible to detect them within the image and consequently discard such adhering noise. A final mask detecting bright components is used to blank out reflections that are not handled by the other masks. While default values are suggested, the user has full control over the edge detection process *via* the interactive *ThreshGUI*. Once the parameters are saved, they are used automatically for all other frames in the series, which allows effective multi-frame analysis. All values are saved in matrix form in the output folder and can be recalled for a new run.

The key aspect of this methodology is the ability to isolate different types of noise in complex images and thus identify the pixels belonging to the true edge while using a high fidelity edge detection algorithm such as Canny edge detection. All image modifications are done on masking layers, leaving the original image untouched for accurate edge detection.

Part III

Application to static and dynamic systems

The remaining chapters of this thesis detail specific experiments in which the holm technique has been applied. Each experiment highlights a particular aspect of the technique that made certain measurements possible – from the ability to measure in non-quiescent solutions, to measuring highly viscous fluids, to performing measurements in small, enclosed spaces or over long periods of time.

Much of the material in this section has been presented in the following publications:

- Hyde, A.; Phan, C.; Ingram, G.; Determining liquid–liquid interfacial tension from a submerged meniscus, Colloids Surfaces A Physiochem. Eng. Asp. 459 (2014) 267–273.
- Hyde, A.; Horiguchi, M.; Minamishima, N.; Asakuma, Y.; Phan, C.; Effects of microwave irradiation on the decane–water interface in the presence of Triton X–100. Colloids Surfaces A Physiochem. Eng. Asp. 524 (2017) 178–184.
- Hyde, A.; Phan, C.; Yusa, S.; Dynamic interfacial tension of nonanoic acid/ hexadecane/water system in response to pH adjustment. Colloids Surfaces A Physiochem. Eng. Asp. (2018) *[In press – Accepted manuscript]*..

Figures from the above publications have been noted where applicable.

CHAPTER 8

Static measurements of oil-water interfaces

8.1 Overview

The measurement method developed in this thesis was originally designed and tested with static systems. Common water-alkane systems (hexane, dodecane and hexadecane) were chosen as they can be easily measured by both the pendant drop method (ADSA[72,129]) and the new holm method described in this thesis. The measurement error associated with the two methods was comparable and the values in good agreement. Additional benefits of the using the holm interface were shown by measuring the interfacial tension between water and high-density silicone oil (polydimethylsiloxane), a highly viscous oil with a density only 3% less than water, physically an incredibly difficult measurement using the pendant drop method. The data presented in this chapter was published in Hyde et al.[78].

8.2 Interfacial tension measurements

8.2.1 A note regarding fluid properties

As the purpose of this experiment was to compare measurements between the holm method and the regular pendant drop method, and not to measure the interfacial tension of the pure systems, the chemicals were used as received and no attempt was made to purify them further.

All measurements were done in a climate controlled laboratory at 18 °C. Fluid densities were calculated from literature values. The fluids and their densities are shown in 8.1 below.

Table 8.1: Fluid densities

Fluid	Density (kg/m^3)
Hexane	660.6 (at 25 °C) [66]
Dodecane	749.5 (at 20 °C) [67]
Hexadecane	770.1 (at 25 °C) [67]
Silicone oil	973.7 (at 18 °C) *(Meas)*
Water	998.6 (at 18 °C) [131]

8.2.2 Holm measurments

Water-oil interfaces were prepared between deionised water and various alkanes (hexane, dodecane, hexadecane) in small, transparent plastic cells with sides 3 cm and 5 cm in length. Likewise, the silicone oil-water interface was formed between deionised water and 1000 cp silicone oil (ρ = 973.7 kg/m^3), resulting in a density difference of only 24.7 kg/m^3. The interfaces were deformed using small Teflon spheres hung on a Teflon thread. Spheres of four different sizes were used, from 6.35 to 12.70 mm in diameter with a tolerance of 0.1 mm, resulting in different capillary numbers. The interface was photographed, resulting in roughly 700 – 1000 coordinate points on each side of the detected edge. The Teflon spheres were of accurately known size and provided the link to scale the photographs to the size of the real system.

8.2.3 Pendant drop measurements

The same oils were used to prepare pendant drops of each alkane in a bulk water phase, using a stainless steel needle with an outer diameter of 1.2 mm. Images of this system were analysed using both commercial[156] and in-house software. Examples of the images used for fitting are shown in FIGURE 8.1.

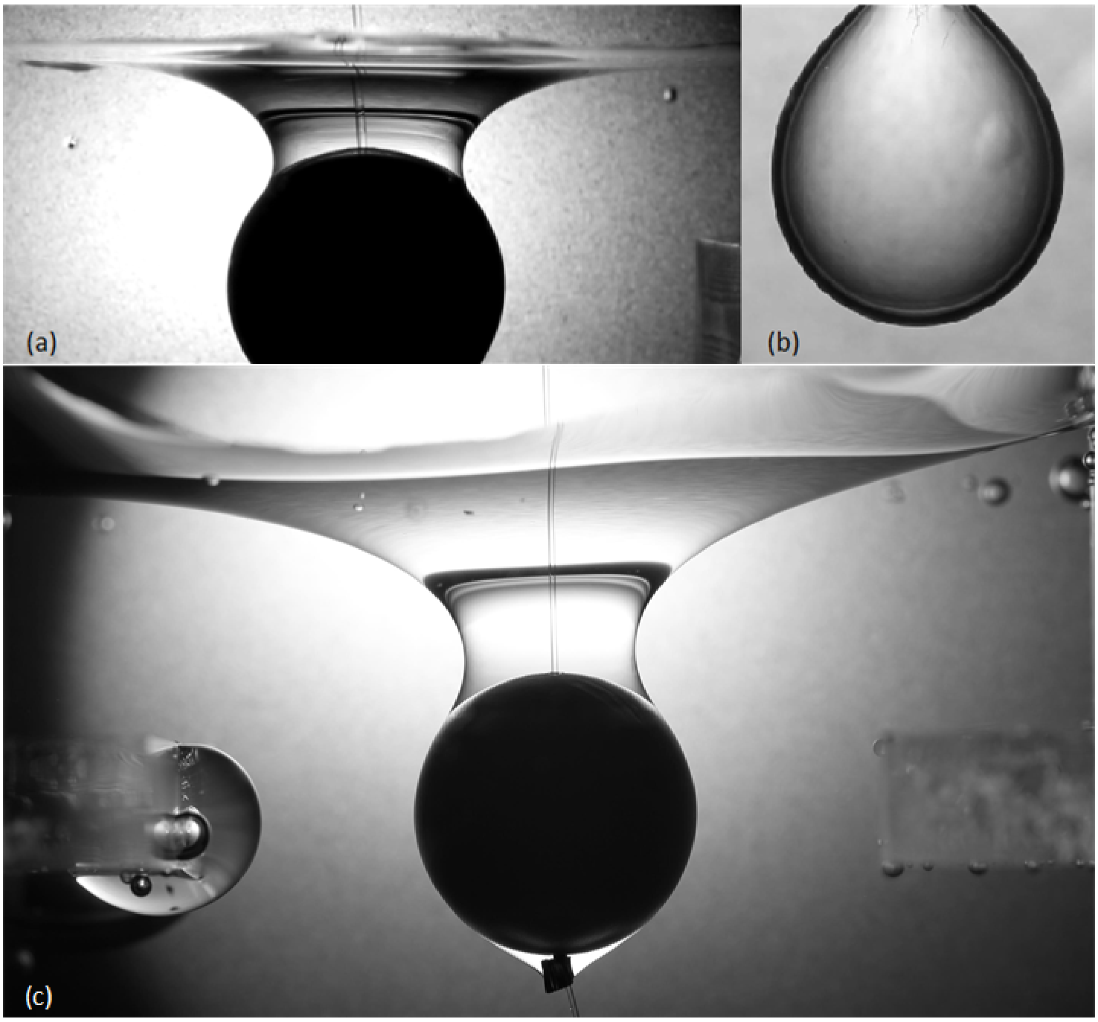

Figure 8.1: Sample images used for the analysis: (a) holm meridian between hexadecane and water, around a 9.53 mm sphere, (b) pendant drop from a 1.2 mm capillary, (c) holm meridian formed between silicone oil and water.

8.2.4 Comparison of ADSA and the holm method

The interfacial tension of the alkane-water system was measured successfully using the holm method, to an accuracy comparable to the established ADSA technique. The results are compared in TABLE 8.2.

Table 8.2: Comparison of the results obtained using the new method and commercial pendant drop software for four water-oil systems. The average and 95% confidence interval is calculated over at least 5 images. This data was originally presented in Hyde et al.[78]

System	$\Delta\rho$	Method	Interfacial tension (Average)	Confidence interval (95%)
	kg/m^3		mN/m	mN/m
Hexadecane-water	224.8	Holm meridian	31.5	±0.4
		Pendant drop	31.0	±0.3
Dodecane-water	249.4	Holm meridian	45.6	±1.2
		Pendant drop	45.7	±0.7
Hexane-water	342.0	Holm meridian	38.2	±1.3
		Pendant drop[156]	40.7	±2.0
Silicone oil-water	24.7	Holm meridian	37.7	±1.2

8.2.5 A note on measuring the air-water interfacial (surface) tension

Fitting of air-water interfaces was found to be difficult and error-prone, for two key reasons. Firstly, the sharp contact angle reduced the available curvature of the holm and made picking the starting angle more difficult. Secondly, the images were prone to reflections and blurring that made determining the interfacial coordinates difficult. In FIGURE 8.2, the black-white thresholding of the image illustrates how the blurred regions (which are substantially lightened) are lost during thresholding.

Gradient edge detection algorithms, such as the Canny method, are unable to identify such gradual changes in intensity as a true edge. As a wide variety of techniques exist that can provide extremely accurate measurements of surface tension, no further work was done on these systems.

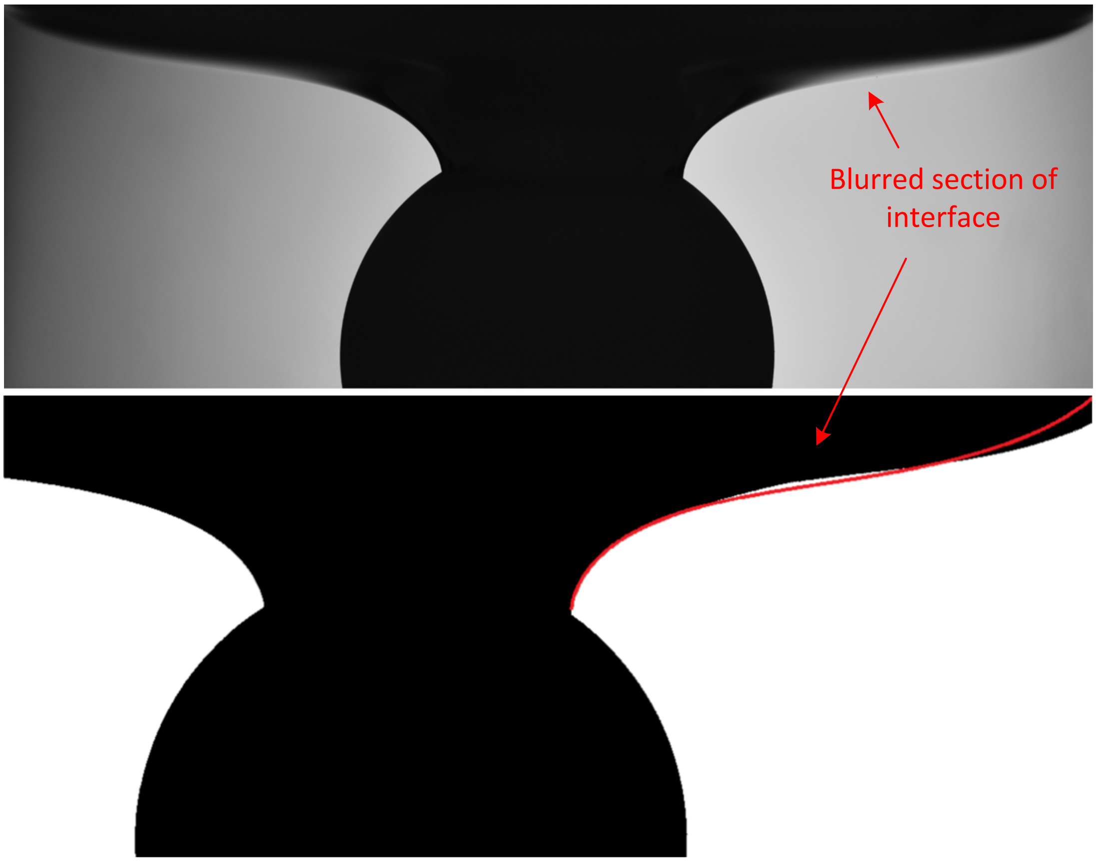

Figure 8.2: A sample image of the air-water interface: (a) holm meridian between water and air, showing a region of poor contrast, and (b) the fitting based on image(a).

8.3 Shape fitting in a finite (bounded) fluid

8.3.1 Non-axisymmetric containers

Unlike a pendant drop, the holm meridian used in this technique is affected by both the submerged sphere and the walls of the cell itself. Clearly, a square-based cell cannot be axisymmetric. However, the optical distortion produced by a curved surface precludes the use of a cylindrical cell. Even with a cylindrical cell, the bulk fluid will form a second holm as it approaches the wall. Consequently, there is a real question regarding whether wall effects have any bearing on the accuracy of the technique.

8.3.2 Wall effects

The capillary number (8.1) describes the distance for which a bulk interface will be deformed by an object. In other words, in an unbounded fluid of height h, the interface will be deformed for a distance λ_c from a submerged object. After this point, the bulk interface will have returned to the height of the horizontal, undisturbed interface. One can conclude, therefore, that a distance of $2\lambda_c$ between the edge of the ball and the cell wall would be sufficient to eliminate effects from the wall on the profile for fitting.

$$\lambda_c = \sqrt{\frac{\gamma}{\Delta\rho g}} \tag{8.1}$$

8.3.3 The effect of sphere size on the measured value

Measurements were taken using Teflon spheres ranging from 7.14 to 12.7 mm in diameter in order to test the effect of the sphere size, hence the distance between the sphere and the wall, on the measured value. Unlike the Bond number used during fitting, the capillary length is independent of the system geometry. A distance of at least twice the capillary number between the sphere and the wall is required to produce a section of horizontal interface between the two deformed regions, insuring that the holms produced by the wall and the sphere do not interact. As can be seen in TABLE 8.3, even the larger spheres amply meet this criteria for the alkane-water system, with sufficient space between the wall and the sphere to produce a flat "unbounded" interface. A consistent interfacial tension was obtained for the alkane-water system, with successful fitting of the four sphere sizes. These results are shown in FIGURE 8.4. In contrast, the very small density difference between silicone oil and water results in a significantly larger capillary length of 11.3 mm. As can be seen in FIGURE 8.1(c), there is not sufficient distance between the wall and the sphere to prevent interaction between the two holms.

Clearly, if the distance between the ball and the cell wall is less than $2\lambda_c$, wall effects will act on the part of the holm being used for fitting. This is illustrated by FIGURE 8.1(c), where the rising tail of the holm as it approaches the wall is clearly visible. In fact, this "rising tail" was used by Princen[116] as part of an early fitting method, as discussed in SECTION 4.2.2 (pg. 41). In this work, we show that the Young-Laplace equation can produce a fit to this rising version of the holm. In fact, this holm is arguably more unique for fitting than a short holm of near-spherical cross-section that quickly reaches its horizontal asymptote, or a small portion of the holm from a system with a large capillary length, as a significant length of the

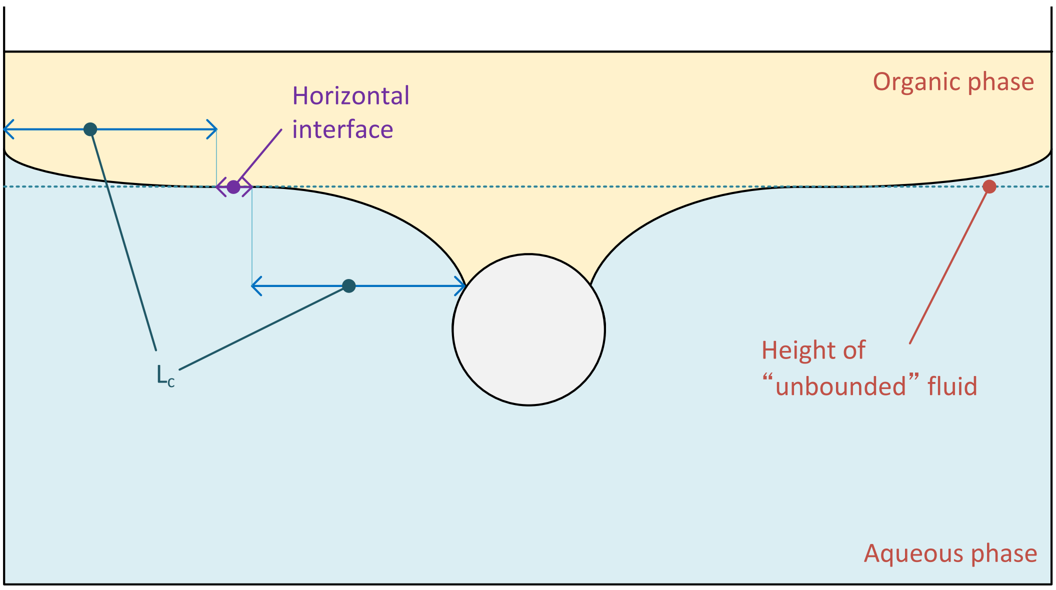

Figure 8.3: Schematic representation of distances in the cell. The blue lines, equal in length to the capillary length, correspond to the distance required from a submerged object for the bulk height to be retained. If the available distance between the sphere and the cell wall is more than twice the capillary length then there will be a portion of the flat interface present (purple) and the sphere can be considered to be in an infinite bulk meniscus.

Table 8.3: The effect of sphere size on system geometry for the measurement of the hexadecane-water (λ_c=3.76 mm) and silicone oil-water (λ_c=11.3 mm) interfacial tensions in a rectangular cell with 5 cm sides. The holms formed around the sphere and around the wall will interact if the distance between the sphere and the wall is less than $2\lambda_c$.

Sphere size (mm)	Distance to the wall (mm)	Distance to the wall / λ_c
Hexadecane-water	*(λ_c = 3.67 mm)*	
7.14	21.9	5.8
9.53	20.7	5.5
11.11	19.9	5.3
12.7	19.15	5.1
Silicone oil-water	*(λ_c = 11.3 mm)*	
12.7	19.15	1.7

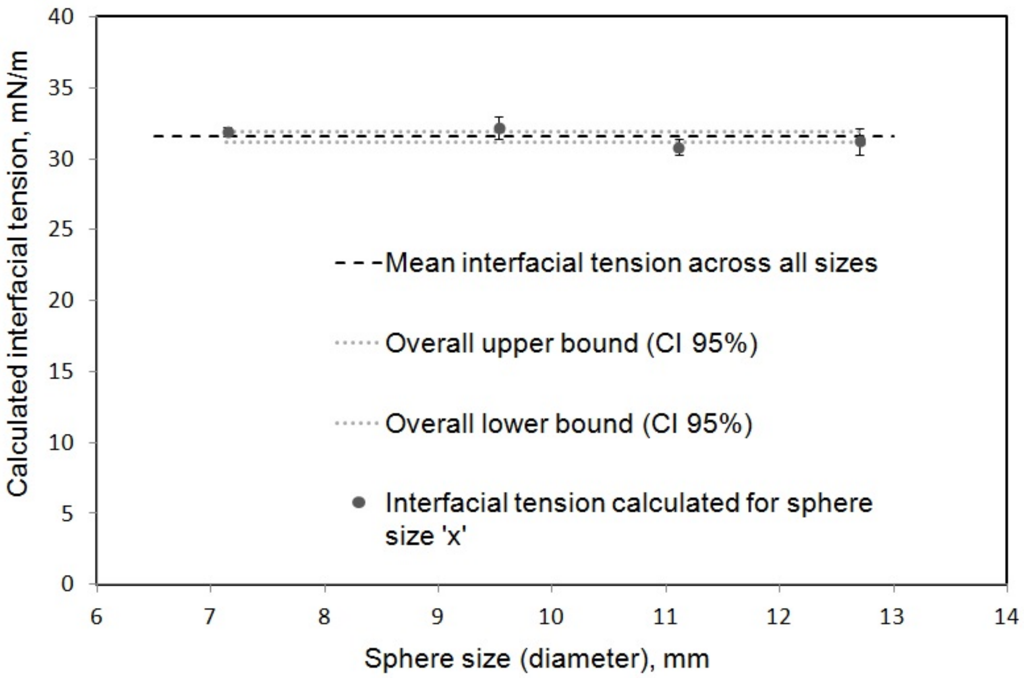

Figure 8.4: The hexadecane-water interfacial tension calculated from four distinct sphere sizes. The size of the sphere was not found to affect the measured interfacial tension. *Originally published in Hyde et al.*[78]

curved holm is available for fitting. Nonetheless, sufficient distance is required to allow enough curvature for adequate fitting, and similar issues can arise if the holm is short enough to be nearly straight. Furthermore, it is important to ensure that optical distortion from the edges of the cell do not impact on the image quality when fitting close to the wall.

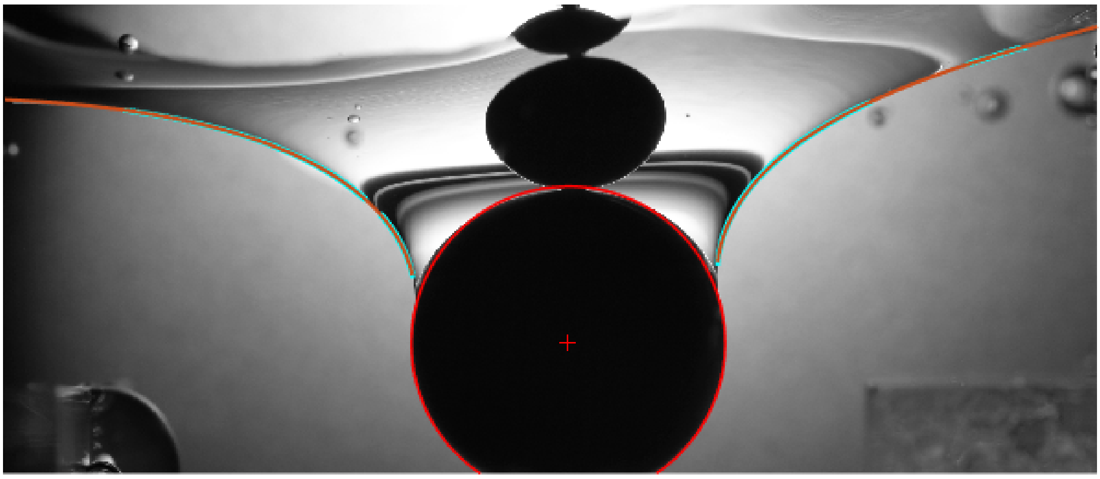

Figure 8.5: Successful fitting of the silicon-water interface with the sphere placed less than $2 \times L_c$ from the wall. The fitted Young-Laplace curve follows the holm as it rises towards the cell wall. The fitting (orange) is shown overlaying the detected coordinates (cyan).

8.4 Measurement of interfacial tension in systems with low Bond numbers

8.4.1 Failure of pendant-drop fitting

All shape-based methods require unique interfaces to solve accurately. This "uniqueness" is strongly correlated to interfacial deformation. In general, there are two extremes where this unique deformation is lost and fitting with the pendant drop method becomes error-prone, or even impossible.

Consider a soap bubble. Because the density of the interior and exterior phases are the same (both are air), there is no deformation due to gravity and the drop is spherical. A similar phenomenon is observed in liquid drops when the density difference between the two fluid is small. It is widely accepted in the literature that shape fitting using both pendant and sessile drops starts to incorporate significant errors as the drops approach a perfect sphere. In such a scenario, large changes to the interfacial tension produce very small changes in the shape of the drop, increasing the fitting error substantially or even making a drop impossible to fit. [102,129]

The second scenario occurs when the drop is too small to have sufficient weight to deform under gravity. In other words, the inwards-acting tension force maintaining the drop's spherical shape is not overcome by the drop's weight. This results in pendant and sessile drops approaching spherical caps. Again, shape fitting techniques are unable to accurately analyse near-spherical images as the solution is non-unique.

8.4.2 Criteria for accurate fitting (pendant and sessile drops)

These drops are often reported as having a "low Bond number", where the Bond number, B_o is given as:

$$B_o = \frac{\Delta\rho g R^2}{\gamma} \tag{8.2}$$

Where γ is the surface or interfacial tension (mN m^{-1}), g is the acceleration due to gravity (ms^{-1}) and $\Delta\rho$ is the density difference between the two phases (kg m^{-3}). R is the characteristic length of the system, in meters. The Bond number, essentially expressing the ratio between gravitational and surface forces (and hence the tendency of the drop to deform) has been widely used as a criteria to express the suitability of a particular drop for measurement. Saad and Neumann[129] found that drops with Bond numbers less than one, or in excess of 5, tended to result in poor fitting.

While it is not possible to fit these systems using a pendant or sessile drop without incurring large errors, this issue is not observed with the holm method, provided that a sufficient length of the interface has been captured for fitting, as it is still possible to get good deformation of the interface by altering the height of the sphere. Figure 8.6 shows the effect of changing oil density on the deformation of the interface, all other parameters being the same. Indeed, we can hypothesize that the lower density difference may actually be easier to measure using this technique, as the strong deformation around the 'neck' of the holm should provide the uniqueness required for accurate fitting.

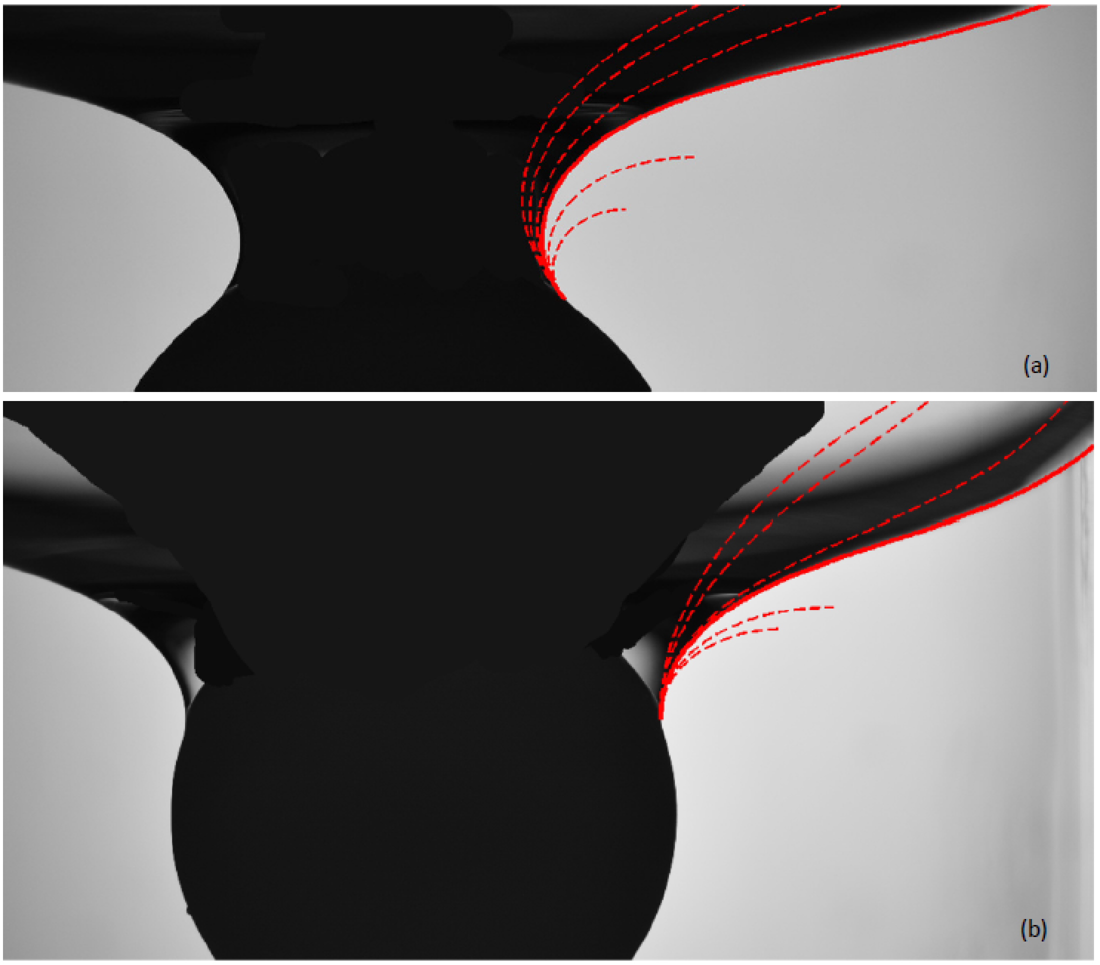

Figure 8.6: The theoretical effect of changing the density difference between fluids, with all other parameters kept constant. The effect can be repeated with different system geometries (a) and (b). The oil densities used are: 990, 900, 700, 655 (actual oil density), 500 and 400 kg/m3. The Bond number decreases as the density difference is reduced. Parameters: x0, y0 as shown, $R_{sphere} = 5.57$ mm, $\gamma = 41.9$ mN/m, $\rho_{water} = 997$ kg/m^3). In (b), reflections have been blacked-out for clarity. The effect of penetration depth on the interfacial curvature is clearly shown.

8.4.3 Alternative measurement methods for low Bond number systems

There is a considerable dearth of available data for the measurement of the interfacial tension of silicone-oil systems. Its high viscosity tends to preclude pumping to form drops, and it is rather too adhesive to allow for convenient measurement using plate or ring methods. Peters and Arabali[112] recently published data for the temperature dependence of the silicone oil-water interface measured using a new contour method. While this method has provided a significant data set for the system, it does require a significant investment in machinery and highly sensitive force measurements.

8.5 Alleviating complications with fluid handling

8.5.1 The measurement of viscous and/or opaque samples

The measurement of dense oils presents other challenges to existing methods. Typically, the dense oils are highly viscous, and by consequence, extremely difficult to pump through a syringe to actually produce a drop. However, as they are typically not fully transparent, it is generally not possible to have them as the bulk fluid and pump the less viscous fluid, such as water, without obscuring the interface.

In the present work, it was not possible to pump the silicone oil sample due to its high viscosity. Further, the oil was too opaque to obtain clear photographs with the oil as the bulk phase. These complications make it nigh-on impossible to measure the system using the pendant drop method. In comparison, pumping difficulties were eliminated by the holm method developed in this thesis. As there is no requirement to produce a drop, pumping is no longer required. Additionally, blurring and other optical issues are avoided as the both fluids are in contact with the cell walls. Opaque fluids can be handled successfully by choosing between the raised and submerged holm (facilitated by altering the chemical affinities of the deforming sphere). This will be discussed in further detail in CHAPTER 11.

8.6 Alleviating stability issues

Drop stability is a recurring issue for measuring liquid-liquid interfacial tensions using the pendant drop method. Formation of a drop of sufficient size to produce good deformation can be challenging, particularly if the interfacial tension is high. Automation of the pumping system is possible, but can be a significant investment and constrains the space around the cell. If a drop detaches from the needle, producing a new drop will result in a fresh interface, a potential problem in cases of long equilibrium times or for measuring dynamic effects. In the holm method, the stability of the interface is greatly increased by forming a bulk interface around a fixed solid object. The extent of curvature of the holm is modified simply by adjusting the height of the sphere.

8.7 Chapter summary

In this chapter, the application of the holm technique for the measurement of liquid-liquid interfacial tension of simple water-oil systems is shown and found to be in good agreement with the well-known pendant drop method. As illustrated by the measurement of high-density silicone oil, the technique is found to provide a far simpler experimental component than the pendant drop method for liquid-liquid samples. The measurement of highly viscous samples was facilitated by avoiding pumping requirements, and good deformation of the holm made the measurement of low Bond number systems possible. The method is shown to ameliorate three major issues encountered when trying to measure liquid-liquid interfacial tensions with viscous fluids: pumping difficulties, stability issues, and fitting error due to near-spherical drops.

Chapter 9

Measuring the effect of microwave irradiaton on interfacial tension

9.1 Overview

Microwaves are seeing increasing use in both industrial and domestic situations as a means of rapid heating of certain responsive materials. From an industrial viewpoint, microwaves offer efficient, selective heating with no need for contacting phases. To the domestic user, microwaves have revolutionised our storage and heating of foods. However, while the physical process of dielectric heating is fairly well understood, there remains little information on its effects on substances themselves. This is of interest not only in terms of unwanted side effects in industrial processes, but also regarding potential health effects in food and domestic use. On such a stage, further insights into the effects of microwaves on particular substances remain pertinent.

The sensitivity of surface and interfacial tension to the presence of surface active agents in a solution makes them a strong indicator of minute changes in the concentrations of organic and other surface-active molecules. Nonetheless, measuring inside an operating microwave reactor incurs several challenges:

- The apparatus must be small and able to be fully enclosed inside the protective shielding of the reactor.

- It must be possible to maintain a single interface throughout the length of the experiment, vibrations from the reactor notwithstanding.

The *in situ* measurement of the surface tension of water has been measured using the pendant drop technique with small, millimeter-sized droplets.[109] Heating using microwaves was found to significantly reduce the surface tension of water, a phenomena which persisted for several minutes after the microwave had been switched off. Indeed, the surface tension was found to recover significantly more slowly than did the temperature.

This chapter details the use of the holm method for *in situ* measurement of the interfacial tension of the alkane-water system during microwave irradiation. This chapter is partly based on the publication Hyde et al.,[79] which features a comparison of microwave and conventional heating methods on the decane-Triton X-100 (0.66 mM) interface.

9.2 The mechanism of microwave heating

Microwave induced heating, otherwise known as dielectric heating, is the result of molecular rotations disrupting intermolecular bonds. The rapidly switching magnetic field produced by a microwave exerts torque on molecules with a dipole moment, causing them to rotate (FIGURE 9.1). Energy released by the disruption of the bonding network is realised as increases in internal and kinetic energy of the substance, and transferred via molecular collisions to provide volumetric heating.[146]

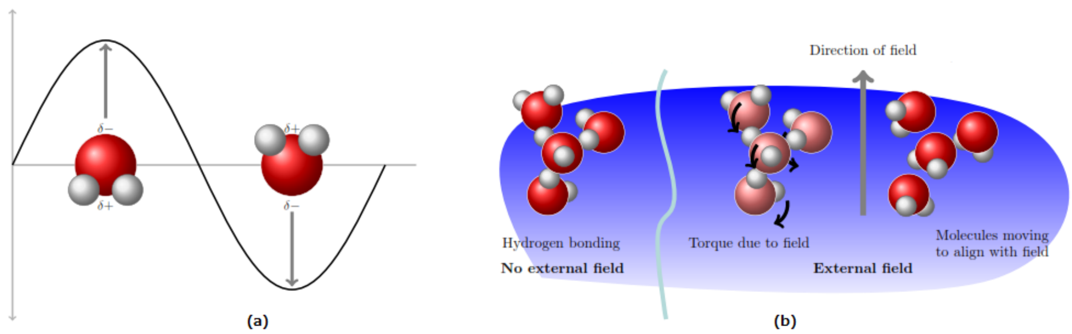

Figure 9.1: The effect of microwaves on intermolecular bonds. (a) Alignment of water molecules to the magnetic field. (b) Disruption of intermolecular bonds.

The extent to which a material will interact with microwaves is described by two complex parameters, the dielectric permittivity and the magnetic susceptibility,[144] and the microwave penetration depth depends on the emission frequency. Polar compounds, such as water, react strongly to microwaves and will tend to align their polar bonds with the external field. These compounds are characterised by a high dielectric permittivity and rapid heating. EQUATION 9.1 quantifies the power dissipated within a given volume of material, known as the 'power density' (P_D, W/m^3) as a function of the dielectric loss factor (where ϵ_r'' is the imaginary component of the relative dielectric permittivity, and ϵ_0 is the permittivity of free space, 8.85×10^{-12} F/m), the microwave frequency (f, Hz) and the strength of the electric field (E, V/m).[16]

$$P_D = 2\pi f \epsilon_0 \epsilon_r'' |E|^2 \tag{9.1}$$

Microwaves have little to no effect on non-polar molecules: compared to water's dielectric constant of 76.7,[130] the dielectric constant of decane is only 2.1.[51] Because of this difference in responsiveness, microwaves offer selective heating of the aqueous phase over the non-polar alkanes.

9.3 Interfacial tension measurements

9.3.1 Using the holm meridian for microwave measurements

When measured using pendant drops in air, microwaves were found to significantly reduce the surface tension of water, a phenomena which persisted for several minutes after the microwave had been switched off.[109] Indeed, the surface tension was found to recover significantly more slowly than did the temperature. Similar long-term effects were seen with the oil-water interface in the presence of some surfactants.

Use of the holm method for *in situ* measurement during microwave irradiation highlights three key objectives:

- The interface remains stable during external vibration and internal movement due to convection currents,

- A single interface can be held stable over a period of hours or days, and
- The system can be completely enclosed.

9.3.2 Microwave experiments

Interfacial tension measurements were taken using the holm method. Bulk interfaces between an aqueous solution (deionised water with and without surfactants) and decane were deformed by Teflon spheres of various sizes. The solutions were contained within a glass cell (27 mm^3). Approximately 6 mL of the aqueous and 2 mL of the organic phases were used. The Teflon spheres were threaded onto thin Teflon tubes, through which was passed a fibre-optic cable for temperature measurements. The complete cell was contained within the shielding of the microwave reactor, as shown in FIGURE 9.2(a). The optical fibre was threaded through the shielding to the thermometer (AMOTH FL-2000), measuring the temperature of the aqueous phase in contact with the sphere. As shown in the figure, the custom-built shielding included viewpoints that could be use to film the cell and interface or provide lighting for the images. Densities were calculated from empirical correlations based on the solution temperature.

The microwave reactor (IMG-2502 microwave generator) was used to irradiate the cell with 60 W for 60 s intervals. Chemicals were used as received.

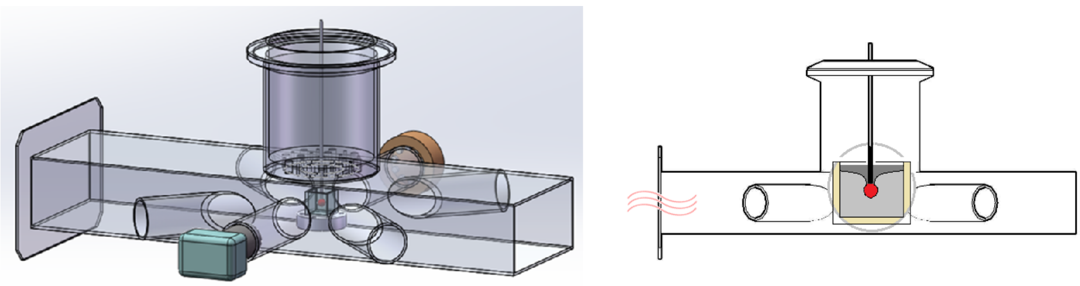

Figure 9.2: A schematic of the setup used for the measurement of interfacial tension inside of the microwave reactor. *Originally published in Hyde et al.* [79]

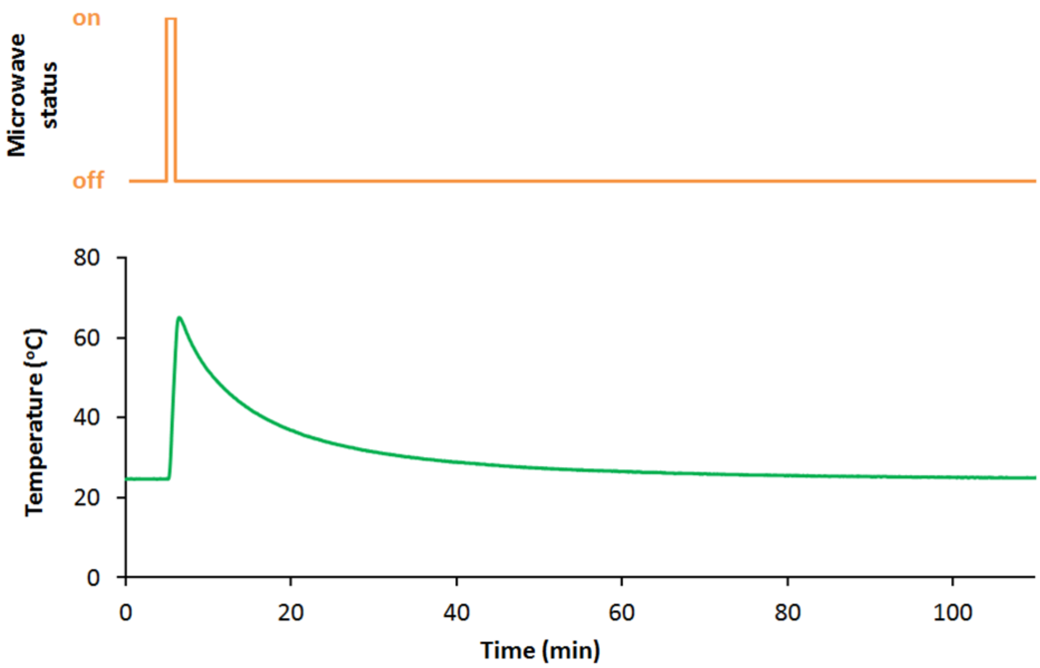

Figure 9.3: An example of the temperature profile of the aqueous layer during microwave irradiation. In this figure, the sample (decane-Triton, 0.66 mM) was recorded for five minutes at equilibrium before being subjected to irradiation at 60 W for 60 s. The cell was then allowed to cool naturally inside the reactor. *Originally published in Hyde et al.*[79]

9.3.3 Monitoring the system temperature

The temperature at the base of the sphere (i.e. the temperature of the aqueous solution in the middle of the cell) was monitored at 1 s intervals throughout the experiment. This temperature was used to calculate the densities of the solutions from empirical equations for decane[69] and water.[131] Dilute surfactant solutions were assumed to have the same density as water, and the densities of brine were determined experimentally.

While the aqueous layer is heated directly by the microwave, the non-polar alkane layer is non-responsive to microwave radiation and thus will not undergo dielectric heating. As a result, the organic layer is heated from the aqueous layer, incurring a temperature discrepancy during rapid heating. However, as the alkane layer is very thin (a few millimeters), this heat transfer is assumed to be quite rapid. Furthermore, a temperature discrepancy of 10 ^{o}C between the two layers will effect the calculated density difference by less than 0.1%, and will thus have an insignificant effect on the calculated interfacial tension. Due to its thinness, temperature gradients within the organic layer are expected to be minimal.

In contrast, temperature gradients are expected within the aqueous phase. Due to the construction of the reactor shielding, it is anticipated that microwaves will be bounced around inside the reactor, ensuring incident radiation from all angles. Microwaves are reported to produce localised "hot spots" during heating, although these are expected to even out due to convection in the bulk. Microwaves have a penetration depth (in water) of approximately 1.4 cm at 25 ^{o}C and 5.7 cm at 95 ^{o}C, further attenuated by the cell wall (material dependent). (The penetration depth is defined as the point where $1/e$ (37%) of the original power is present, and is both material- and temperature-dependent.[103]) The penetration depths of the non-reactive alkanes are very high, as little radiation is absorbed. Thus, with rectangular cells of roughly 3 cm a side, this should not pose an issue regarding dead volumes or pockets of fluid. Consistently measuring the temperature at the same point in the cell (the base of the sphere) assists in making the values comparable across experiments.

9.3.4 Systems without surfactants

There have been conflicting reports regarding the response of the alkane-water interfacial tension to changes in temperature. Provided that the interface is chemically pure, interfacial tension should decrease at higher temperatures. This is predicted by the Antoine equation, and has been confirmed experimentally.[80] However, the reverse trend has also been reported with unpurified oils,[10] leading to emphasis on the requirements of chemically pure interfaces in interfacial tension measurements.[60] The phenomena of the interfacial tension increasing as temperature increases is observed in systems containing certain surfactants, including some that occur naturally as alkanes degrade or found in crude oil systems.[81]

The interfacial tension between water and unpurified decane was monitored during irradiation with 60 W for 60 s. (Shown in FIGURE 9.4.)

Neither the microwave nor the temperature changes themselves appeared to have a significant impact on on the interfacial tension of the water-oil interface, although although there was an overall reduction of roughly 5 mN/m, attributed to the system coming to equilibrium prior to irradiation. The measurements show more scatter than observed in the static systems, attributed to moving reflections obscuring certain parts of the interface.

9.3.5 Brine

Salt is known to affect the distribution of other surfactants between aqueous and organic phases by altering the organisation of water molecules. This "salting out" effect, which reduces the affinity of non-polar groups to the aqueous layer, tends to concentrate surfactants in the interfacial zone, causing an increase in the chemical potential of the surfactant and thus decreasing the interfacial tension.[125] FIGURE 9.5 shows that the interfacial tension of the decane-brine system increased slightly during microwave heating, reducing again as the solution cools to reach an interfacial tension slightly lower than the initial value. This agrees with observations where salt (NaCl) was found to reduce the interfacial tension between water and commercial vegetable oils.[58] In contrast, Al-Sahhaf et al.[4] found temperature to have a negative impact on the brine-water interfacial tension in the presence of three industrial surfactants.

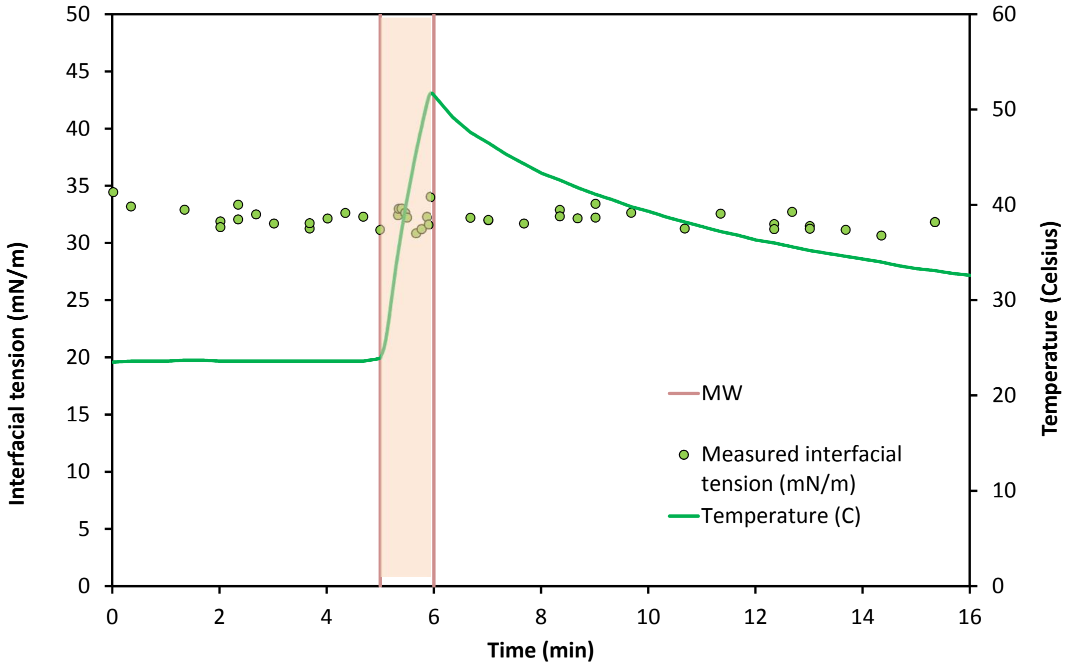

Figure 9.4: The interfacial tension of the decane-water interface, subjected to irradiation at 60 W for 60 s. No significant change in interfacial tension was noted.

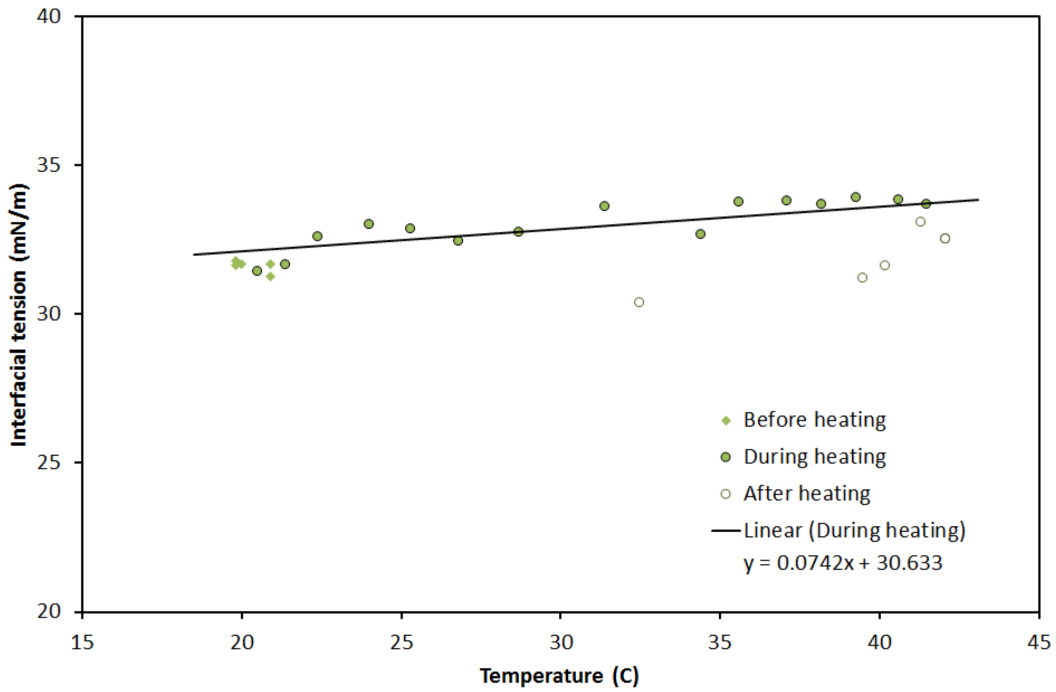

Figure 9.5: A sample of brine (0.01 M) and decane irradiated at 60 W for 60 s.

9.3.6 Charged surfactants: CTAB

In contrast to water, the cationic surfactant CTAB (cetyltrimethylammonium bromide) showed a marked response to microwave irradiation. The interfacial tension appeared to drop suddenly as the microwaves were turned on and off, resulting in a 'Z' shaped pattern within the irradiation window, shown in FIGURE 9.6. A steady decrease in interfacial tension below the initial value is also observed in both cases as the sample cools. In addition, the measurement scatter and number of failed frames is noticeably greater as the concentration increased, but it is unclear whether that is attributable to the clarity of the images themselves (the available fitting area was reduced), or if the charged surfactants, being more susceptible to the switching microwave field, actually induce more movement at the interface.

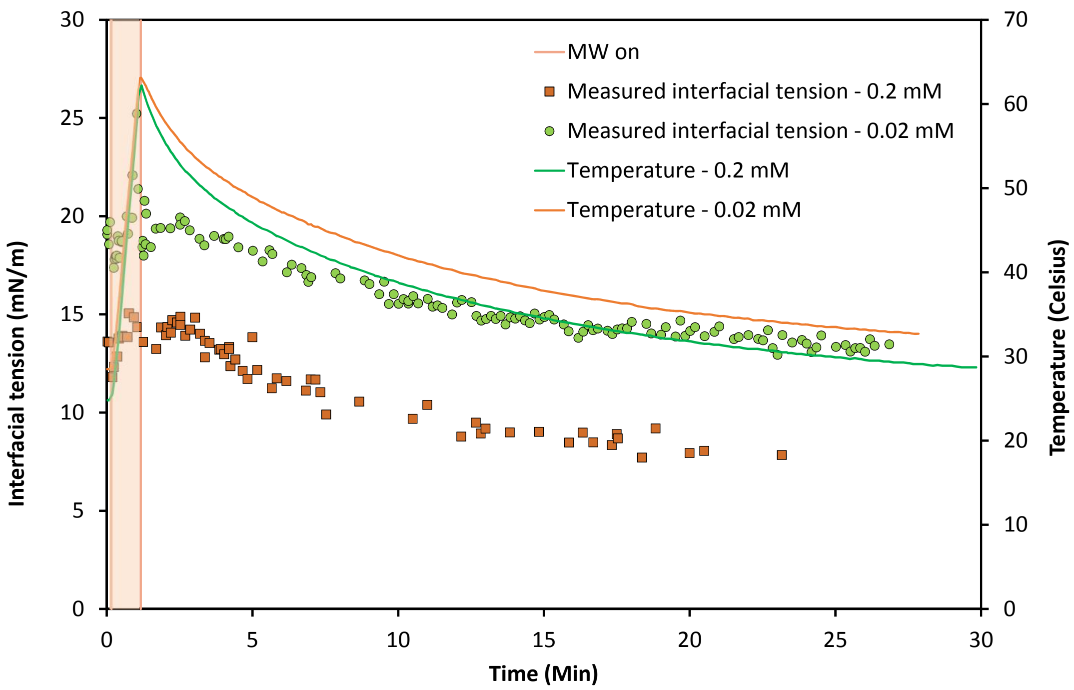

Figure 9.6: The change in interfacial tension of the decane-water interface with the cationic surfactant, CTAB, irradiated at 60 W for 60 s. The trends were observed at two concentrations: 0.2 mM and 0.02 mM

9.3.7 Non-ionic surfactants: Triton X-100

The Triton group of polyethylene glycol octylphenyl ethers ($C_{14}H_{21}O(C_2H_4O)_nH$, commonly referred to as Triton X-10n) are common industrial non-ionic surfactants. The effect of microwaves on the interfacial tension of Triton X-100 ($n \approx 9.5$) was investigated as part of a comparison between microwave and conventional heating. The results are discussed in detail in SECTION 9.4 (pg. 132).

9.4 Comparing conventional and microwave heating (Triton X-100)

9.4.1 The effect of microwave heating on interfacial tension

Microwave measurements with Triton X-100 revealed interesting results. A significant and sudden change in interfacial tension was observed when the microwave was turned on, dropping away rapidly once irradiation stopped. During cooling, the interfacial tension changed more rapidly than did the temperature, eventually dropping below the original value. The interfacial tension was not seen to recover during the 90 minute experiment. An example of the measured interfacial tension and aqueous temperature during the experiment is given in FIGURE 9.7.

The sample was irradiated a second and third time, with sufficient intervening time for the cell to cool back to room temperature. The same rapid increase in interfacial tension was observed each time that the microwave was turned on, and the IFT depressed further each time upon cooling. These trends were observed with different concentrations of Triton X-100, as shown in FIGURE 9.8.

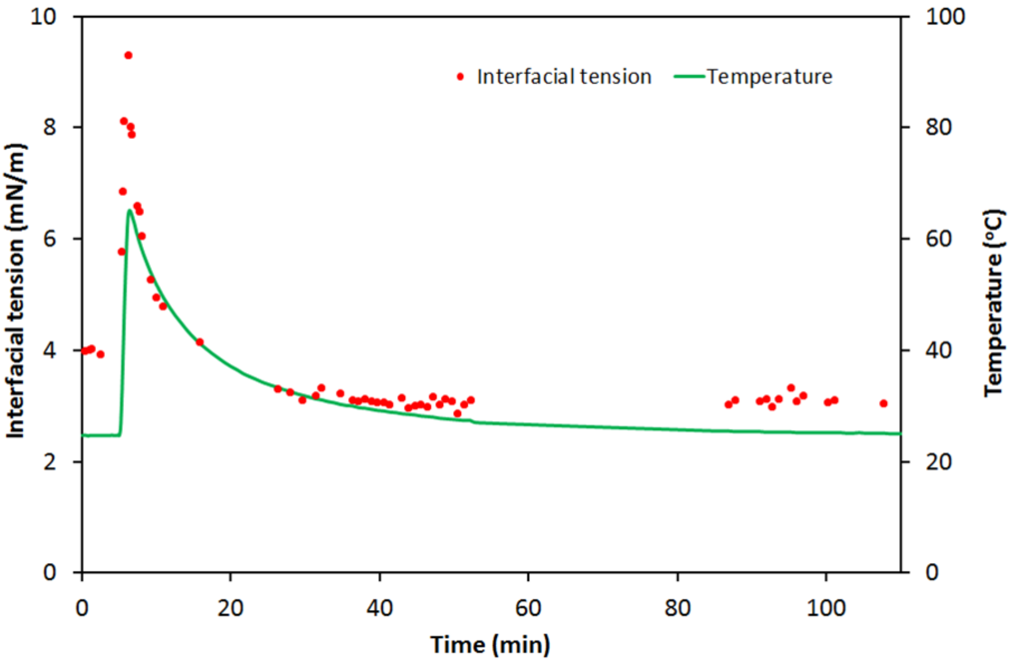

Figure 9.7: The interfacial tension of the decane-Triton (0.66 mM) interface, subjected to irradiation at 60 W for 60 s. The interface changed rapidly during irradiation. *Originally published in Hyde et al.*[79]

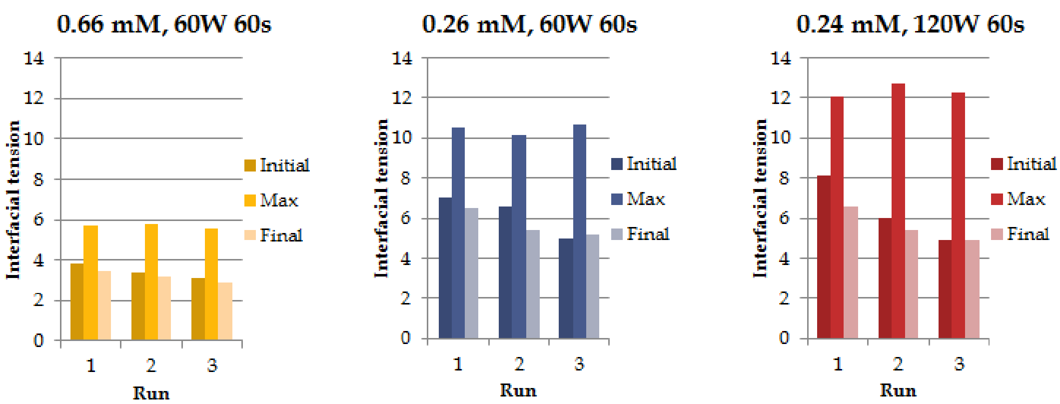

Figure 9.8: The interfacial tension of the decane-water interface in the presence of Triton X-100, subjected to irradiation at 60 W for 60 s. Similar trends were observed across three concentrations.

9.4.2 The effect of conventional heating on interfacial tension

As a comparison, the decane-Triton (0.66 mM) interface was measured over the same 25-60 ^{o}C range as the microwave experiments. The sample cell was set up as previously described, but rather than being placed inside the reactor, was partially immersed inside a hot water bath. The cell was heated to the required 60 ^{o}C, removed from the water bath, and allowed to cool in air. A schematic is shown in FIGURE 9.9.

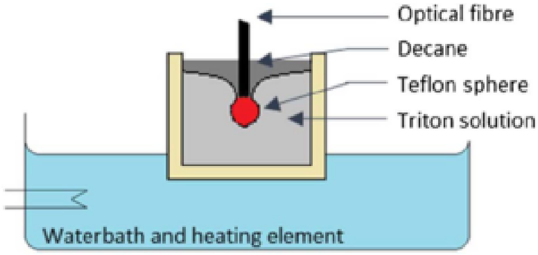

Figure 9.9: A schematic of the setup used for the measurement of interfacial tension using a waterbath for heating. *Originally published in Hyde et al.* [79]

Again, the interfacial tension was found the vary with temperature, indicating that temperature is a large contributor to the effects observed under microwave irradiation. However, as shown by FIGURE 9.12, the shape of the interfacial tension-heating curves are significantly different. Under microwave heating, the increase in interfacial tension was almost doubled, and the recovery occurring at a fairly consistent speed. In contrast, the interfacial tension measured during conventional heating reduced linearly until the cell cooled to roughly 45 ^{o}C, after which it abruptly flattened out. The interfacial tension began to recover slightly after that point. In contrast, there was no evidence that the interfacial tension depressed by the microwaves would recover to its original value.

9.4.3 Hysteresis

Hysteresis was observed in the IFT respone to temperature in systems containing the non-ionic surfactant Triton X-100 and the cationic surfactant CTAB. However, the shape of the curves differed between the conventional heating and microwave trials. Two main differences are noted. Firstly, the increase in interfacial tension

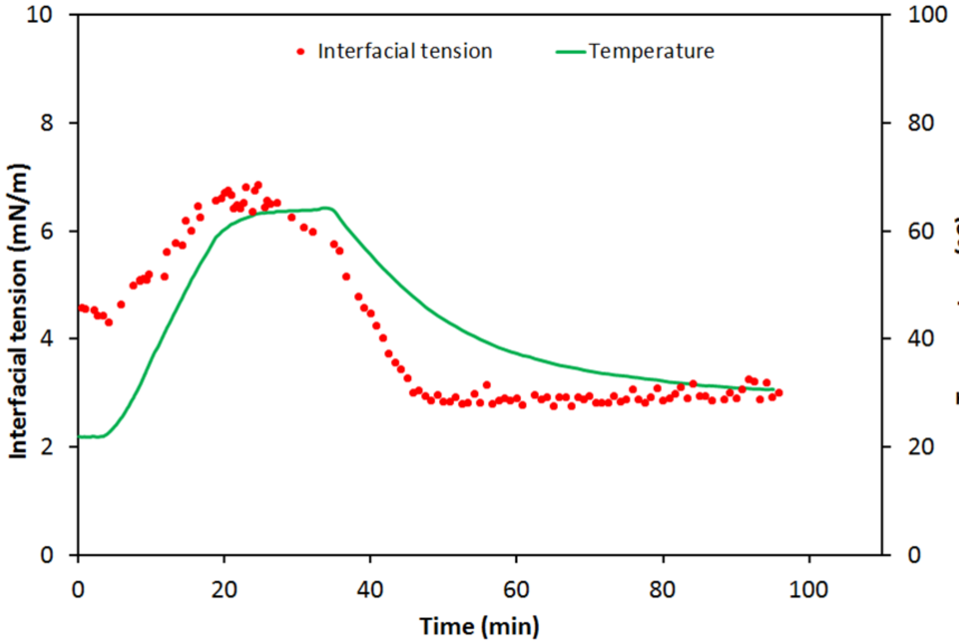

Figure 9.10: Changes in the interfacial tension of the decane-water system with time in the presence of Triton X-100 (0.66 mM) when heated in a waterbath. *Originally published in Hyde et al.*[79].

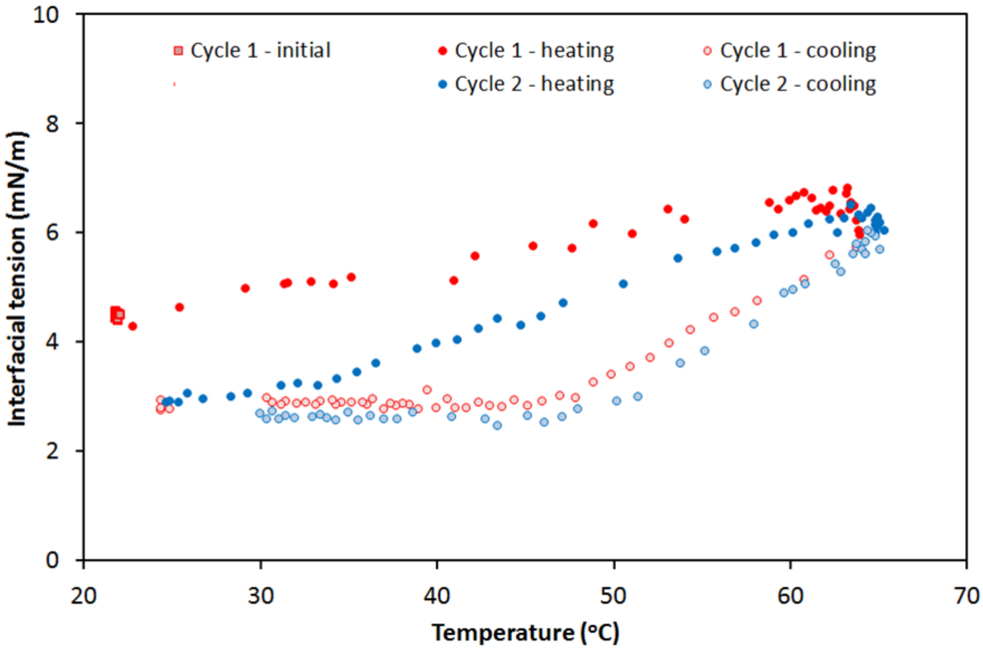

Figure 9.11: Changes in the interfacial tension with the temperature of the decane-water system in the presence of Triton X-100 (0.66 mM) when heated in a waterbath, comparing the IFT-temperature curves for the first and second instances of heating. *Originally published in Hyde et al.*[79]

was significantly more marked during microwave heating. Secondly, the recovery during cooling was much slower in the microwave sample. The heating and cooling curves are compared in FIGURE 9.12, and the effect of heating multiple times in FIGURE 9.11.

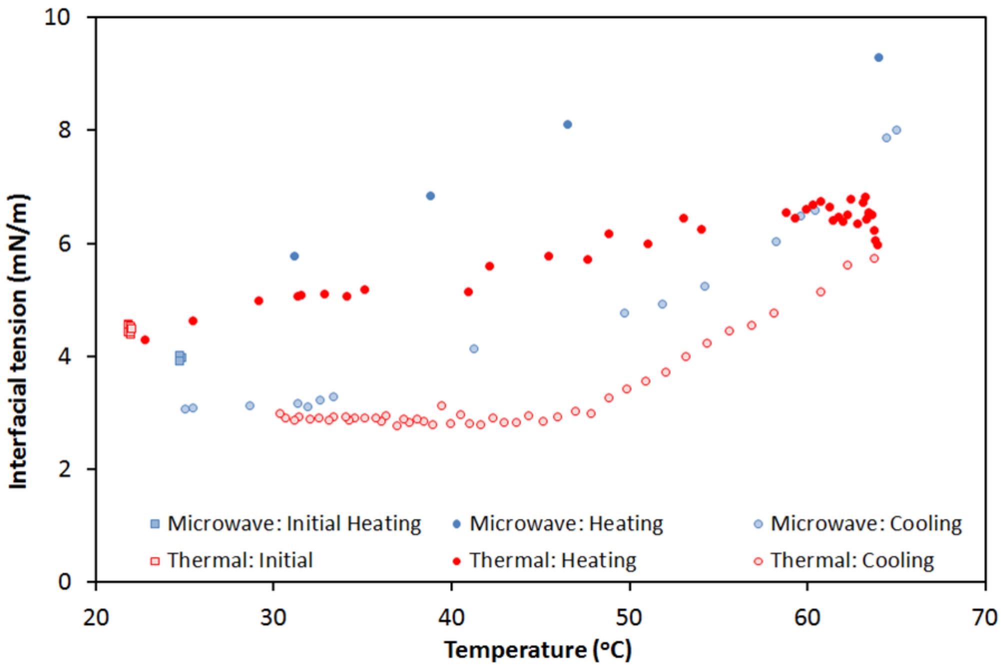

Figure 9.12: Comparing the temperature-IFT curves of the decane-Triton X-100 (0.66 mM) interface when heated in a waterbath and using a microwave. *Originally published in Hyde et al.* [79]

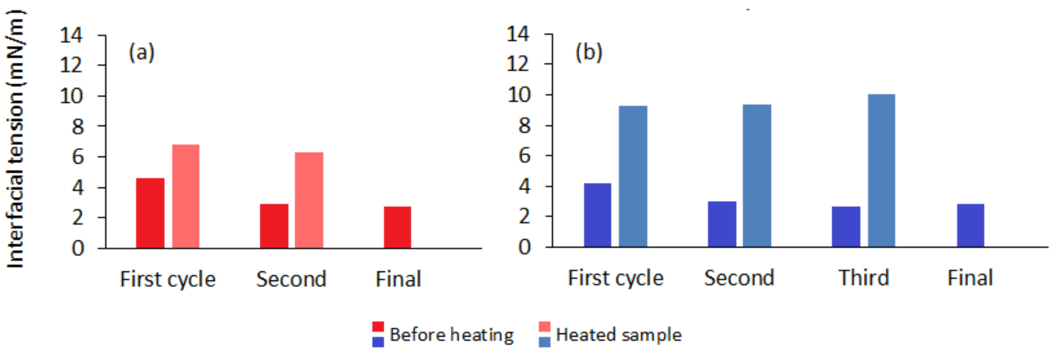

Figure 9.13: Comparing the initial and final values of interfacial tension for 0.66 mM samples of Triton X-100 with decane, heated multiple times. (a) heated by conventional heating, and (b) by a 60 W microwave for 60 s. *Originally published in Hyde et al.* [79]

In both, the interfacial tension of the cooling sample reduces until it falls past the initial point. However, while this depression in interfacial tension is of a similar magnitude, experiments with conventional heating reached this point after only one heating instance, after which the same maximum and minimum values were reached. In contrast, the cooled interfacial tension continued to decrease after three instances of irradiation.

9.4.4 The effect of temperature on interfacial tension

The effect of temperature on interfacial tension depends strongly on the chemistry of the system. The presence and nature of surface active species are major contributors to the observed trends.

While the interfacial tension of pure alkane-water systems is expected, based on thermodynamics, to decrease at higher temperatures,[80] conflicting results have been reported.[10] These discrepancies are attributed to surfactants produced through the natural oxidation of alkanes.[60] A similar phenomenon is observed between commercial and purified vegetable oils.[58] These observations would argue that purified alkanes are imperative for reproducible surface tension measurements. However, there is no way to ensure such purity in unsealed containers for the lengths of time involved. Instead, surfactants were used to overwhelm any naturally occurring surface active species. Changes in surface saturation due the temperature-dependence of the critical micelle concentration are addressed by using concentrations well above the CMC (0.22 mM).[148]

The oil-water interfacial tension is reported to increase with rising temperatures in the presence of Triton X-100,[98] although it is interesting to note that not all members of the series act the same way.[86] The ethylene oxide groups in Triton are less soluble in water at higher temperatures,[98] meaning that the surface concentration is expected to reduce due to a net migration from the aqueous to organic phases at higher temperatures. The micelle aggregation number is sensitive to temperature, and changes to the aggregate structures have been observed by quasi-elastic light scattering spectroscopy (QELSS)[142] and fluorescence measurements.[120] The critical micelle concentration, however, is only marginally affected.[120]

9.4.5 The effect of electromagnetic waves

Our understanding of how molecules orient themselves at interfaces remains incomplete. It is generally accepted that amphiphillic molecules will orient themselves so that like phases are in contact, a state of lower energy. Molecular simulations have gone a long way in increasing our understanding of how molecules act in solution but surface phenomena remain much harder to model.

An atomistic model with second-generation force fields was used to model the hydration of an isolated Triton X-100 molecule in water.[50] The model predicted the hydrogen bonding network surrounding the surfactant: a total of five to 12 hydrogen bonds localized around the electronegative oxygen atoms, the exact number depending on the conformation and steric hindrance around these atoms. The spontaneous associated of Triton molecules was modelled by molecular dynamics found each individual surfactant to be associated with four to 11 water molecules on average.[46] Again, the hydrogen bonds were localised around the oxygen molecules in the tail, as shown in FIGURE 9.14.

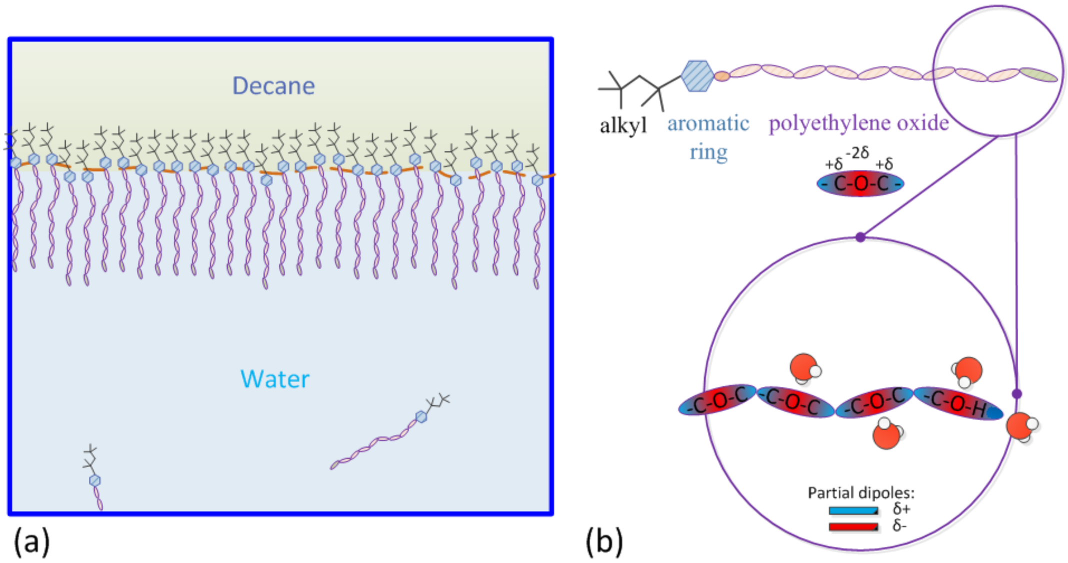

Figure 9.14: Visualising the effect of microwaves on hydrogen bonding networks: (a) adsorption of Triton X-100 at the interface, and (b) proposed bonding structure of the hydrophilic part and water (as determined from molecular simulations.[50]) Triton X-100, t-Octylphenoxypolyethoxyethanol, has an average of 9.5 repeating units per molecule. *Originally published in Hyde et al.*[79]

While simulations of the interfacial behaviour have not been published, some conclusions can still be drawn from the data available. The strength of hydrogen bonding is proposed as the fundamental reason for the higher surface tension of water. Due to its amphiphilic nature, Triton molecules concentrate at the interface until the interface is saturated. Their presence disrupts the hydrogen bonding network of water molecules in the interfacial zone. Furthermore, as the polarity of the C–O–C bonds is less than O–H, the hydrogen bonding between water and water, and Triton molecules and water, must be different, making the two local environments different. Dielectric heating occurs as polar molecules are rotated in a switching magnetic field, and energy from the broken bonds is released as heat. It is conceivable that differences in the local environments can result in heating differences between the bulk (where the temperature is known) and the interfacial layer. This discprepency may account for some of the differences observed between microwave and conventional heating.

9.4.6 The formation of nano-bubbles and micro-emulsions

Aside from local temperature differences, microwaves have been shown to promote the formation of nano-bubbles in water, even below the boiling temperature.[8] The molecular oscillation and disruption of the water bonding network is thought to play a significant role in the formation of these bubbles.[7] Nanobubbles concentrate contaminants or other surface active species at their interfaces. These bubbles rise to the interface, carrying the additional surfactants with them and increasing the interfacial concentration. As a consequence, the interfacial tension is reduced.[133]

Microemulsions can be formed by heating oil and water to the phase inversion temperature, where the two phases are fully miscible, and rapidly quenching the solution, resulting in the formation of water-in-oil and oil-in-water microdrops.[47] Local hotspots around the interface may result in a similar phenomenon on a much smaller scale, due to changes in the alkane-water solubility. The solubility of decane in water rises from 0.007% to 0.014% as the temperature is raised from 25 ^{o}C to 40 ^{o}C.[138] It is conceivable that small, localised microemulsions may form around the interface, contributing to the long-term reduction in interfacial tension and concentrating surfactant molecules at the interface.

9.4.7 Disolved gasses

The presence of dissolved gasses is known to increase the interfacial tension of crude oils.[93] However, gas solubility decreases at higher temperatures. Dissolved gasses forced out of solution during heating may be prevented from returning by the organic layer. This may be a contributing factor in the depressed interfacial tension after cooling.

9.4.8 Thermal degradation

While alkanes are considered inert to microwaves, they do actually adsorb a very small proportion of the energy.[150] Alkanes have a dielectric constant of 0.076,[150] significantly smaller than the 76.7 of water,[130] which results in the marked difference in the speed of heating. It is well known that purified alkanes, if allowed to react with oxygen, will oxidise rapidly, producing small quantities of surface active chemicals, and thermal degradation of crude oils has been reported during microwave irradiation.[18]

9.5 Advantages offered by the holm method

One of the key differences separating the holm meridian from pendant and sessile drops is the presence of a fixed solid object and the inherent stability that implies. The holm meridian thus allows measurement of systems where pendant and sessile drop techniques fail – systems with some sort of vibration or other movement, including strong convection currents induced by temperature gradients in the sample. A side effect of a stable interface ensures that the same interface is present (as opposed to a pendant drop falling off an a new one created from the bulk solution). The holm also overcomes issues such as evaporation, making it an ideal method for long-term analysis. For example, in a sample of 6.5 mL water and 4 mL of decane (approximately 27 g), only 0.1 g was lost over the length of the experiment.

9.5.1 Measurements in non-quiescent samples

Maintaining an interface in a system when mechanical forces of some sort or another are present is difficult with the fragile pendant drop. The accuracy of the pendant drop measurement technique is enhanced as the drop deformation and necking increases. The effect is that higher accuracy is obtained from less stable drops. Consequently, pendant drops are exceedingly susceptible to vibration and other movement, and even more so in liquid-liquid systems where the interfacial tension is reduced. Furthermore, vibrations, leading to drop oscillation, will significantly affect the quality of the image and can thus induce significant errors in the analysis.

Conversely, the holm technique works well for two liquid systems. The interface is stable and the solid object is easily affixed. External vibrations do not cause the same stability issues, with no risk of the aqueous phase coming detached from the sphere unless the ball is depressed to its extreme limit. Convection currents are potentially a greater problem, depending on lighting, as they can induce rapidly moving reflections in the upper fluid that can make image analysis difficult. This issue can be effectively minimised by good lighting and appropriate edge detection algorithms, as developed in this thesis.

It is worth noting, however, that the number of failed frames (frames that could not be analysed due to issues with image analysis) was significantly higher when the microwave was running then when the machine was turned off. Two reasons spring to mind. Firstly, as can clearly be seen, reflections move rapidly across the image, obscuring parts of the interface and making edge detection difficult. Secondly, the microwave does produce significant vibrations, which translates to some movement of the fluids in the sample and may have some unquantified effect.

9.5.2 Measurements over long periods of time

The ability to maintain the same interface over a period of hours or days is of great importance in dynamic studies of slowly-equilibrating surface-active chemicals. The adsorption of proteins are an example of this. Maintaining a pendant drop over a period of hours or days can be difficult, and the experiment is wasted if the drop should fall as a new drop will possess a fresh interface. As part of our studies on the effect of microwaves, the basic experiment was run over a period of two to six hours, and long-term measurements were run on the same interface over several days.

Evaporation is also a significant issue affecting long-term measurements on pendant drops. It is possible to reduce the rate of evaporation by saturating the cell with water vapour. However, as the drop volume is itself quite small, analysis at high temperature can be particularly challenging. Parmar et al.[109] also reported boiling in smaller drops.

9.5.3 Measurements in a fully-enclosed space

Lastly, the ability to completely enclose the system is an important attribute for industrial applications. The ambient temperature and pressure conditions of a typical laboratory do not reflect the high temperature high pressure systems where knowledge of the interfacial tension can provide useful information. While pressurized cells have been used with ADSA,[145] pumping is typically required to produce the drop. Pressurising the pendant drop is difficult as the syringe must also be pressurized. In contrast, the holm can be easily enclosed and pumping is not required, making it a good candidate for pressurization and mimicking industrial conditions.

9.6 Chapter summary

The holm method was used successfully for the *in situ* measurement of the oil-water interfacial tension in the presence of various salts and surfactants. The vibration from the microwaves increased the number of failed frames in the analysis and increased the scatter during some fittings. However, unlike a pendant drop, the holm method was robust and remained stable for hours or days despite the vibration.

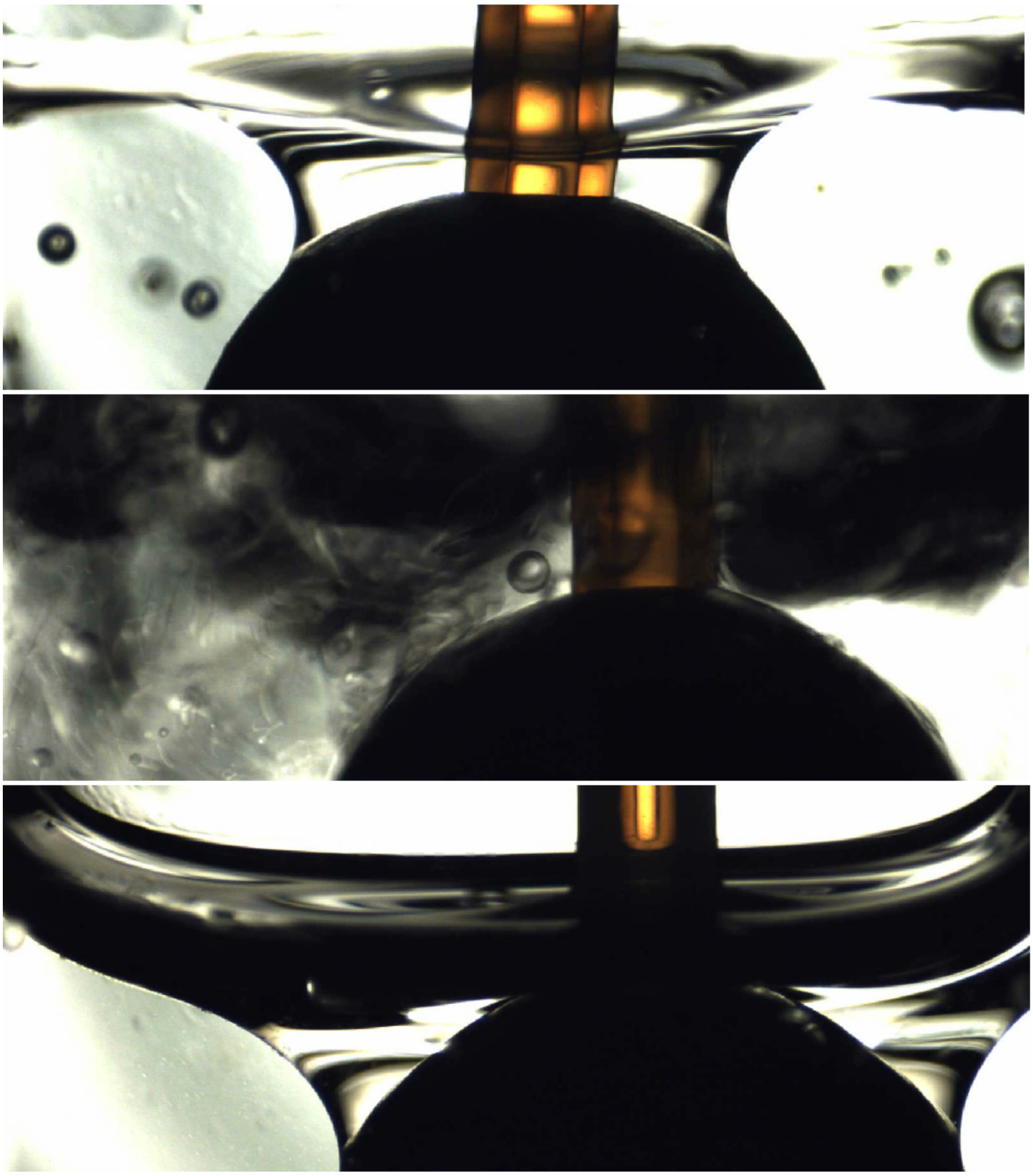

Figure 9.15: Showing an oil-water interface heated to boiling inside of the microwave reactor. While the boiling completely obliterated the holm meridian, the miniscus reformed spontaneously as soon as boiling stopped. Unlike a pendant drop, where creating a new drop would produce as new liquid bulk, the interface here forms from the original liquid bulk.

The interfacial tension of the decane-water system was found to decrease only slightly during irradiation. In contrast, the interfacial tension increased slightly at higher temperatures in the presence of brine. CTAB solutions, even very low concentrations, were found to react strongly with microwaves. The change in interfacial tension was evidenced in visible changes to the shape of the holm. The increase in interfacial tension as temperature increased was noted at different concentrations. During cooling, the interfacial tension lowered further than the original start value.

The oil-water interface with Triton X-100 was measured during heating both by microwaves and conventional methods (a hot water bath). The interfacial tension increased with temperature during both heating methods. However, the extent of the increase was nearly doubled when heated by microwaves, possibly due to localised heating. Although cooled from the same temperature with near-identical temperature profiles, the tension recovered faster in samples heated by a water bath. During cooling, the tension recovered more quickly than temperature, eventually dropping below the original values. If heated and cooled a second time, the depressed value returned to the same point if conventional methods were used. However, using a microwave, the tension reduced successively further for at least three cycles. As microwaves are known to promote the formation of nanobubbles and micro emulsions, it is thought that these process may play a part in the decreasing interfacial tension.

The results demonstrate the applicability of the holm method to dynamic systems influenced by vibrations and currents within the sample, and its suitability for use inside of an enclosed space.

CHAPTER 10

Modelling the dynamic tension of the decane-carboxylic acid interface in response to changes in pH

10.1 Introduction

Stimuli-responsive surfactants have seen significant recent interest.[31] The ability to modify the properties of a surfactant through external stimuli, such as light, or through properties of the system, such as pH or ionic strength, is an attractive proposition for industrial purposes as they represent potentially reversible effects. While a variety of pH-responsive surfactants exist, an aqueous solution of a carboxylic acid is one of the most simple examples.

The petroleum industry frequently makes use of a technique known as "alkaline flooding" to improve oil recovery. By pumping a caustic solution into wells containing acidic oils – crude oils which contain large quantities of sulfur or naturally-occurring carboxylic ("fatty") acids – the reaction of the caustic with the carboxylic acids produces the surface active carboxylate species. In other words, an example of *in situ* soap production. The carboxylate ion dramatically lowers the oil-water interfacial tension, overcoming capillary forces in the rock pores and significantly improving the mobility of the oil.[43]

It has been known for some time that flooding with alkaline solutions can significantly improve the recovery of certain types of crudes. However, for some time, there was little understanding of the chemistry or mechanisms behind the behaviour that was observed. In the 1980s, a series of models [19,20,38,43,119,135,136,147] were proposed by various groups, attempting to predict the behaviour of various crude oils in contact with caustic. Nonetheless, a simple and effective method to model the dynamic interfacial behaviour is yet to be developed.

This study presents a simple model which describes the behaviour of fatty acids at the aqueous-oil interface as the pH of the aqueous layer is changed from high pH (carboxylate ion) to low pH (carboxylic acid). The interfacial tension is described as a function of the carboxylic acid/carboxylate ion components, which can in turn be described as a function of the pH. The strength of the holm configuration for measuring the interfacial tension is demonstrated, as the entire experiment can be conducted on a single interface, and hence a single bulk solution, allowing the dynamic effects to be observed. Furthermore, the aqueous solution can be stirred rigorously without destroying the interface, making possible to measure the equilibrium state on the same interface. The large interfacial area offered by the holm provides space for a pH probe, allowing the acid concentration to be measured close to the interfacial layer throughout the experiment.

This chapter details the use of the holm method for *in situ* measurement of the non-equilibrium tension of the hexadecane-carboxylic acid interface. This chapter is partly based on a submitted paper, [77] which features a model to predict the dynamic interfacial tension of the carboxylic acid-carboxylate system.

10.2 The system and reaction equilibria

10.2.1 Equilibrium states

This study considers the effect of pH on the carboxylate/carboxylic acid system and the corresponding oil-aqueous interfacial tension. For universality, AH and A^- are used to represent a generic carboxylic acid in its protonated and deprotonated states, respectively. The carboxylic acid is almost completely dissociated in a strongly alkaline system. Given sufficient time, the interface will saturate with carboxylate ions and the interfacial tension will be at its lowest. Conversely, the acid will be fully associated at low pH and interfacial adsorption will be at a minimum, with the adsorbed species diffusing to either the aqueous or organic phases. These limiting conditions are shown in FIGURE 10.1.

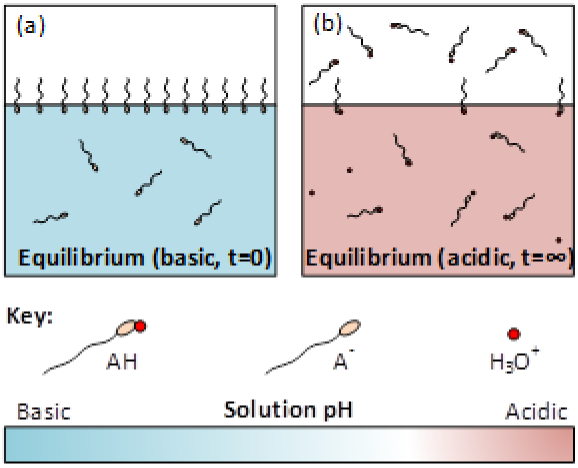

Figure 10.1: The status of the systems at equilibrium, showing the distribution between phases at (a) high and (b) low pH. Excess sodium ions maintain charge neutrality. The top and bottom layers represent, respectively, the organic and aqueous phases. *Included in Hyde et al.* [77].

Consider highly alkaline system at equilibrium, such that the interface is saturated with carboxylate ions. If a single drop of concentrated acid is added to the bulk without stirring, hydronium ions will diffuse slowly through the solution, altering the equilibrium between protonated and deprotonated acid. When this reaction occurs at the interface, the now neutrally charged acid loses its surface affinity and diffuses to either phase. Consequently, the interfacial tension increases. In this way, the dynamic interfacial tension is affected directly by the solution pH, as will be shown.

The system can be summarised as follows:

- The acid starts fully dissociated in the aqueous phase,
- In a dynamic equilibrium, A^- migrates to the interface, from whence it can either desorb back into the aqueous phase, or collect a hydrogen atom and desorb into the oleic phase as AH,

- The concentrations of A^- and AH are governed by the acid dissociation equilibrium, and the partitioning between the oelic and aqueous phases is governed by the partition constant,
- Mass transport throughout the system is governed by mollecular diffusion.

It is assumed that the protonation/deprotonation reaction occurs very rapidly, such that the changes in interfacial tension are diffusion-rate limited, and relate the model to the speed of the hydronium ion diffusing through the bulk.

10.2.2 The carboxylate/carboxylic acid equilibrium

Carboxylic acids, such as nonanoic acid, partially dissociate in aqueous solutions. The equilibrium between the carboxyalate ion and its conjugate acid is described by (10.1) below:

$$AH + H_2O \rightleftharpoons A^- + H_3O^+ \tag{10.1}$$

The equilibrium constant for (10.1) is given by:

$$K_a = \frac{[A^-][H_3O^+]}{[AH]} \tag{10.2}$$

A higher equilibrium constant indicates a greater tendency for the acid to dissociate, producing the surface active species. Using the pKa, it is possible to predict the extent of ionisation of the acid-water system at any concentration of the hydronium ion, given by the pH.

10.2.3 The oil-water equilibrium

The carboxylic acids, which are only sparingly water-soluble, partition between the oil and water phases according to the partition coefficient, K_D. The partition coefficient is defined as the ratio between the concentrations of the acid in the two phases:

$$K_D = \frac{[AH]_o}{[AH]_w} \tag{10.3}$$

The affinity of the acid for the two phases is affected predominantly by the tail length. The charged deprotonated form is insoluble in the non-polar organic layer.

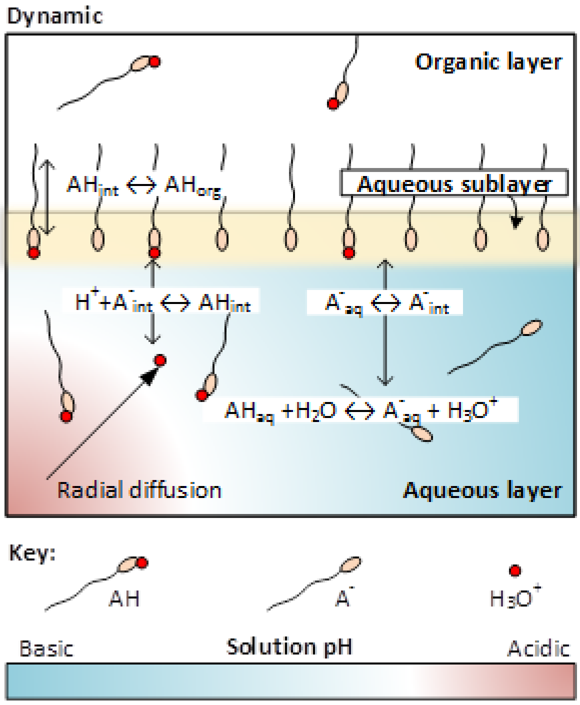

Figure 10.2: Key species, their equilibria and distribution through the two phases. *Included in Hyde et al.*[77].

10.2.4 Acid distribution between phases

Expressing the volumes of the two phases as V_w and V_o, the distribution of the acid between the phases can be described by the molar balance:

$$V_w[AH]_w] + V_o[AH]_o + V_w[A^-]_w = V_w[AH \cdot A^-]|_{t=0} \tag{10.4}$$

Where $V_w[AH \cdot A^-]|_{t=0}$, based on the concentration of the stock solution, gives the total number of moles of the acid (in any form) in the system.

Substituting (10.2) and (10.3):

$$V_w[AH]_w + V_o(K_D[AH]_w) + V_w\left(\frac{K_a[AH]_w}{[H_3O^+]}\right) = V_w[AH \cdot A^-]|_{t=0} \tag{10.5}$$

Rearranging:

$$[AH]_w = \frac{[AH \cdot A^-]|_{t=0}}{1 + \frac{K_a}{[H_3O^+]} + \frac{V_o}{V_w}K_D} \tag{10.6}$$

The concentrations of AH_o and A_w^- can then be determined from (10.2), (10.3) and (10.6).

10.2.5 Sodium salts

It has been proposed by various sources that the charged carboxylate ion may pick up a strongly associated sodium ion. However, there is some dispute as to solubility of this compound. Chan and Yen[38] proposed that these strongly associated compounds remain in the aqueous layer, potentially precipitating out. This is not an unreasonable assumption, both because the vast majority of the charged species will be located in the aqueous bulk (as the volume of this region is significantly higher than the volume of the interface), and because there are numerous reports of precipitation in the aqueous layer. Observations of precipitation is inconclusive, however, as the sparingly soluble acids may actually be precipitating in their protonated form. Other reports observed no precipitate even at high sodium concentrations.[119] Some models propose that these salts, which have no charge, may be sufficiently oil-soluble to migrate into oil layer if they were absorbed at the interface when in their charged state, such as the heptane-soluble sodium palimate.[19]

In alkaline flooding, where the fatty acids are found initially in the organic phase and so must migrate to the interface to deprotonate the alcohol, it is reasonable to assume that the vast majority of the acid will remain in the organic layer, including the organic-soluble sodium salts.[19,119] However, as the present system starts with the acid dissolved in the aqueous layer in a highly basic solution, the acids will react throughout the aqueous layer. Since the number of molecules in the aqueous bulk far exceeds the quantity in the interfacial layer, for this simplified model, we neglect transfer of the sodium salt into the organic phase.

10.3 Surface adsorption

Models of the dynamic interfacial tension of surfactant systems where the surfactant must diffuse from the bulk to the interface have been proposed.[114] In these systems, however, the species responsible for changes to the interfacial tension is the same as the species diffusing through the bulk. While in this system it is the hydronium ion which diffuses throguh the cell, the carboxylate ion is the surface active species.

At the alkaline equilibrium, carboxylate ions saturate the interface with surface concentration Γ_{A^-}. The hydronium ion diffuses through the cell, altering the bulk pH, and neutralising the carboxylate ion. The occurance of reaction (10.1) at the interface can be interpreted as the adsorption of a hydronium ion to the interface. This allows the adsorption of hydronium to be modelled through a Langmuir isotherm (10.7) where the maximum surface adsorption of hydronium, Γ_m (mol/m^2) is equal to the saturation concentration of the carboxylate ion in the alkaline solution. The presence of an aqueous sublayer is proposed, with a hydronium concentration of $c_{s,H}(t)$ (M). The Langmuir isotherm describes the equilibrium between this layer the true interface, with an adsorption constant of K_L (M^{-1}).

$$\Gamma_H(t) = \Gamma_m \frac{K_L c_{s,H}(t)}{1 + K_L c_{s,H}(t)} \tag{10.7}$$

10.4 Mass transport

Ramakrishnan and Wasan[119] proposed a method to calculate the surface concentration Γ_{A^-} from the equilibrium of the forward and reverse diffusion equations. However, the method is inhibited by requiring a number of difficult-to-measure physical constants: the desorption energy barrier, W, the forward and reverse rate constants k_1 and k_2, and the partition coefficient, K_D.

As an alternative, this model uses the Ward-Tordai equation[151] to describe the diffusion of the hydronium ion through the quiescent bulk. In (10.8), D_H is the diffusivity of hydronium (m^2/s) and τ is a dummy variable for integration.

$$\Gamma_H(t) = 2\sqrt{\frac{D_H}{\pi}}\left(c_{b,H}\sqrt{t} - \int_0^{\sqrt{t}} c_{s,H}(\tau)\mathrm{d}(\sqrt{t-\tau})\right) \tag{10.8}$$

The bulk concentration, $c_{b,H}$ (M) refers to the equilibrium hydronium concentration in the bulk:

$$-\log_{10}(c_{b,H}) = \mathrm{pH}_{t=\infty} \tag{10.9}$$

The surface concetration of hydronium is determined by solving (10.7) and (10.8) simultaneously.

10.5 Equilibrium interfacial tension

For solutions of a single surfactant, the effect of the surfactant concentration on the equilibrium interfacial tension (γ) is described by the Gibbs equation. However, the interfacial tension of this complex system is a function of the the surface concentrations of the three reacting species:

$$\gamma = \mathrm{fn}(\Gamma_{A^-}, \Gamma_{AH}, \Gamma_{H_3O^+}) \tag{10.10}$$

While it is possible to expand the Gibbs equation for multiple species, this requires a knowledge of the species' interactions and non-ideality expressed as the species' activities,[64] which are typically not known. In very few occasions is it appropriate to use the bulk concentrations in the place of activities.[143] For example, for low concentrations of decanoic acid, where the activities could reasonably be replaced by the bulk concentrations, the effect of solution pH on the aqueous surface tension was modelled using the Gibbs equation.[11] However, the higher concentrations of the more soluble acids and the partitioning of the acids between the two phases makes this approach unfeasible.

A few models propose to model the equilibrium interfacial tensions of systems with pairs of non-reacting surfactants. One model uses surface tension data or adsorption isotherms of the individual surfactants to estimate the interfacial tension of binary surfactant mixtures.[54] In another, the synergistic adsorption of two surfactants was modelled using interaction parameters.[137] Neither model can take into account reactions between surfactants, and are quite complex even when considering only two surfactants. Instead, a simple empirical model will be proposed.

SECTION 10.2 (pg. 148) outlined the reactions and their equilibria. As can be seen, the concentrations of H^+, AH and A^- are interrelated, and the concentrations of all three species can be calculated from the pH if the constants K_a and K_D are known. The effect of bulk surfactant concentration on the interfacial tension has been modelled successfully using empirical equations, where the equation took the form of an exponential decay function.[113] Likewise, this model uses a sigmoidal function to describe the effect of bulk concentration using a single parameter, χ (M^{-1}):

$$\gamma = \gamma_a - (\gamma_a - \gamma_b)\exp\left(\chi[H_3O^+]\right) \tag{10.11}$$

The interfacial tension of the system under acidic (fully associated) and basic (fully dissociated) conditions are given as γ_a and γ_b, respectively. These points are determined from the experimental data, being the measurements taken at the beginning (γ_b) and end (γ_a) of the experiment.

Under the assumption that the neutralisation reaction is significantly faster than the speed of diffusion of the hydronium ion, (10.11) can be assumed to hold under dynamic conditions with $[H_3O^+]$ replaced by the subsurface concentration, $c_{s,H}$. As $t \to \infty$ and the system tends to equilibrium, $c_{s,H}(\infty) \to c_{b,H}$ and $\Gamma_H(\infty) \to \Gamma_{eq,H}$.

10.6 Interfacial tension measurements

Nonanoic acid and hexadecane (all >98%) were purchased from TCI Chemical and Sigma Aldrich and used as received. A 2.6 mM solution of nonanoic acid was made up in DI water and the pH adjusted above 11 with concentrated sodium hydroxide. A quartz cell was partially filled with 25 mL of solution, covered with a thin layer (12 mL) of hexadecane and stirred vigorously for 15 minutes. The resulting meniscus was deformed by submerging a 6.35 mm diameter Teflon sphere. High-resolution images of the interface were taken using a digital camera. The images were analysed using the holm method to determine the interfacial tension.

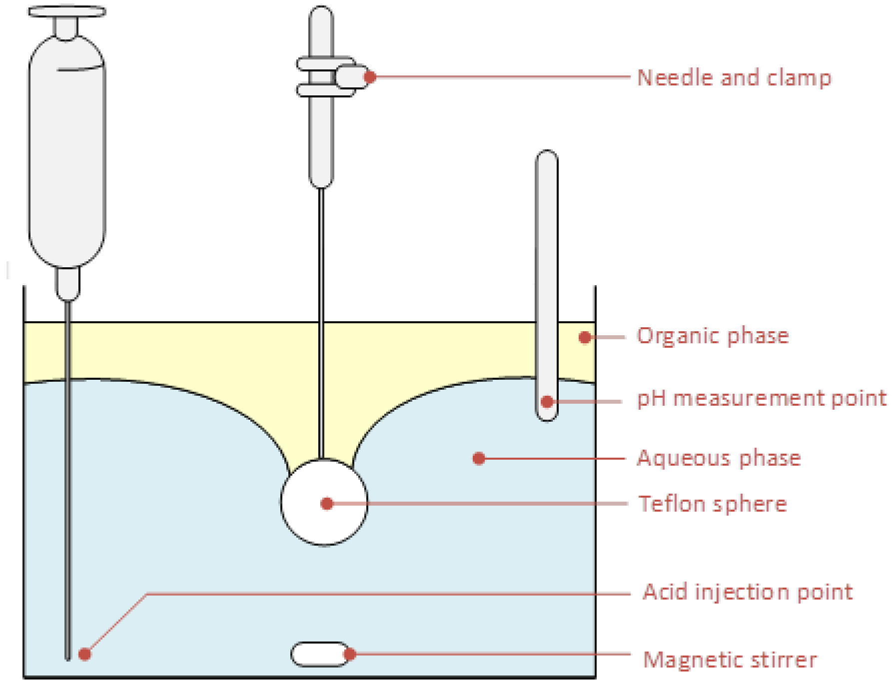

Figure 10.3: Schematic showing the experimental setup. *Included in Hyde et al.*[77].

The solution pH was measured using a (Aqua-pH from TPS Instruments Ltd.) pH meter in one corner of the cell, with the sensor positioned just below the interface. At $t = 0$, hydrochloric acid (30%) was injected at the opposite corner and allowed to diffuse slowly, without stirring, with the pH and interfacial tension being measured at one minute intervals. FIGURE 10.4 shows example images used for fitting. The mixture was stirred for five minutes to ensure that the aqueous solution was fully homogeneous once a constant pH reading was obtained. The pH and interfacial tension of this 'equilibrium' solution was also measured. It is notable that the holm method allows the solutions to be stirred, which causes significant disturbance of the interface, while maintaining the same bulk interface. This is a significant advantage of the holm method. Conversely, the interface can be monitored throughout the experiment without further contact with the interface, unlike the plate and ring methods, which disturb the interface.

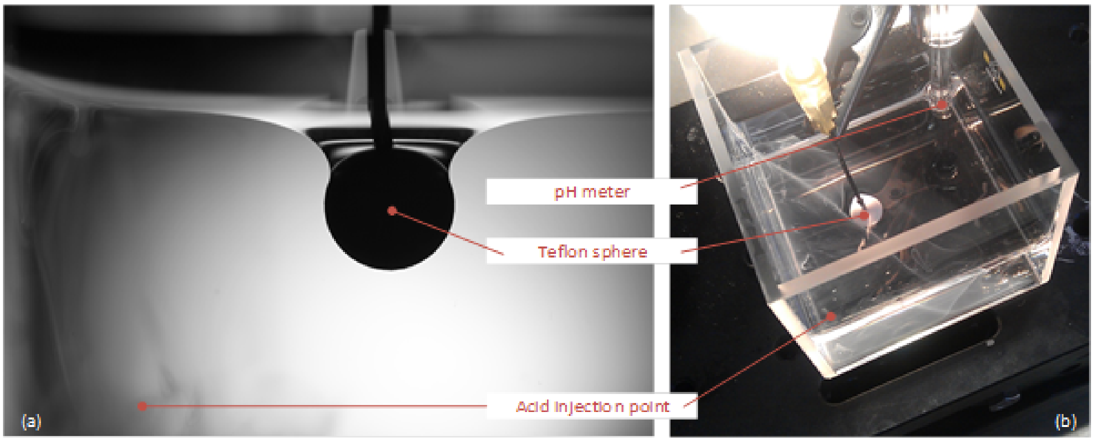

Figure 10.4: Images of the hexadecane-nonanoic acid interface: (a) An image used for fitting. (b) A solution of nonanoic acid near its solubility limit, showing the white precipitate marking the diffusion of hydronium ions through the cell. A lower concentration of nonanoic acid was used for the analysis, to ensure that no acid precipitated. *Included in Hyde et al.*[77].

10.7 Dynamic modelling

Modelling of the dynamic interfacial tension was achieved by a three-step process:

- The surface concentration of hydronium was calculated by solving (10.8) and (10.7) simulaneously.[113]
- The interfacial tension was predicted from (10.11)
- Γ_m, K_L and χ were optimised simultaneously to minimise the fitting error (10.12). Optimisation was carried out using Excel **Solver**.

$$\Delta^2 = (\gamma_{(mes)} - \gamma_{(model)})^2 \tag{10.12}$$

FIGURE 10.5 shows the predictions of the dynamic model. The model predicts the change in interfacial tension based on the diffusion of hydronium ions through the bulk. The concentrations of the carboxylic acid and carboxylate ion can be predicted if the two constants, K_a and K_D, are known.

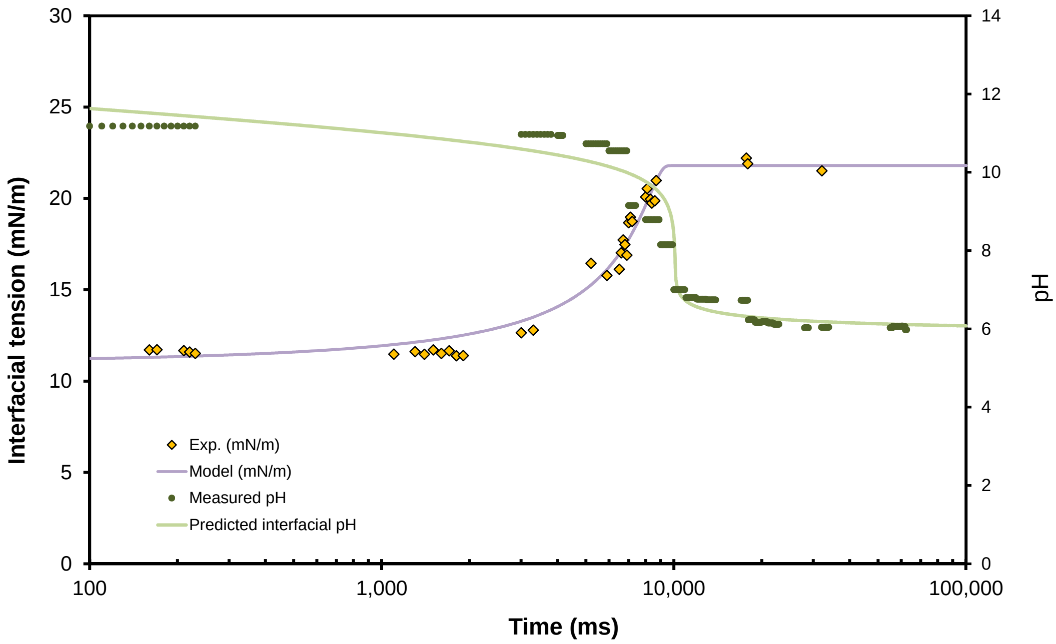

Figure 10.5: Modelling the dynamic interfacial tension. The time $t = 0$ is defined from the time of acid (HCl) injection. *Included in Hyde et al.*[77].

Table 10.1: Fitting parameters and physical properties

Parameter	Value
Equilibrium coefficients for ST prediction:	$\gamma_a = 21.8$ mN/m
	$\gamma_b = 11.0$ mN/m
Diffusion coefficient (hydronium, D_H):[3]	9.3×10^{-5} cm^2/s
Equilibrium constant (nonanoic acid, K_a):[121]	3.80×10^{-4} M
Dynamic model parameters:	$\Gamma_m = 1.15 \times 10^{-7}$ mol/m^2
	$K_L = 4.73 \times 10^7$ m^3/mol
	$\chi = -9.31 \times 10^9$ M^{-1}

10.8 The effect of pH on interfacial tension

10.8.1 Evaluating the empirical model

The charged carboxylate form of the fatty acids acts as an *in situ* surfactant. The tension is significantly lower at the start of the experiment, when the pH is high and the acid fully dissociated into its ionised form. As acid is pumped into the system and the pH reduced, the interfacial tension climbs steadily until it reaches a new equilibrium value corresponding to the fully associated acid. Changes to the pH of the system has no effect on the interfacial tension while the fatty acid remains either fully ionised or fully associated. This implies that the concentration of the hydronium ion itself has minimal effect on the interfacial tension: it is the concentrations of the two acid species that significantly affects it. A sigmoidal function is the most appropriate choice to fit such data. Because the concentrations of the three species are interrelated, a single-parameter sigmoid is sufficient to model the equilibrium interfacial tension relation.

10.8.2 Evaluating the dynamic model

As was shown in FIGURE 10.5, the measured bulk pH lags behind the predicted surface pH of the model. This is consistent with differences in the rate-determining steps of the two interfaces: the measurement of pH at the probe is controlled by diffusion and adsorption onto the surface of the sensor. In contrast, the interfacial concentration depends on diffusion and adsorption to the liquid interface, as well as the interfacial reaction. However, the two concentrations can be seen to coincide at equilibrium, as would be expected.

10.8.3 Measuring hydroxide diffusion: the reverse direction

Two important simplifications are achieved by beginning with a highly alkaline system. Firstly, the acid can be reasonably assumed to exist in the aqueous phase only, as the charged ions are insoluble in the organic phase. Consequently, movement of the acid between phases will be in one direction only. Secondly, it is assumed that the interface is near-saturated with the carboxylate ion. Thus, the diffusion of hydronium ions become the rate-determining step, and the extend of diffusion can be monitored by the neutralisation reaction.

The reversed model, taking the diffusion of hydroxide ions through an initially acidic solution, removes those assumptions. The acid is significantly more soluble in the organic phase than in the aqueous layer[19] and so the majority of the acid will begin in the organic phase and migrate into the aqueous phase. The carboxylic acid does not start at the interface – after reacting with the hydroxide ion, it will need to migrate to the interface to affect the interfacial tension. Between the three diffusion processes – hydroxide, carboxylate ion and diffusion of the acid through the organic layer – and the adsorption/desorption process of the protonated acid at the interface, it is not clear what the rate determining step will be. Modelling the reverse direction will require a more detailed knowledge of the diffusion rate constants and adsorption equilibria.

10.9 Chapter summary

As industrial systems are rarely at equilibrium, the ability to predict dynamic interfacial tension is often of more practical value than equilibrium measurements. In general, dynamic effects are governed by the length of time required for a species to diffuse to the interface and affect the interfacial tension. In these rate-limited systems, the changing interfacial tension can be predicted from the diffusion of the active species.

In this model, the neutralisation of carboxylate ions at the oil-water interface is predicted from the diffusion of hydronium ions through the bulk. The model predicted the rising interfacial tension resulting from protonation of the surface active carboxylate ion, as well as the concentrations of the other species in solution. The interfacial tension of a complex reacting system was modelled using a simple empirical relationship. The model used the unique capabilities of the holm method to measure the interfacial tension over long periods of time without disturbing the sample.

CHAPTER 11

The effect of a magnetic field on interfacial tension

11.1 Magneto-responsive surfactants

Stimuli-responsive surfactants have generated significant recent interest for their ability to affect changes to a system after receiving some type of external stimulus. These responsive surfactants offer the potential to reversibly alter the interfacial properties of a system without direct contact. Surfactants have been developed which respond to a range of factors, including changes to pH, light, electrical potential and various chemicals.[31] Amongst these, magneto-responsive surfactants are sensitive to changes in an external magnetic field. Certain types of nano-particles and metallic complexes, including ferrofluids, are also responsive to magnets.

11.1.1 Ferrofluids

Magnetic nanoparticles stabilized by surfactants and suspended in a solvent are referred to as "ferrofluids".[5] Under the influence of a magnetic field, the nanoparticles are pulled towards the magnet, much like any magnetic solid.[92] In addition, the impact of magnetic fields on ferrofluids is realised in terms of changing viscosity[104] and rotation of the nanoparticles.[122] There have been reports of using magnetic particles to assist in demulsification of oil-water systems. For example, if an oil-water emulsion is treated with surfactant-coated magnetite nano-particles, the surfactant coating ensures that the particles accumulate at the oil-water interfaces. When they are separated from the emulsion by a strong magnetic field, the entrained oil phase is pulled with them, providing an effective demulsification method.[92]

It has been demonstrated that the contact angle of ferrofluids is altered significantly by the strength of the surrounding magnetic field.[74] The shapes of the drops themselves are strongly affected as well,[56] and magnetic fields can be used to alter the shape of liquid marbles formed with magnetic nanoparticles.[74] By making nano-structured surfaces infused with magnetic particles, changes in contact angle of regular (i.e. water) fluids can be achieved by altering the properties of the substrate.[153] However, surface instabilities produced by an external magnetic field make classical interfacial/surface tension measurements difficult.[56] Methods to calculate the surface and interfacial tensions of ferrofluids from the magnetic field peaking instability[5,56] and the deformation of a drop in a magnetic field[56] have been proposed, although they offer few insights into the effect of a magnetic field itself on the magnitude of the interfacial tension. It is interesting to note that these shape techniques, like regular drop shape techniques, fail when the magnetic Bond number (the ratio of magnetic forces over capillary forces, analogous to the regular Bond number describing the ratio between gravitational and capillary forces) is too low. Importantly, Flament et al.[56] did not find the surface tension to be affected by the field strength, although there are reports that the surface tension of water is increased approximately 1.5 mN/m by a strong magnetic field.[57] Some controversy still remains, as the measurement of interfacial tension is extremely sensitive to impurities.[6] Undeniably, however, magnetic nanoparticles can be used to manipulate a fluid drop, as the particles themselves can be moved by a magnet, producing a force acting on the drop wall that can roll the drop along.[74]

11.1.2 Metal complexes ($FeCl_3/FeCl_4^-$)

As was discussed in SECTION 2.3 (pg. 18), the effects of inorganic impurities on interfacial tension is particularly difficult to quantify. Iron chloride dissolves in water to form the complex tetrachloroferrate ion. Exchanging the counter ion from chlorine to tetrachloroferrate has been shown to give certain polymers magnetic properties,[105] and iron complexes formed the basis of many of the early magnetic surfactants discussed below. However, tetrachloroferrate is not particularly surface active, and has only a minimal effect on the surface tension of water. Its lack of surface affinity notwithstanding, a significant change in the deformation of a pendant drop of an aqueous solution of iron (III) chloride was noted in the presence of a magnetic field.[28] Given that the salt is not surface active, this argues that ions in the bulk play a significant role in magnetic phenomena.

11.1.3 Metallo-coordinated surfactants

In contrast to nanoparticles, magneto-responsive surfactants consist typically of a charged organic surfactant with a metal or metal-complex counter-ion at the head.[29–31,110] Ionic liquids using paramagnetic transition metal complexes were an important early step in the creation of ionic liquids that remained liquid at room temperature (Magnetic Ionic Liquids, or "MILS"). The separation between the metal centers, typically in excess of 6 Å, was thought to preclude magnetic coupling between the metal centers.[59] However, although fluid, these ionic liquids show simple paramagnetic behaviour over a wide temperature range (50–350 K),[49] and 3D ordering at low temperatures (4 K) has been observed.[59,111] Furthermore, it has been widely reported that the surface tension of ionic liquids containing transition metal complexes can be affected by an external magnetic field.[97] Many ionic liquids are extremely hydrophobic and difficult to dissolve in water.[49] There is considerable interest in adding the same type of magnetic responses to simple surfactants, which are soluble and surface active, thus tending to concentrate at interfaces, and which may provide useful functionality with far smaller bulk concentrations.

Like regular surfactants, magneto-surfactants are surface active and lower interfacial tension. Being paramagnetic, they are more effective at reducing the surface tension of water than their non-magnetic analogues, even without the influence of an external magnetic field.[29–31,110] Unlike ferrofluids, the magnetic response of these surfactants is linked to electronic and molecular spin and thus is related to their self-assembly and aggregation phenomena.[88] There is some speculation that unpaired electrons may align to the magnetic field, or that ion partitioning may occur across the fluid interface, accounting for their strong effect on surface tension.[29–31,110] Brown found that the reduction is increased further in the presence of a magnetic field. In other words, magneto-surfactants are bifunctional, with both chemical (adsorption) and mechanical (interaction with a magnetic field) effects on the solution.[30] However, measurements using the pendant drop technique require deformation due to the magnetic field to be considered as part of the Young-Laplace equation,[154] something that was not done in this instance, and the authors note that the estimates obtained are qualitative only.

Early magneto-surfactants were synthesised with iron counter-ions. More recent research has also explored the use of f-block metals for their particularly high magnetic moment, as well as the luminescent and catalytic properties associated with the lanthanide series.[31] The magnetic surfactants are based on normal cationic surfactants and produced by simple counter ion exchange. A mangeto-surfactant CTAF (cetyltrimethylammonium bromotrichloroferrate, or CTAB with $FeCl_3$) has been used as a structure directing agent mesoporous silica[88] and the DTAB equivalent (dodecyltrimethylammonium bromotrichloroferrate) has been shown to be a stronger antimicrobial agent than regular DTAB.[45]

11.2 Interfacial tension measurements

11.2.1 Silicone oil-water interface in the presence of $FeCl_{3(aq)}$

$FeCl_3$ was purchased from Sigma Aldrich and used as received. The black solid was made up to 0.112 g in 100 mL of water, just above the saturation limit, and dissolved with sonication. Excess solid was allowed to settle. 1000 CPS silicone oil (polydimethylsiloxane) was purchased from Brookfield and used as received. Measurements were undertaken in a square cell with 3 cm sides using a 9.53 mm sphere. The cell was allowed to rest for 20 minutes to come to equilibrium, and the equilibrium interfacial tension measured. A pair of rare earth magnets were then attached on either side of the cell (neodymium ring magnets from Lodestone Industries, 0.46 T), as shown in FIGURE 11.1.

The interfacial tension was monitored for 45 minutes after the magnets were applied, as shown in FIGURE 11.2. A gradual reduction in the apparent interfacial tension was observed over that period, reducing by 25% (to 25.6 mN/m after 45 min, down from 34.2 mN/m, Δ = 8.55 mN/m). The dynamic effect can be reasonably attributed to the high viscosity of the oil, and hence slow deformation of the interface. In addition, if iron complexes are being pulled towards the magnets, consequently being drawn from the bulk phase toward the interface, the strong hydrogen bonding network responsible for the high surface tension of water may be disrupted, resulting in a lower interfacial tension. As such a migration would be diffusion controlled, this may in part account for the observed dynamic effect. As shown in FIGURE 11.2, the interfacial tension measured with $FeCl_3$ at equilibrium, without a magnet, was approximately 3.5 mN/m lower than measured for silicone oil and water alone (see CHAPTER 8).

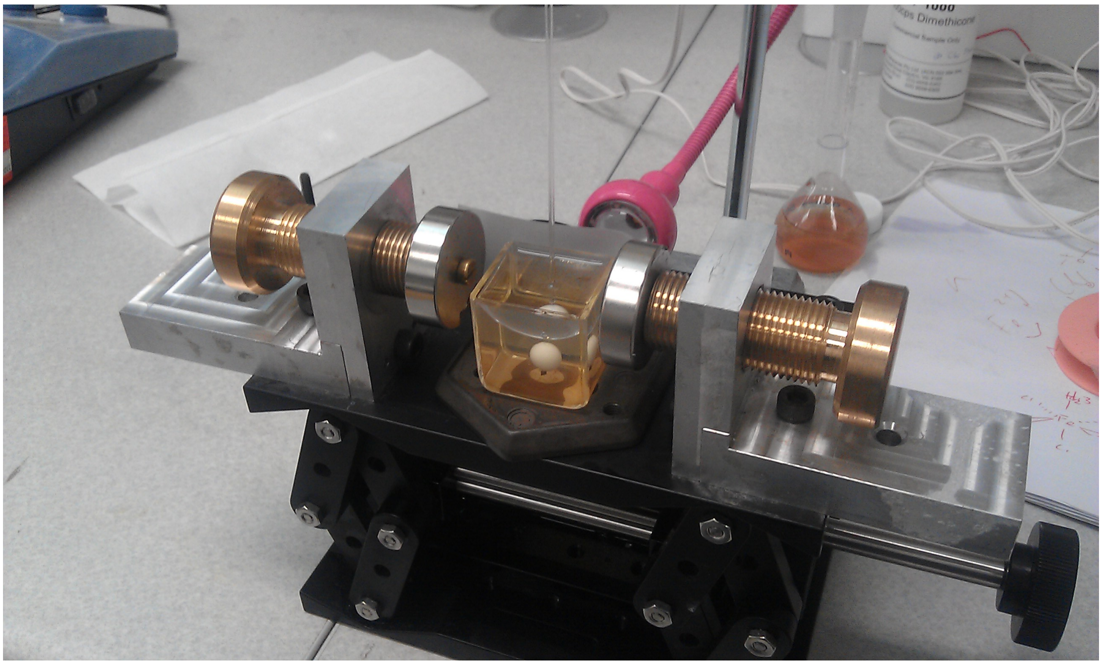

Figure 11.1: The cell used for interfacial tension measurements in the presence of a magnetic field. *Shown here:* measurement of silicone oil-water+$FeCl_3$ with a submerged holm.

11.2.2 Decane-water interface with magneto-surfactants (DTAB-Gd)

Trichlorides of transition metals with intrinsic magnetic susceptibility ($FeCl_3$ and $GdCl_3$) were added to common cationic surfactants dodecyltrimethylammonium bromide (DTAB) and cetytrimethylammonium bromide (CTAB) with the aim of creating magnetic cationic surfactants.

Cetyl trimethyl ammonium bromide (CTAB) and dodecyltrimethlammonium bromide (DTAB) were purchased from TCI Chemicals and used as received. Trichloride salts ($FeCl_{3(an.)}$ and $GdCl_3 \cdot 6H_2O$) and decane (99%) were used as received. The substituted surfactants were synthesized by dissolving equal molar portions of the trihalide metal and surfactant in methanol and mixing together overnight at room temperature. Excess solvent was removed by rotary evaporation and the solid dried overnight under vacuum at 30 ^{o}C. No further purification steps were attempted.

Evidence of successful counter ion exchange was found in terms of the melting point, which was found to change significantly. For surfactants based on CTAB and DTAB, substitution with $GdCl_3$ substantially raised the melting point, and substitution with $FeCl_3$ did the reverse. The melting points of CTAF and DTAF were consistent with reported values.[45,88] Further information on the characterisation of the surfactants is included in APPENDIX A.

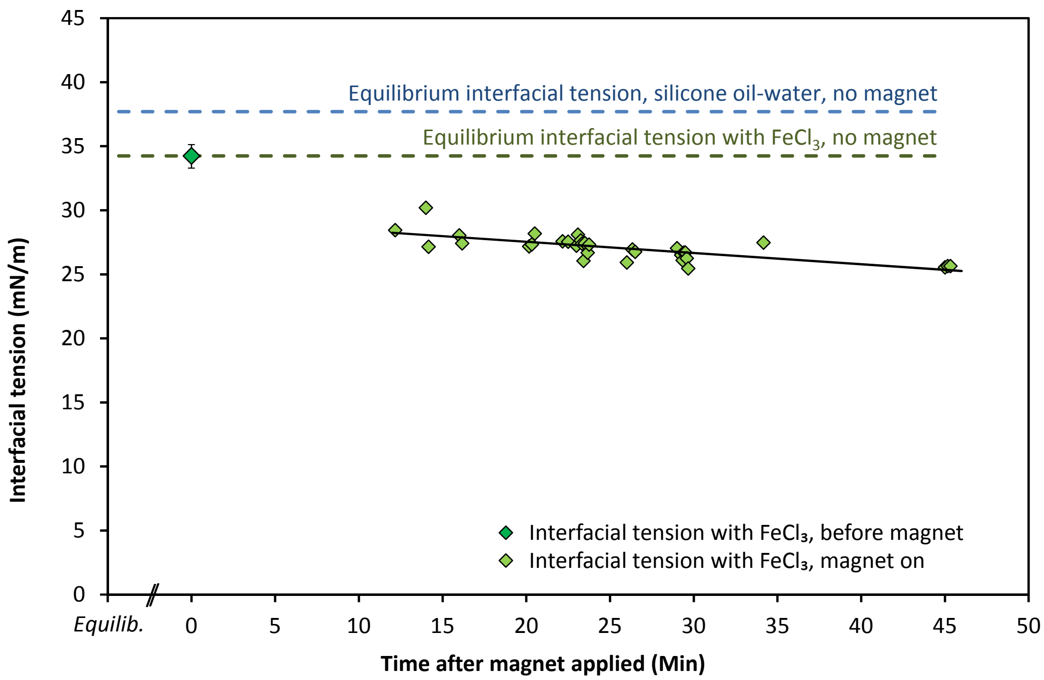

Figure 11.2: The change in interfacial tension of the silicone oil-water+$FeCl_3$ system under the effect of a magnetic field.

Surfactant solutions (0.1 M) were made up in water and the interfacial tension measured against decane. The holm meridian was formed with a 5.55 mm Teflon sphere in a small quartz cell. The cell was of rectangular cross section, 2 cm×4 cm. The cell was placed between two rare earth magnets (neodymium ring magnets from Lodestone Industries, 0.46 T) held 8 cm apart.

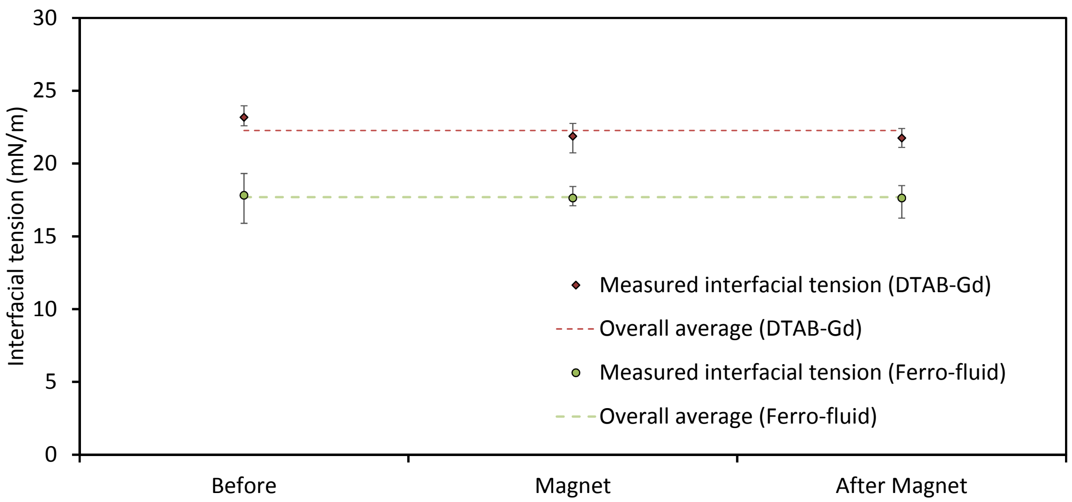

Figure 11.3: The changes in interfacial tension of the decane-water system with (•) 1 mM DTAB-Gd and (•) dilute ferro fluid, measured between two rare-earth magnets spaced 8 cm apart.

The effect of the magnets on the interfacial tension of DTAB is shown in FIGURE 11.3. While a small reduction was measured, the interfacial tension did not return to the original value, although a reversible change was anticipated.

11.2.3 Decane-water interface in the presence of a water-soluble ferrofluid.

A sample of a ferro fluid (EMG-707 – consisting of magnetic nanoparticles dispersed in water using an anionic surfactant) was purchased from Ferro Tec and used diluted at 1.5 mL in 98.5 mL of water.

The holm was formed with decane around a 6.35 mm ceramic (hydrophilic) sphere. Due to the intense colour of the aqueous layer, it was not possible to see the submerged holm formed around the Teflon sphere. (This issue was discussed in SECTION 5.7.1 (pg. 63).) By using the ceramic ball, it was possible to create a raised holm, which could be clearly seen in photographs. The results are shown in FIGURE 11.3. No change in interfacial tension was observed at this dilution.

11.3 Discussion

11.3.1 Significance of surface/interfacial tension changes

Brown et al.[28,30] reported changes in the surface tension of all surfactants in the study, both magnetic and otherwise, in the presence of an external magnetic field. The surface tensions with the magnetic surfactants was found to decrease slightly, and the surface tensions of the non-magnetic analogues were found to increase slightly. However, the significance of these changes must be considered.

While surface tension can be accurately measured, it is very susceptible to experimental conditions. Changes to the surface tension of the non-magnetic analogue of the order of 1~1.5 mN/m are quite small, and potentially within measurement error. There remains some controversy over the effect of a magnetic field on pure water. Measurements using a very strong magnetic field (of the order of 10 T) highlighted changes in the surface tension of water of the order of 1.5 mN/m,[57] more than 10 times stronger than the magnets used in this study. Amiri and Dadkhah[6] found the experimental conditions to be the most significant factor in trials on the magnetic effects on surface tension.

11.3.2 Physical effects of a magnetic field on responsive drops

The images presented by Brown et al.[30] leave no doubt that the drops containing magnetic surfactants are affected by the close proximity of the magnet, with deformation of the droplets clearly visible. However, as the measurement of interfacial tension was undertaken using the pendant drop method, this in itself is problematic, as the magnetic field produces a force which introduces additional deformation in the drop.[141] By consequence, the usual Young-Laplace equation is no longer strictly valid,[154] as noted by the authors themselves.[28] In a like vein, a version of ADSA was produced to fit sessile drops subjected to an external electric field,[14] and the effect of an external magnetic field on ferrofluids has been estimated using a version of the Young-Laplace equation modified by Maxwell's equation.[28,123] In the reported work, the alignment of the magnet is such that the elongation in the direction of the field induced by the magnets[154] will manifest as a lowered surface tension in the pendant drop fitting, hence it is not clear whether an apparent change in surface tension of 2 to 5 mN/m is truly significant. It is unlikely that the holm meridian will see the same type or extent of positive feedback as the pendant drop method, due to its shape, and thus it is unsurprising that the changes measured using the holm technique were quite small.

11.3.3 Magnet location, and the strength and direction of the magnetic field.

The location of the magnet and its distance from the fluids will affect the strength of the field experienced by the interface. The reported experiments detail a magnet held 1 mm from the base of the drop. Constraints due to the cell size necessitated a much larger separation between the magnet and the interface, hence effects from the magnets on the surfactants are anticipated to be weaker. However, attempts to measure the interfacial tension in a small cuvette (1 cm sides) resulted in significant scattering and a good fit was not obtained. A longer interface was required to obtain an accurate fit.

The direction of the magnetic field is also important, as the field results in deformation of the drop. A magnet placed beneath a pendant drop results in vertical deformation, and the drop remains axisymmetric. However, the change in deformation must be accounted for in the Young-Laplace equation used for fitting. In the experiments detailed here, the magnets were positioned on either side of the cell (as shown in FIGURE 11.1) and the field will run perpendicular to the axis of symmetry. Consequently, it is reasonable to assume that the magnets upset the axial symmetry of the holm. As can be seen from FIGURE 11.2, significantly more scatter was observed when measuring in the magnetic field (as compared to the equilibrium measurement). As the left and right sides of the holm are fitted separately, uneven deformation may well impact the measurement. However, the consistent downwards trend observed in the silicone oil/water/$FeCl_3$ measurement argues that a dynamic effect does exist.

11.4 Chapter summary

A steady decrease in the interfacial tension of the silicone oil-water+$FeCl_3$ system was observed in the presence of a magnetic field, reducing significantly (~8.5 mN/m) from the measured interfacial tension before the magnets were applied. This is consistent with changes observed for the surface tension of $FeCl_3$ in the presence of a magnetic field.[28] A small reduction in interfacial tension was measured with the magnetic surfactants using the holm technique, however the interfacial tension did not return to the equilibrium value once the magnets were removed. As the elongation of the droplet in the reported experiments will manifest as a reduction in surface tension, the changes reported by Brown et al.[30] is likely in excess of the actual change. The larger cell sizes used to obtain the holm interface separates the magnet from interface and weakens the observed magnetic field. In addition, aligning the field perpendicular to the axis of symmetry may result in uneven distortion of the interface and hence a poorer fit, which may account for the increased scatter when measuring with the magnet in place. In light of these difficulties, the holm technique would not be the most suitable choice for further investigations in this field.

Chapter 12

Concluding remarks

12.1 Method overview

The interfacial tension is a measure of the surface excess energy at the interface that describes how much work must be done to expand the interfacial area, and one of the most important factors for monitoring interfacial phenomena. This thesis adapts the basic principals underlying drop shape techniques and applies them to the submerged holm meniscus for the measurement of interfacial tension. To the best of the author's knowledge, the holm meridian has not previously been used for the analysis of interfacial tension, making this a substantial and original contribution to knowledge.

One of the key differences separating the holm meridian from pendant and sessile drops is the presence of a fixed solid object. By forming the meniscus around a solid sphere, the technique is made robust to vibration and other movement. The extent of curvature of the meniscus can be controlled by altering the penetration depth and chemical affinity of the sphere. The holm meridian facilitates measurement of liquid-liquid interfacial tension in systems with some sort of vibration or other movement, including strong convection currents induced by temperature gradients in the sample. The interface can be held stable for long periods of time and is only minimally affected by evaporation if a high vapour pressure fluid is used as the top phase. These factors make it an ideal method for long-term analysis.

12.2 Key objectives of the holm technique

PART III of this thesis presented its use in experiments which highlighted certain key objectives of the technique:

- maintaining a stable interface in a sample during external vibration and internal movement due to convection currents or stirring,
- an interface that can be held stable over a period of hours or days, and
- a system that can be completely enclosed during measurement.

The holm technique works well for measuring interfacial tension. The interface is stable and the solid object is easily affixed, making the interface robust against external vibrations. In one instance, the cell was heated to boiling point, which of course disturbed the two-phase bulk solutions. However, the holm interface was found to reform with no external interference once the solutions had cooled. Convection currents are potentially a greater problem, depending on lighting, as they can induce rapidly moving reflections in the upper fluid. This issue is effectively minimised with appropriate edge detection settings.

Secondly, the ability to maintain the same interface over a period of hours or days is of great importance in dynamic studies of surface-active chemicals with long equilibrium times. The adsorption of large proteins is an example of this. Maintaining a pendant drop over a period of hours or days can be difficult, and the experiment is wasted if the drop should fall, as a new drop will result in a fresh interface. As part of our studies on the effect of microwaves, the basic experiment was run over a period of two to six hours, and long-term measurements were run on the same interface over several days. The ability to maintain a single interface over hour-long experiments was a crucial aspect in analysing the effect of pH on fatty acid systems.

Lastly, the ability to completely enclose the system is an important attribute geared towards industrial application. The ambient temperature and pressure conditions of a typical laboratory do not reflect the high temperature high pressure systems found in many industrial settings, where knowledge of the interfacial tension could provide useful information. Pressurising cells for measurement with pendant or sessile drops requires pressurised syringes *etc.* for pumping and maintaining the drops, and apparatus for force-based measurements are generally too bulky to be enclosed. In contrast, the holm can be enclosed easily, making it a good candidate for pressurization, and is robust a against the convection currents produced when heating samples. These points make it an excellent candidate for measuring at high temperature and pressure.

12.3 Key differences between the holm technique and existing methods

There are a few key physical differences between the holm technique and existing alternatives:

- The quantity of liquid used by the holm method is significantly larger than other shape-fitting methods, although different sized cells can be used. In contrast, pendant and sessile drop methods require only a single drop of fluid inside a bulk fluid. Sample volumes are comparable to what is needed for the plate and ring techniques.
- The experimental setup for liquid-liquid measurement is significantly easier using the holm method as the requirement to form drops (ADSA) or align a plate or ring at the surface (Wilhelmy/du Nuöy methods) is eliminated.
- The interfacial area of the holm is significantly larger than that of drops.
- In comparison to force-based methods, once the sphere is set up, no direct interaction with the sample is require for measurement.

There are, of course, limitations to using the holm method. The various measurement methods for interfacial tension were each developed to suit a particular niche in the experimental landscape and thus have particular scenarios to which each is best suited.

All of the drop-shape techniques, being based on optical data (photographs), are fundamentally susceptible to issues with image analysis, optical distortion and lighting. And of course, each of these methods require an optical line of sight to the cell to allow photography. In contrast, force methods measure the interfacial tension through direct contact with the sample. However, the techniques are incredibly sensitive to non-uniformity in the apparatus, such as a bent wire causing the Wilhelmy plate to hang slightly askew, and the accuracy of the force measurement.

Dynamic bubble tensiometers focus mainly on dynamic measurements over very short time periods, and are totally unsuited to slow equilibrium times measuring hours or days. The spinning bubble tensiometer is particularly suited to the measurement of samples with low interfacial tension. The holm method is suitable to dynamic measurements over long periods of time and is also suited to measuring systems with low Bond numbers.

The sheer size of the apparatus and the multitude of moving parts surrounding force measurements such as the Wilhelmy plate method ensures that their integration into industrial settings is highly unlikely. Likewise, high precision measurements using pendant drops typically involves at least a pump, among various other apparatus to improve the repeatability/reliability of the experimental component. Measurements are taken on vibration-proof surfaces to minimise movement of the drop – movement which causing blurring of the images, rendering them unusable. Furthermore, the available software is often unwieldy and quite difficult to use.

The experimental technique developed in this thesis was intentionally pared down as much as possible, with the equipment reduced to a glass cell and a small ball with some way to tether the ball. With no need for pumps or even a syringe, it was quite possible to contain the entire system within the glass cell, allowing it to be totally enclosed for certain measurements. The larger cell and bulk fluid interface allows probes – for temperature, conductivity, pH – to be inserted, providing additional information about the bulk fluid. The down side is that the holm method requires significantly more sample than a single drop, although sample is not wasted being squirted out of a syringe, and the same sample is used throughout the experiment. Changing the cell size can also adjust the sample requirements to some extent.

The computational analysis of the holm method is longer than the pendant drop method. However, as the sphere and hence interfacial location is fixed, it is generally possible to provide the required manual inputs at the beginning of a long analysis and those inputs can be used for the remaining images. Edge detection is critical in all drop techniques, and the larger interfacial length and more complicated reflections makes lighting a critical consideration in the experimental setup, arguably more so that for pendant or sessile drops.

12.4 Novelty

The niche that the holm method fills is squarely focused on the measurement of liquid-liquid interfacial tension and long measurement times. It is uniquely resistant to movement in the sample and provides measurement of the bulk solution without direct contact during the experiment, and is suited to measurement under chemically or physically dynamic conditions. The rate of analysis is limited solely by the rate at which images can be gathered, making it useful for measurements of dynamic interfacial tension. As the cell can be completely enclosed, the technique is a stepping stone towards interfacial tension measurements that can mimic industrial conditions.

APPENDIX A

Synthesis of magnetic surfactants

A.1 Synthesis

The substituted surfactants were synthesized by dissolving equal molar portions of trihalide metals ($FeCl_3$, $GdCl_3$) and surfactant (CTAB, DTAB) in methanol and mixing together overnight at room temperature. Excess solvent was removed by rotary evaporation and the solid dried overnight under vacuum at 30 ^{o}C. No further purification steps were attempted.

Henceforth, CTAF/DTAF and CTAG/DTAG will refer to cetyltrimethylammonium bromide (CTAB) and dodecyltrimethylammonium bromide (DTAB) where the counterions have been exchanged to $FeCl_3Br^-$ and $GdCl_3Br^-$, respectively.

A.2 Characterisation

A.2.1 Melting point

The clearest evidence for successful counter-ion substitution from Br^- to $[MCl_3Br]^-$ is the dramatic change in melting point of the solid surfactants. Iron chloride reduced the melting point significantly, while addition of gadolinium chloride increased the melting point above (300 °C). The two gadolinium samples were noted to discolour at higher temperatures and may decompose rather than melt. Melting point data is given in TABLE A.2.

Table A.1: Melting point data

Counterion	CTAB	DTAB
Br^-	237-243 °C (decomp.)	246 °C
$FeCl_3Br^-$	60-62 °C (lit. 64 °C) [88]	28-29 °C (lit. 32 °C [45])
$GdCl_3Br^-$	< 297 °C (discoloured)	< 297 °C (discoloured)

A.2.2 Solubility

The substituted surfactants and their parent surfactants displayed the same solubility trends with the solvents tested, as shown in TABLE A.2.

Table A.2: Solubility trends (all surfactants displayed the same general trends)

Solvent	CTAB/DTAB	After substitution
Water	O	O
Methanol	O	O
THF	O	O
Chloroform	O	O
Hexane	X	X

A.2.3 FTIR

The characteristic IR peaks of CTAB and DTAB are near-identical.[61] Characteristic peaks relating to iron are only observed in the fingerprint region, which could not be measured.

A.2.4 UV-Vis spectroscopy

UV-VIS measurements over the range 200-700 nm in a methanol matrix confirmed the presence of iron (III) in the appropriate samples. $GdCl_3$, CTAB and DTAB did not display any characteristic absorbance.

A change in the characteristic two peak absorbance from Fe(III) (from $FeCl_3$) to three peaks in the tetrachloride form $[FeCl_4]^-$ when measured in THF has been reported.[2] The characteristic three peaks were observed in a 0.1 mM solution of CTAF in THF (not stabilised). From this and the drastic change in melting point between the samples it is reasonable to conclude that the counter-ion was changed successfully from Br^- to $[FeCl_3Br]^-$.

APPENDIX B

Coding

B.1 An overview of the coding structure

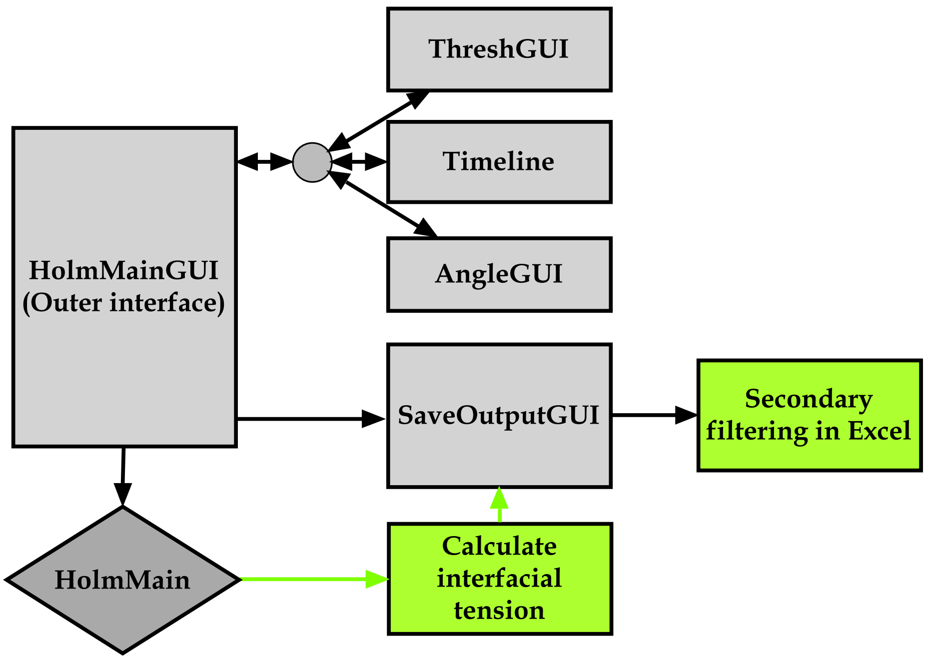

Figure B.1: Main program GUI.

Access to all of the GUIs is through `HolmMainGUI`.

B.2 Code included in this abridged appendix

As the coding is extensive, only that pertaining to the back-end analysis is included in this abridged appendix. For a complete copy of the code, please see the extended appendix (200 pg.) on the attached CD-rom.

B.3 Holm main GUI (controller GUI)

The full code for `HolmMainGUI` is included in the extended appendices.

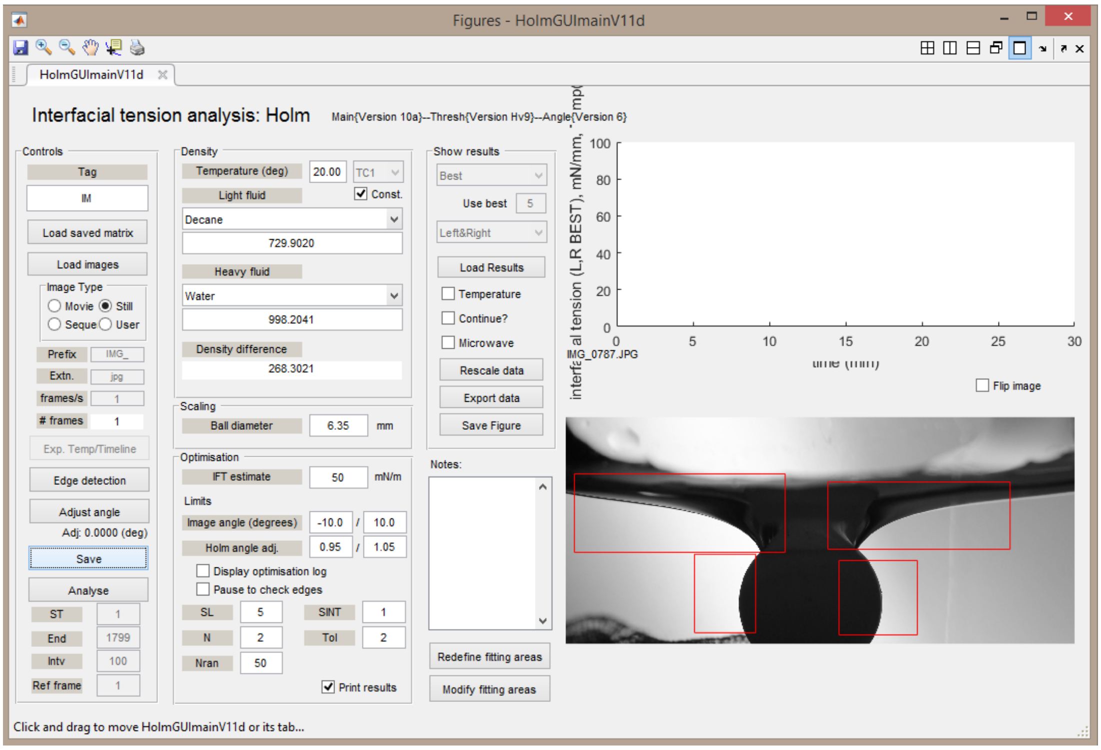

Figure B.2: Main program GUI.

B.4 ThreshGUI (interactive thresholding and edge identification)

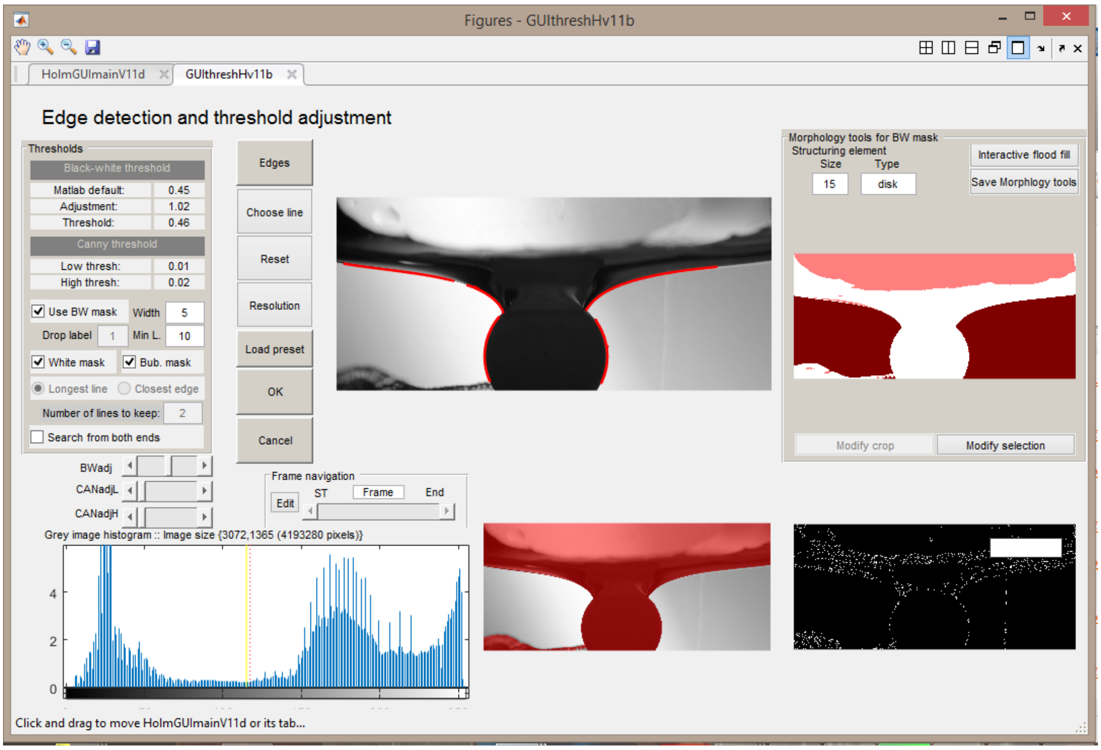

Figure B.3: Interactive thresholding GUI.

The full code for `ThreshGUI` is included in the extended appendices.

B.5 AngleGUI (automated angle adjustment)

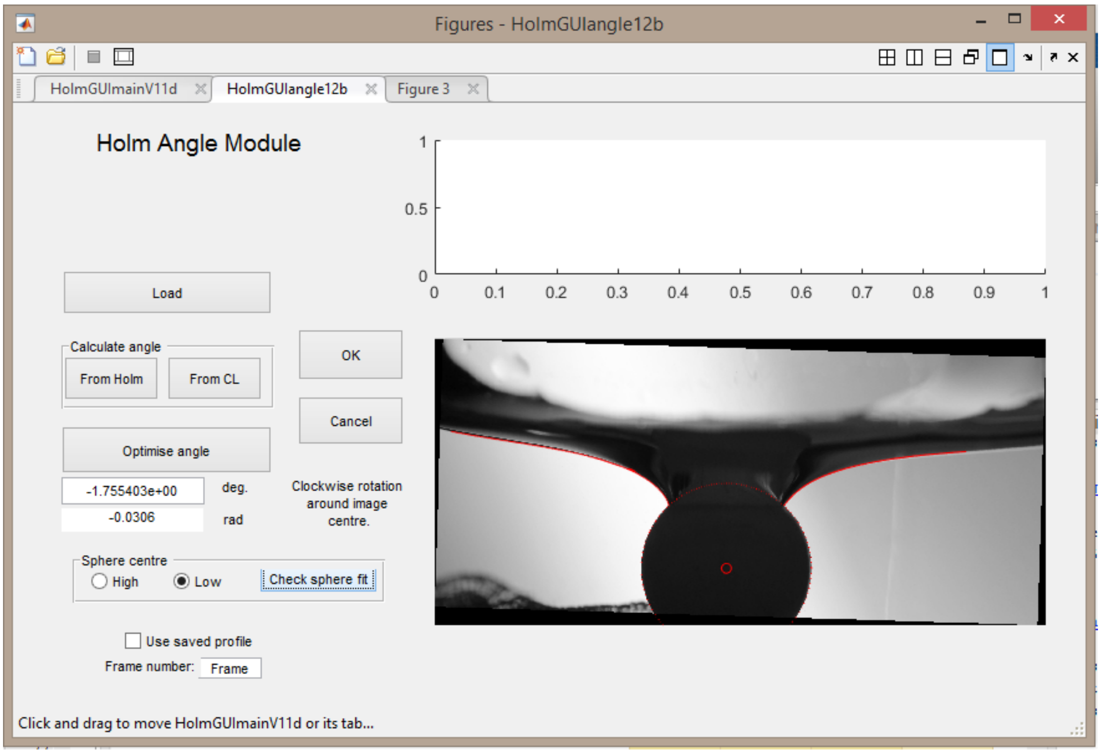

Figure B.4: Angle adjustment GUI.

The full code for `AngleGUI` is included in the extended appendices.

B.6 Timeline (formatting of temperature files)

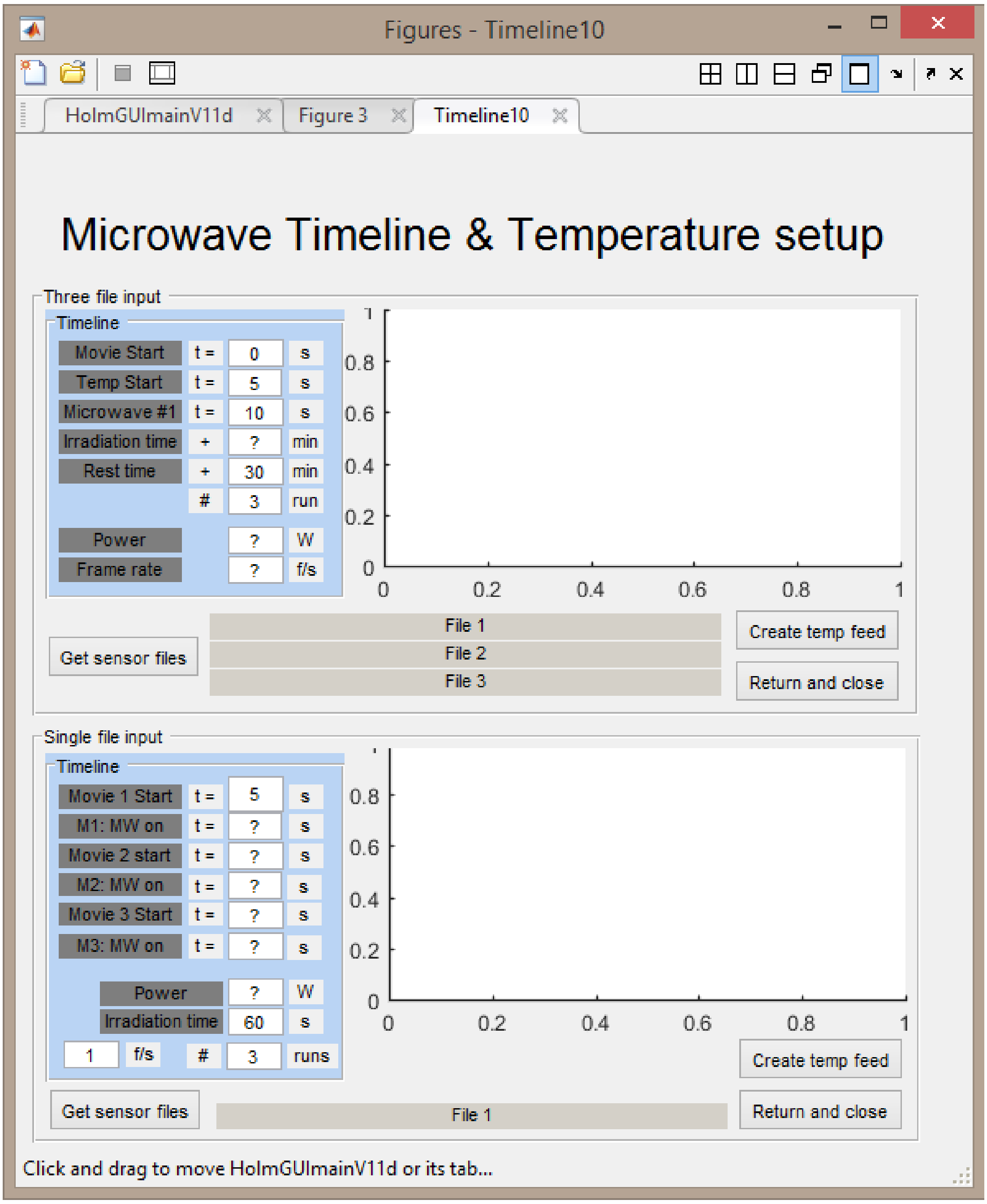

Figure B.5: Formatting for temperature files.

The full code for `Timeline` is included in the extended appendices.

B.7 SaveOutputGUI (export output .txt file to Excel template)

The full code for SaveOutputGUI is included in the extended appendices.

B.8 Excel template

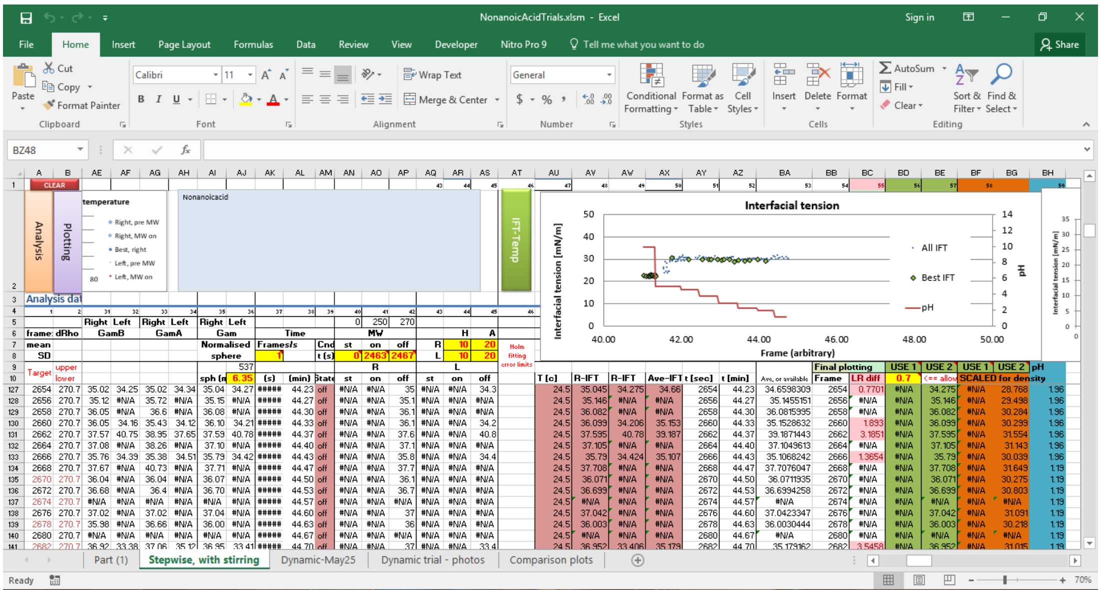

Figure B.6: Template for Excel files.

VBA code used with the template is included in the extended appendices.

B.9 Holm_Main (code for back-end analysis)

Listing B.1: Holm_MAIN_3.m

```
%-----------------------------------------------------------------------%
%DETERMINATION OF FLUID INTERFACE PROPERTIES THROUGH IMAGE ANALYSIS
%-----------------------------------------------------------------------%
%% Version-up info
% New edge detecton module to handle broken edges and reflection.

```

```
%% INFORMATION
%Matlab code to calculate fluid−fluid surface tension Submerged holm
%Can be used repetitively with movies (use MovieHolmX to read frames in ).

%CALLS EXTERNAL FUNCTIONS:
%
%−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−%
%Written for PhD(ChemEng), 2014−2018
%Anita Hyde, 14291160
%Supervisor: Chi Phan
%Written in MATLAB R2012.  Requires Image Analysis Toolbox and Curve Fitting Toolbox.
%Migrated to Matlab 2015b in 2016 (major change to graphics system introduced in 2014b).

%% START
function [Gamma,aBF,fval,OptStore]=Holm_MAIN_3(Side,ImCase,ImName,PrintYN,x0,y0,R,HolmCoord,Grey,folder,fig,FDisp,Data)
    %% DECLARATIONS
    %ImCase= num %IDENTIFY DATABASE ENTRY
    %askBox = use getrect function to unput holm and sphere location
    %Xu, Yu: points to estimate edge location
    %Grey − Greyscale image.  For movies.
    %SL=5; %number of points to search
    %N=5; %number of reruns at each starting point (different set of random
    %coordinates)

    %−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
    %READ IMAGE DATA
    %−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
    SL=Data.SL;
    N=Data.N;
    nran=Data.nRan;
    Tol=Data.Tol;
    g=9.81;
```

```
    GamEst=Data.GamEst;
    Size=Data.Size;
    dRho=Data.dRho;
    %ImName=Data.ImName;
    SphCnr=Data.SphCnr;
    HlmCnr=Data.HlmCnr;

    %----------------------------------------------------------------
    %SAVE DATA - FILES - PrintYN=Yes
    %----------------------------------------------------------------
    if PrintYN==true
        %files
        Result_file=sprintf('%s-O-%s-%i.txt',folder,Side,ImCase);
        %fullFileNameA = fullfile(folder, Result_file);
        fileID=fopen(Result_file,'a');%use a+ for reading & writing, append
        %--abbrv file header
        fprintf(fileID,'%s: results for file, %s\r\n',datestr(now),ImName);
        %===============CLEANUP================
        finishup = onCleanup(@() myCleanupFun(fileID));
        %=====================================
        %fprintf(fileID,'[   a     |  hlim   | theta(rad)| fvalh  | Coord num]\r\n',datestr(now),ImName);
    else
        fileID=1;%print to screen
    end
    %----------------------------------------------------------------
    %ANALYSIS
    %----------------------------------------------------------------


%FITTING (2 pass)
axes(fig),
%--Estimate shape factor
a0=sqrt(dRho*g/(GamEst/1000));%gamma in mN/m -> a (m).  See {Huh, Chun and Scriven, L.E. 1969}
```

```
Scale=Size/2/R;%image scale mm/p. 'Size'=sphere diameter in mm
a0=a0/1000*Scale;%Scale = mm/pixeRl
fprintf(1,'starting "a": %f\r\n',a0);
%--fit polynomial to initial segment to calculte tangent
[poly2,MonSrt,PolyCoord]=FitPoly(x0,y0,R,Tol,int32(70),SL,HolmCoord);%(x0,y0,R,Tol,PL,HolmCoord). TOl is sum(diff)

if PrintYN==1
    %print edge information
    fprintf(fileID,'Search area: sphere x1,y1,x4,y4; coordinates x/y \r\n');
    fprintf(fileID,'% 5i',SphCnr);
    fprintf(fileID,'\r\n Sphere profile: x0: %f, y0: %f, R: %f \r\n',x0,y0,R);
    fprintf(fileID,'\r\n \r\n');
    fprintf(fileID,'Search area: holm x1,y1,x4,y4; coordinates x,y \r\n');
    fprintf(fileID,'\r\n');
    fprintf(fileID,'% 5i',HolmCoord(1,:));
    fprintf(fileID,'\r\n');
    fprintf(fileID,'% 5i',HolmCoord(2,:));
    fprintf(fileID,'\r\n \r\n Polynomial (Poly2): % 10f, % 10f, % 10f \r\n',poly2);
    fprintf(fileID,'\r\n \r\n Additional information: Holm starting coordinate:% 3i; SL:% 3i; N:% 3i; number of points (udist): %i\r\n \r\n',MonSrt,SL,N,nran);

    %labels
    fprintf(fileID,'%s\r\n','First past - searching for best initial point');
    fprintf(fileID,'%12s','a0','a','hlim','theta','fvalh','coord','fitting index');
end

%--Initialise matrices for storage
HOsize=1:SL;%Specify indices (in HolmCoord) of starting points to consider.
```

```
FitCoordX=zeros(length(HOsize),length(HolmCoord)+1);
FitCoordY=zeros(length(HOsize),length(HolmCoord)+1);
ODEX=zeros(length(HOsize),length(0:0.001:3*pi));
ODEY=zeros(length(HOsize),length(0:0.001:3*pi));
OptStore=zeros(length(HOsize),7);%Stores the optimium values found at
    each iteration.

hold all;
%OPTIMISATION
fprintf('Evaluating...');
HOsize=zeros(2,SL*N);
SL=double(SL);N=double(N);MonSrt=double(MonSrt);
for i=0:SL
    j=N*i+1;
    HOsize(1,j:j+N-1)=(i+1)*ones(1,N);
end
for i=1:3:SL*N
    HOsize(2,i:i+2)=[a0/2,a0,a0*2];
end

% fprintf(fileID,'\r\n Best starting points found: %i, %i \r\n', S)
% fprintf(fileID,'Fitting %i times using %i random points \r\n',N*p,
    nran)

for k=1:length(HOsize)
    CoordNum=HOsize(1,k);
    a=HOsize(2,k);
    fprintf('%d...',CoordNum)

    %--Select points for fitting (nran + starting point)
    pts=random('unid',length(HolmCoord)-MonSrt-SL,1,nran);%does not
        pick points from initial SL
    pts=sort(pts,2)+MonSrt+SL;%50 random points > starting point
%     pts=[CoordNum,pts];%starting point + 50 random points as index
    Coord=[PolyCoord(:,CoordNum:end),HolmCoord(:,pts)];
    %plot(Coord(1,:),Coord(2,:),'o','markersize',4);
```

```
    fvalS=1;%%TEMP
    %--FIT HOLM CURVE (3 VAR OPT)
    [HolmOpt,fvalH,Xxi,Yyi,X,Y]=holm(x0,y0,R,a,Coord,poly2);
    FitCoordX(k,1:length(Xxi)+1)=[length(Xxi),Xxi];
    FitCoordY(k,1:length(Yyi)+1)=[length(Yyi),Yyi];
    ODEX(k,1:length(X)+1)=[length(X);X];
    ODEY(k,1:length(Y)+1)=[length(Y);Y];
    OptStore(k,:)=[a,HolmOpt,fvalH,fvalS,CoordNum];%[a0,(a,hlim,theta(
        rad),fvalh),Coord num]

    %--print output to file
    if PrintYN==1
        fprintf(fileID,'\r\n');
        fprintf(fileID,'%4i, % 12.4f, % 12.4f, % 12.4f, % 12.4f, % 12.4
            f % 12i',k,HOsize(2,k),OptStore(k,:));
        fprintf(fileID,'% 5i','',pts);
        fprintf(fileID,'% 12.4f','',X,'',Y);
    end
    %plot(X,Y)
end%REPEAT at sucessive coordinates from start
    %Identify the minimum and mean error
    MinError=min(OptStore(:,5));
    %Keep entries less than X times the minimum error.
    S=OptStore(:,5)<2*MinError;%logical matrix of rows which 'pass'
    Store=zeros(sum(S),8);
    Store(:,1:7)=OptStore(S==1,:);%Rows which pass
    Store(:,9)=dRho*g*1000./(Store(:,2)./Scale*1000).^2;
    ODEX=ODEX(S==1,:); ODEY=ODEY(S==1,:);%limit ODEX/Y to match Store.

    [~,Best]=min(Store(:,5));
    if strcmp(Side,'Left')==1
        for i=1:sum(S)
            if PrintYN==1
                fprintf(fileID,'% 5i, % 12.6f, % 12.6f, % 12.4f, %
                    12.4f, % 12.4f, % 12i, % 12.4f, % 12.2f\r\n',i,
                    Store(i,:));
            end
```

```
                st=ODEX(i,1);
                Xrev=size(Grey,2)-ODEX(i,2:st);
                plot(Xrev,ODEY(i,2:st),'-','color',[0.5,0.5,0.5]);
                %figure(2), plot(ODEX(i,2:st),ODEY(i,2:st),'-y')%,'
                    color',[0.5,0.5,0.5]);
        end
    else %right
        for i=1:sum(S)
                if PrintYN==1
                    fprintf(fileID,'% 5i, % 12.4f, % 12.4f, % 12.4f, %
                        12.4f, % 12.4f, % 12i, % 12.4f, % 12.2f\r\n',i,
                        Store(i,:));
                end
                st=ODEX(i,1);
                plot(ODEX(i,2:st),ODEY(i,2:st),'-','color'
                    ,[0.5,0.5,0.5]);
        end
    end

%SCALING AND CALCULATIONS
fHave=mean(Store(:,5));%filtered, average fitting error
aBest=Store(Best,2);
fprintf('Best "a": %f',aBest);
GammaAveF=mean(Store(:,9));
Gamma=Store(Best,9);
fvalH=OptStore(Best,5);fvalS=OptStore(Best,6);
fval=[fvalH,fvalS,fHave];
aBF=[aBest,mean(Store(:,2))];
Gamma=[Gamma,GammaAveF];
stDev=std(Store(:,9));

%Show best result
    st=ODEX(Best,1);
    axes(fig), hold on
    if strcmp(Side,'Left')==1
        Xrev=size(Grey,2)-ODEX(Best,2:st);
```

```
            plot(Xrev,ODEY(Best,2:st),'-','color',[0.8,0.3,0.1],'linewidth',1.5);
            figure(FDisp), plot(Xrev,ODEY(Best,2:st),'-','color',[0.8,0.3,0.1],'linewidth',1.5);
            figname=sprintf('Edge-%d',ImCase);
            fullFileName = fullfile(folder, figname);
            saveas(FDisp, [fullFileName '.png']) %2nd side
            plot([1,1200],[Store(Best,3),Store(Best,3)],'y');
        else
            plot(ODEX(Best,2:st),ODEY(Best,2:st),'-','color',[0.8,0.3,0.1],'linewidth',1.5);
            figure(FDisp), plot(ODEX(Best,2:st),ODEY(Best,2:st),'-','color',[0.8,0.3,0.1],'linewidth',1.5);
            plot([1,1200],[Store(Best,3),Store(Best,3)],':y');
        end

    %=======================================
    %Results
    %=======================================
    Res=[Gamma(1);GammaAveF;MinError;stDev;sum(S)];
    Res=array2table(Res,'VariableNames',{Side},'RowNames',{'IFT-Best','IFT-Ave','MinError','stDev','n'});
    disp(Res)

end %end main function


%% FIT POLY
function [poly2,MonSrt,PolyCoord]=FitPoly(x0,y0,R,Tol,PL,SL,Coord)
%--------------------------------------------------------------------------
%FN FITPOLY: fits polynomial over short range at start (bubble end) of
%curve. Returns coefficients as matrix (polyfit format) to allow for
%tangent calculation. ALso return MonSrt - 1st coordinate.
%--------------------------------------------------------------------------
 %STARTING COORDINATES - based on deviation from sphere
%..Max sphere height (pixel position):
    Ym=y0-R;
```

```
    i=length(Coord);
    while Coord(2,i)<Ym
        i=i-1;
    end
    Ym=[Ym, i];
%..Half Sphere height
    j=i;
    while Coord(2,j)<y0 && j>2
        j=j-1;
    end

%      %re-order lower segment by x, not y.
%      CrdLow=sortrows(transpose(Coord(:,j:i)),2);
%      Coord(:,j:i)=transpose(flipud(CrdLow));

%..Search backwards
    Diff=10;
    while Diff>1&&i>2
        i=i-1;
        X=Coord(1,i);
        Y=Coord(2,i);
        x=sqrt(abs(R^2-(Y-y0)^2))+x0;%(x-X)^2+(y-Y)^2=R^2
        Diff=X-x;%not abs - do not want to start inside the circle
    end
    %plot(X,Y,'yo')
%..Search forwards
    Diff=0;
    while Diff<Tol
        i=i+1;
        X=Coord(1,i);
        Y=Coord(2,i);
        x=sqrt(abs(R^2-(Y-y0)^2))+x0;%(x-X)^2+(y-Y)^2=R^2
        diff=X-x;%not abs - do not want to start inside the circle
        if diff>1%Will not pick up negative values.
            Diff=Diff+diff;
        end
        %plot([X,x],[Y,Y],'y')
```

```
        PolStart=int32(i);
    end

    %plot(X,Y,'mo')
    MonSrt=PolStart;
    if PolStart<=2
        display(sprintf('Unable to determine starting point for
            polynomial. Default is first coordinate.  Please check
            interface between sphere and holm'));
    elseif PolStart<6
        PolStart=1;
    else
        PolStart=PolStart-5;
    end

    %temp for sharp angles
    PolStart=MonSrt;
    MonSrt=MonSrt+2;

  %FIT ploy/power curve to edge section.  Use X=fn(Y, quadratic)
    poly2=polyfit(Coord(2,PolStart:MonSrt+PL),Coord(1,PolStart:MonSrt+
        PL),2);
    PolyCoord=[polyval(poly2,Coord(2,MonSrt:MonSrt+SL));Coord(2,MonSrt:
        MonSrt+SL)];
  %PLOT (plots on right side)
    plot(polyval(poly2,Coord(2,PolStart:MonSrt+PL)),Coord(2,PolStart:
        MonSrt+PL),'b')
    plot(Coord(1,MonSrt),Coord(2,MonSrt),'go',Coord(1,MonSrt+PL),Coord
        (2,MonSrt+PL),'go')
    %plot(Coord(1,:),Coord(2,:),'r')

end
%------------------------------------------------------------------------%

%% HOLM
function [HolmOpt,fvalH,Xxi,Yyi,X,Y]=holm(x0,y0,R,a,Coord,poly2)
%------------------------------------------------------------------------%
```

```
%HOLM - outer function, calls solver
%----------------------------------------------------------------------%
 %COORD has intital coordinate + 50 random points for fitting
 %--estimate x-axis position
     Xxi=[];Yyi=[];X=[];Y=[];
     HLim=Coord(2,end);%--y=0 (lim(holm edge))-->average final 10 points
 %--Scaling
     ScaleR=[1,10,1];

 %--Initial conditions
     [Initial]=initialCoord(Coord,poly2);%
     Guess=[a/ScaleR(1),0.2*HLim/ScaleR(2),Initial(3)/ScaleR(3)];%a, x axis position. Scaled.

 %--Boundaries for fmincon
     LowerBound=[0.01*a/ScaleR(1),-100*HLim/ScaleR(2),0.9*Initial(3)/ScaleR(3)];
     UpperBound=[5*a/ScaleR(1),5*HLim/ScaleR(2),1.1*Initial(3)/ScaleR(3)];%NOTE: Y=0 at top of image

 %OPTIMISATION
     Opts = optimset('Display','none','Algorithm','active-set','LargeScale','on','MaxFunEval',1000);
     [HolmOpt,fvalH,~,output]= fmincon(@(Guess)HolmObj(Guess,x0,Coord,ScaleR,Initial),Guess,[],[],[],[],LowerBound,UpperBound,[],Opts);
         %% OBJECTIVE FUNCTION (HOLMOBJ - nested)
         function [eMin]=HolmObj(Guess,x0,Coord,Scale,Initial)
         %----------------------------------------------------------------------%
         %Objective function called by fmincon
         %----------------------------------------------------------------------%

           %--OPTIMISATION PARAMETERS
               a_=Guess(1)*Scale(1);%scaling/reducing factor
               HLim_=Guess(2)*Scale(2);%estimate of y(inf) value (asymptote)
               theta0=Guess(3)*Scale(3);
```

```
        %--imported variables
            X0=Initial(1);Y0=Initial(2);

        %----Convert to calculation space
            %Origin a x0,Hlim.  Regular axis direction (image, y axis
                reversed)
            %--y=0 at asymptote -> HLim
            Y0=-(Y0-HLim_);%image coordinates from top right corner
            %--x=0 (sphere centre)-->circle x0
            X0=X0-x0;%same direction. Always +ve.

        %--NUMERICAL INTEGRATION
            v0=[theta0,X0*a_,Y0*a_];
            Span=[0:0.01:3*pi];%why was is 20 Pi?
            [S,v] = ode45(@(S,v) holmODE(S,v),Span,v0);%integrate

        %--isolate area of interest from theoretical curve
            i=1;mono=length(v);
            while v(i,1)>1*pi/180&&i<length(v) %for theta > 10
                mono=i;
                i=i+1;
            end%determine theoretical curve before asymptote

        %--THEORETICAL CURVE (with asymptote)
            X = v(1:mono,2)/a_+x0;
            Y = -v(1:mono,3)/a_+HLim_;

            if isempty(X)==1||length(X)<=2
                eMin=2*10^10;%if solver returns only 1 point, large
                    error
                Xxi=[]; Yyi=[];
                disp('<<holm opt>>length(X)==0')
            else %determine error for
                [eMin,Xxi,Yyi]=errorX(X,Y,Coord);
            end
        end %x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)
```

```
    HolmOpt=[HolmOpt(1)*ScaleR(1),HolmOpt(2)*ScaleR(2),HolmOpt(3)*
        ScaleR(3)];%rescale for export
end

%% HOLM OPT INITIAL COORDINATES
function [Initial]=initialCoord(Coord,poly2)
%------------------------------------------------------------------
%Curve starting from first point (CN already adjusted)
%Estimate theta from polynomial at known X
%------------------------------------------------------------------
    %Starting coordinates
     X=Coord(1,1);
     Y=Coord(2,1);
    %calculate tangent&theta
     tangent=polyval(polyder(poly2),Y);%X=fn(Y^2)>>tangent=dX/dY
     theta=atan(-1/tangent);%(y coord reversed)
     if theta<0; theta=theta+pi;end %accounts for reversed holm
     Initial=[X,Y,theta];
     %y=m(x-x0)+y0
     %tang=tan(theta+pi);
     %X2=0.9*X; Y2=Y-tang*(X2-X);
     %plot([X,X2],[Y,Y2],'y','linewidth',2);
     %plot([X,X+100],[Y,Y],'y','linewidth',2);
end

%% ODE (REDUCED FORM - HOLM)
function [dvdS]=holmODE(S,v0)
%-------------------------------------------------------------------------
%Young-Laplace differential equations for ODE solver
%-------------------------------------------------------------------------
    %Import variables
     theta = v0(1);
    X = v0(2);
    Y = v0(3);

    % Differential equations
     if X~=0
```

```
        dthetadS = Y-sin(theta)/X;
    else
        dthetadS = Y/2;
    end
    dXdS = cos(theta);
    dYdS = sin(theta);

    % Pack derivatives for return
    dvdS = [dthetadS; dXdS; dYdS];
end

%% X-ERROR CALCULATION - HOLM
function [error,Xxi,Yyi]=errorX(X,Y,Coord)%X,Y theoretical, x,y
    detected
%-----------------------------------------------------------------------
%Error calculation between theoretical and detected profiles.
%-----------------------------------------------------------------------

%Make theoretical curve monotonic & determine inflection points for
    holm
    k=0;Mid=0;Theo=zeros(2,length(Y));
    for i=1:length(Y)-1
        if Y(i)~=Y(i+1)%&& X(i)<=X(1,end)%monotonic Y for range X =
            HolmCoord(1,:)
            k=k+1;
            Theo(1,k)=X(i);
            Theo(2,k)=Y(i);
        end
    end%Theoretical curve - coordinates at unique Y values only
    Theo=Theo(:,1:k);%remove empty elements
    if isempty(Theo)==1; disp('<<error1D>>Theo empty'); end
    endflag=false;Mid=0;
    for i=1:length(Theo)-1
        if Theo(1,i)>=Theo(1,i+1)&&endflag==false
            Mid=i;
        else endflag=true;
        end
```

```
    end

    if isempty(Coord)==1; disp('<<error1D>>Coord empty'); end
    MidH=0;CE=length(Coord);
    if Mid==0; %disp('<<error1D>>Mid==0');
        for i=1:length(Coord)
            if Theo(2,end)<Coord(2,i); CE=i;end
        end
    elseif Mid~=0
        endflag=false;
        for i=1:length(Coord)
            if Coord(2,i)>=Theo(2,Mid)&&endflag==false
                MidH=i;
            else endflag=true;
            end
            if Theo(2,end)>Coord(2,i); CE=i;end
        end
        %if MidH==0; disp('<<error1D>>MidH==0 (Coord) but Mid~=0 (theo) - lower section empty'); end
    else %disp('<<error1D>>Mid<0');
    end

%INTERPOLATE
  %split X,Y, HolmCoord into monotonic sections
    YyiU=Coord(2,MidH+1:CE);%Y limit at highest point of theo curve
    XxiL=[];XxiU=[];flagempty=true;
    if Mid==0;Mid=1;%not reversed holm - remove lower portion
    else%reversed holm - interpolate lower section
        YyiL=Coord(2,1:MidH);flagempty=false;
        if length(Theo(1,1:Mid))>1%interp requires two points
            XxiL=interp1(Theo(2,1:Mid),Theo(1,1:Mid),YyiL,'linear');% lower region
        else %disp('<<error1D>>no lower coordinates');
        end
    end
    if length(Theo(1,Mid+1:end))>1
```

```
        XxiU=interp1(Theo(2,Mid+1:end),Theo(1,Mid+1:end),YyiU,'linear');%upper region
    else disp('<<error1D>>no upper coordinates');
    end

    if flagempty==true; YyiCat=YyiU; XxiCat=XxiU;
    else XxiCat=[XxiL,XxiU];YyiCat=[YyiL,YyiU];%catonate
    end

%REMOVE NaN >> matlab will return NaN if asked to EXTRAPOLATE using interp1, linear
%& ADD ARTIFICIAL ASYMPTOTE (bias unfeasible solutions)
    Xxi=zeros(1,length(Coord));Yyi=zeros(1,length(Coord));
    Ymax=Coord(2,1);
    for i=1:length(Coord)
        if i>length(XxiCat)%asymptote
            Xxi(i)=Coord(1,i);
            Yyi(i)=Ymax;
        elseif isnan(XxiCat(i))==0%not NaN
            Xxi(i)=XxiCat(i);
            Yyi(i)=YyiCat(i); Ymax=YyiCat(i);
        else %is NaN >> out of range. Replace with Xcoord,max(Yyi) within range
            Xxi(i)=Coord(1,i);
            Yyi(i)=Ymax;
        end
    end

%DETERMINE ERROR
    if length(Xxi)<=1; error=10^7*length(Theo)^2*length(Coord)^2;disp('<<error1D>>length(Xxi)<=1');
    else
        Error=zeros(1,length(Coord));
        for i=1:length(Error)
            Error(i)=(Xxi(i)-Coord(1,i))^2+(Yyi(i)-Coord(2,i))^2;
            %plot([Xxi(i),Coord(1,i)],[Yyi(i),Coord(2,i)],'y')
```

```
        end%error based on x&y difference >> y differences if 'artificial' point
            error=sqrt(sum(Error)/length(Error));%standard deviation
    end
    %plot(Xxi,Yyi);
end
%------------------------------------------------------------------
function myCleanupFun(fileID)
    fclose(fileID);
end
```

Listing B.2: Selected functions appended to `HolmMainGUI.m`

```
%% TRI SPHERE
function [SphereOpt,fvalS]=TriSphere(guessW,SphereCoord)
%-------------------------------------------------------------
%FIT CIRCLE (OPTIMISATION)
%-------------------------------------------------------------
    Guess=[guessW,guessW,guessW];%x0,y0,R
    Opts = optimset('Display','final','TolFun',1e-8,'Algorithm','active-set','LargeScale','on','MaxFunEval',1000);
    %                    x = fmincon(fun,                    x0,   A,b,Aeq,beq,lb,ub,nonlcon,options)
    [SphereOpt,fvalS,exitFlag,~]= fmincon(@(Guess)objTRI(Guess,SphereCoord),Guess,[],[],[],[],[-inf,-inf,0.01],[],[],Opts);
    switch exitFlag
        case 1
            disp('fmincon has converged properly.')
        case 0
            disp('fmincon has reached the max. number of iterations. Function may not have converged.')
        case -2
            disp('fmincon was unable to find a solution.')
    end
%-------------------------------------------------------------


% SPHERE OBJECTIVE
```

```
function eMin=objTRI(Guess,SphereCoord)
%-----------------------------------------------------------------
%Objective function for circle profile oftimisation
%-----------------------------------------------------------------
    x0=Guess(1);
    y0=Guess(2);
    R=Guess(3);

    Error2=zeros(1,size(SphereCoord,2));

    for j=1:size(SphereCoord,2)
        Error2(j)=sqrt(((SphereCoord(1,j)-x0)^2+(SphereCoord(2,j)-y0)^2-R^2)^2);
    end
    eMin=sqrt(sum(Error2))/size(SphereCoord,2);
%-----------------------------------------------------------------


%% TRI-CLEAN
function [Coord]=triClean(Edges,min)
%-----------------------------------------------------------------
% PICK EDGES using Canny edge detection method
% -- Edges = CLEAN edge matrix (white edges on black)
% -- min   = minimum area to count (pixles) (discard with lower area)
%-----------------------------------------------------------------

 %--Label matrix and stats.
    [L,num]=bwlabel(Edges);%repeat?
    Stats=regionprops(L,'Area','PixelIdxList');

 %--Sort by area (descending)
    Area=zeros(num,2);
    for R=1:num %region
        Area(R,:)=[R,Stats(R).Area];
    end %R
    Area=flipud(sortrows(Area,2));%sort based on area.  flipup(ascending)=descending.
```

```

 %--Find cut-off for minimum area (min)
     R=1;
     while R<=num&&Area(R,2)>=min,
         R=R+1;
     end
     R=R-1;
     RegionsOfInterest=Area(1:R,1);

 %--Combine all as linear indices
     IND=[];
     for k=1:R
         R=RegionsOfInterest(k,1);
         IND=[IND;Stats(R).PixelIdxList];%linear indices
     end %K = R plot

 %--Convert Linear indices, order
     s=size(Edges);%size of 'grey' image for converting linear indices
     [y,x]=ind2sub(s,IND);
     Coord=[transpose(x);transpose(y)];
%------------------------------------------------------------------------



%------------------------------------------------------------------------

function [CoordL,CoordR,x0a,y0a]=RotateCoords(Alpha,x0,y0,R,HolmCoordR,
    HolmCoordL,Grey,HL)
%------------------------------------------------------------------
    % rotate Coordinates
%------------------------------------------------------------------
    %--Main image and angles
     Thta=-Alpha*180/pi;
     imcentre=size(Grey)/2;
     GreyA=imrotate(Grey,Thta);%for plotting only
     imcentre2=size(GreyA)/2;


```

```
%--Rotate sphere centre (x0,y0);
xt=x0-imcentre(2);
yt=y0-imcentre(1);
Rs=sqrt(xt^2+yt^2);

if xt<0
    th=atan(yt/xt);
    x0a=-Rs*cos(th+Alpha)+imcentre2(2);
elseif xt>0
    th=atan(yt/xt);
    x0a=Rs*cos(th+Alpha)+imcentre2(2);
else %xt=0 >> phi = 90 deg
    th=pi/2;
    x0a=Rs*cos(th+Alpha)+imcentre2(2);
end

if strcmp(HL,'Low')==1
    y0a=-Rs*sin(th+Alpha)+imcentre2(1);
else
    y0a=Rs*sin(th+Alpha)+imcentre2(1);
end


%--Rotate holm coordinates
HCRA=zeros(4,size(HolmCoordR,2));
HCLA=zeros(4,size(HolmCoordL,2));

%...right
transMat=HolmCoordR;
transMat(1,:)=HolmCoordR(1,:)-imcentre(2);%translate to centre
transMat(2,:)=HolmCoordR(2,:)-imcentre(1);
for c=1:size(HolmCoordR,2)
    HCRA(3,c)=sqrt(transMat(1,c)^2+transMat(2,c)^2);%R
    HCRA(4,c)=atan(transMat(2,c)/transMat(1,c));%Thta
    HCRA(1,c)=HCRA(3,c)*cos(HCRA(4,c)+Alpha);%x'
    HCRA(2,c)=HCRA(3,c)*sin(HCRA(4,c)+Alpha);%y'
end
```

```
    HCRA(1,:)=HCRA(1,:)+imcentre2(2);%translate to centre
    HCRA(2,:)=HCRA(2,:)+imcentre2(1);

    %...left
    alphaL=Alpha-pi;

    transMat=HolmCoordL;
    transMat(1,:)=HolmCoordL(1,:)-imcentre(2);%translate to centre
    transMat(2,:)=HolmCoordL(2,:)-imcentre(1);

    for c=1:size(HolmCoordL,2)
        HCLA(3,c)=sqrt(transMat(1,c)^2+transMat(2,c)^2);%R
        HCLA(4,c)=atan(transMat(2,c)/transMat(1,c));%Thta
        HCLA(1,c)=HCLA(3,c)*cos(HCLA(4,c)+alphaL);%x'
        HCLA(2,c)=HCLA(3,c)*sin(HCLA(4,c)+alphaL);%y'
    end

    HCLA(1,:)=HCLA(1,:)+imcentre2(2);%translate to centre
    HCLA(2,:)=HCLA(2,:)+imcentre2(1);

CoordL=HCLA(1:2,:); CoordR=HCRA(1:2,:);
%       figure(1)
%       imshow(GreyA); hold on
%       plot(HCLA(1,:),HCLA(2,:),'r');
%       plot(HCRA(1,:),HCRA(2,:),'r');
%       plot(x0a,y0a,'ob');
%-------------------------------------------------------------------


% --- Executes during object creation, after setting all properties.
function ST_Version_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ST_Version (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
    called


```

```

%--------------------------------------------------------------
% Handles for ImMask and Tri are sent to ThreshGUI
%--------------------------------------------------------------
function [WIM,ThreshC,Thr]=WholeImageMask(Grey,BWadj,CANadjL,CANadjH,MorphProps,Flag,Path)
%MorphProps.size
%MorphProps.Type
%MorphProps.FilLoc
%--------------------------------------------------------------
    %GENERATE A BW MASK OVER THE WHOLE IMAGE
%--------------------------------------------------------------
%.. Thresholds
    Thr=BWadj*graythresh(Grey);
    if isempty(CANadjL)==1||isempty(CANadjH)==1 %if canny parameters aren't set
        [Edg,ThreshC]=edge(Grey,'canny');
        CANadjL=ThreshC(1);
        CANadjH=ThreshC(2);
    else
        if CANadjL>=CANadjH
            ThreshC=[0.95*CANadjH,CANadjH];
        else
            ThreshC=[CANadjL,CANadjH];
        end
    end

%--------------------------------------------------------------
%--- If the threshold is too low, no edges will be detected. Matrix will be
    %blank causing the mask to obscure all edges.
    % imshow(im2bw(Grey,0)) == while matrix (all ones)
    %The "don't use mask" toggle where Thresh=0 will also trigger this and
    %will send a blank matrix to fn(Tri).
%--- If BW matrix is blank, generate blank mask.
%--- If not blank, generate mask
```

```
%--- dual fitting areas (sphere or holm)
%------------------------------------------------------------------------

%------------------------------------------------------------------------
%-- Whole image mask
%...The drop and sphere should be the largest foreground object
%...Could have a problem with poor lighting if the drop and sphere turn
    out
%...as separate objects.
%------------------------------------------------------------------------
%.. BW image
    BW1=im2bw(Grey,Thr);
    BW2=imfill(~BW1,'holes');%fill holes
    SE3 = strel(MorphProps.Type,MorphProps.Size);%Create structural
        element
    BW3=imclose(BW2,SE3);%close image
    if isempty(MorphProps.FilLoc)==1
        BW4=BW3;
    else %flood fill
        BW4=imfill(BW3,MorphProps.FilLoc);%flood fill from user's
            selected points.
    end
%.. Identify the largest object
    CC=bwconncomp(BW4);
    L=labelmatrix(CC);%Create label matrix
    %imshow(label2rgb(L))
    Stats = regionprops(L,'Area');%get region properties
    if isempty(Stats)==1 %use default thresholds
        disp('Mask failed. Using default thresholds.')
        %.. BW image
        BW1=im2bw(Grey);
        BW2=imfill(~BW1,'holes');%fill holes
        BW3=imclose(BW2,SE3);%close image
        if isempty(MorphProps.FilLoc)==1
            BW4=BW3;
        else %flood fill
```

```
                BW4=imfill(BW3,MorphProps.FilLoc);%flood fill from user's selected points.
            end
            %.. Identify the largest object
            CC=bwconncomp(BW4);
            L=labelmatrix(CC);%Create label matrix
            %imshow(label2rgb(L))
            Stats = regionprops(L,'Area');%get region properties
        end
        [~,iMax] = max([Stats.Area]);%Identify largest region

        WIM=zeros(size(BW4));%create blank mask
        WIM(L==iMax)=1;%Add largest forground object to mask
        WIM=imfill(WIM,'holes');%close any remaining holes.
        %.. For Thresh GUI
        if Flag==1
            save(fullfile(Path,'ThreshTemp1'));
        end


%% MASK (generate mask for edge detection) %%% Call function handle from main GUI
function [CrdL,CrdR]=ImMask(CnrL,CnrR,Grey,ThreshC,Thr,SvFlag,wd,SBD,WIM,Path,NL,BMask)

        [W3]=WhiteMask(Thr,Grey,3);
        ThrB=0.05; Aug=1.5;

%------------------------------------------------------------------------
% LEFT MASK & CORDINATES
%------------------------------------------------------------------------
        Cnr=CnrL;
        try
            GreyL=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));
        catch
            disp('Size misspatch, LHS')
            Cnr=[CnrL(1),size(Grey,2),CnrL(3),size(Grey,1)];
```

```
    end
%MASK EDGES/SIDES (boundaries already defined)

%.. If empty matrix, 1 -> 0
    if Thr==0
        maskL=zeros(size(GreyL));%black
    else
%.. If BW is not empty:
    %---Find perimeter of mask
        PerimL=bwperim(WIM(Cnr(2):Cnr(4),Cnr(1):Cnr(3)));
    %---remove border
        PerimL(:,1)=0;
        PerimL(1,:)=0;
        PerimL(:,end)=0;
        PerimL(end,:)=0;
    %---Dilate perimeter
        maskL = ~imdilate(PerimL,strel('disk',wd),'same');%
    end

%.. Edge detection
[EdgeL]=Tri(GreyL,maskL,ThreshC,2*SBD,NL);%Canny edge detection (Edges,y1,x1)

%.. Bubble mask
if BMask == 1
    [BubMask]=CircMask(GreyL,ThrB,Aug);%BW mask, bubbles are white
    CleanL=EdgeL;
    CleanL(BubMask(:,1:end-1)==1)=0;
    %Red=GreyL;
    %Red(BubMask==1)=0.5;
    %imshow(cat(3, GreyL,Red,Red));
else
    CleanL=EdgeL;
end
%.. White mask
W=W3(Cnr(2):Cnr(4),Cnr(1):Cnr(3));
CleanL(W(:,1:end-1)==1)=0;
```

```
%------------------------------------------------------------------
%--Display matrix for bubble and white mask.
% figure(5);
% DispM=CleanL;
% DispM(W(:,1:end-1)==1)=2;
% DispM(BubMask(:,1:end-1)==1)=3;
% DispM(EdgeL==1)=1;
% imshow(label2rgb(DispM));
%------------------------------------------------------------------

if max(max(CleanL))==0
    disp('No edge found.')
end

CleanSt = regionprops(CleanL,'PixelIdxList');
[y1,x1]=ind2sub(size(CleanL),CleanSt.PixelIdxList);
CrdL=sortrows([x1,y1],2);%sort rows
if isempty(y1)==1
    fprintf('Edge detection has failed (left) - no edge detected.
        Continuing to next frame.')
end     %....Debugging
    %imshow(maskL+EdgeL)
    %plot(CrdL(:,1),CrdL(:,2),'r')

%.. Coordinates
CrdL(:,1)=CrdL(:,1)+CnrL(1);
CrdL(:,2)=CrdL(:,2)+CnrL(2);
%plot(CrdL(:,1),CrdL(:,2),'r')

%------------------------------------------------------------------------------------
% RIGHT MASK & CORDINATES
%------------------------------------------------------------------------------------
Cnr=CnrR;
GreyR=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));

%.. If empty matrix, 1 -> 0
```

```
if Thr==0
    maskR=zeros(size(GreyR));
else
%.. If BW is not empty:
    %---Find perimeter of mask
        PerimR=bwperim(WIM(Cnr(2):Cnr(4),Cnr(1):Cnr(3)));
    %---remove border
        PerimR(:,1)=0;
        PerimR(1,:)=0;
        PerimR(:,end)=0;
        PerimR(end,:)=0;
    %---Dilate perimeter
        maskR = ~imdilate(PerimR,strel('disk',wd),'same');%
end
%.. Edge detection
[EdgeR]=Tri(GreyR,maskR,ThreshC,SBD,NL);%Canny edge detection (Edges,
    y1,x1)

%.. Bubble mask
if BMask == 1
    [BubMask]=CircMask(GreyR,ThrB,Aug);%BW mask, bubbles is white

    CleanR=EdgeR;
    CleanR(BubMask(:,1:end-1)==1)=0;
else
    CleanR=EdgeR;
end
%.. White mask
W=W3(Cnr(2):Cnr(4),Cnr(1):Cnr(3));
CleanR(W(:,1:end-1)==1)=0;

%.. Coordinates
CleanSt = regionprops(CleanR,'PixelIdxList');
[y2,x2]=ind2sub(size(GreyR),CleanSt.PixelIdxList);
CrdR=sortrows([x2,y2],2);%sort rows
CrdR(:,1)=CrdR(:,1)+CnrR(1);
CrdR(:,2)=CrdR(:,2)+CnrR(2);
```

```
%plot(CrdR(:,1),CrdR(:,2),'r')

if isempty(y2)==1
    fprintf('Edge detection has failed (right) - no edge detected. Continuing to next frame.')
end

%.. For Thresh GUI
    if SvFlag==1
        save(fullfile(Path,'ThreshTemp2'));
    end
%--------------------------------------------------------------------------
%% Generate circle mask
%Bubbles are circular and dark
function [BubMask]=CircMask(Grey,Thr,Aug)
%[centers, radii] = imfindcircles(Grey,[10 100],'ObjectPolarity','dark');
%Thr=0.1; Aug=2;
[centers, radii] = imfindcircles(Grey,[10 30],'ObjectPolarity','dark','EdgeThreshold',Thr);
[centers2, radii2] = imfindcircles(Grey,[30 90],'ObjectPolarity','dark','EdgeThreshold',Thr);
[centers3, radii3] = imfindcircles(Grey,[90 150],'ObjectPolarity','dark','EdgeThreshold',Thr);

centers=[centers;centers2;centers3];
radii=[radii;radii2;radii3];

BubMask=zeros(size(Grey));

if isempty(radii)==0
    T=table(centers,radii);
    %imshow(Grey);
    %h = viscircles(centers,radii);

    for i=1:size(T,1)
        %(x-x0)2+(y-y0)2=R2 -> y=sqrt(R2 -(x-x0)2)+-y0
```

```
            xMin=int32(max(T.centers(i,1)-Aug*T.radii(i),1));
            xMax=int32(min(T.centers(i,1)+Aug*T.radii(i),size(Grey,2)));
            for j=xMin+1:xMax-1
                a=sqrt((Aug*T.radii(i))^2-(double(j)-T.centers(i,1))^2);
                yMin=int32(max(T.centers(i,2)-a,1));
                yMax=int32(min(T.centers(i,2)+a,size(Grey,1)));
                BubMask(yMin:yMax,j)=1;
            end%for x range of circle
        end
end

%% White Mask
function [W3]=WhiteMask(Th,Grey,n)
%Create a "white" mask
%...Th is the adjusted BW threshold (Thresh*BWadj)
W1=~im2bw(Grey,Th);
imshow(W1);

W2 = ~bwmorph(W1,'close');
%.. Identify the largest object
CC=bwconncomp(W2);
L=labelmatrix(CC);%Create label matrix
imshow(label2rgb(L));

 Stats=regionprops(L,'Area');
 [~,iMax] = max([Stats.Area]);
W2(L==iMax)=0;
W3=W2;
for i=1:n
    W3 = bwmorph(W3,'dilate');
end
%-----------------------------------------------------------------------

%% TRI (%%%Call function handle from main GUI)
%Called from ImMask
function [CrdMat]=Tri(Grey,Mask,ThreshC,FlagOptns,NL)
 %FlagOptns={SBD,WhiteMask(WM),BubMask}
```

```
%———————————————————————————————————————————————————————————
% PICK EDGES using Canny edge detection method
%———————————————————————————————————————————————————————————
 %——Parameters
    N=2;%number of lines (units/components) to process
    C=10;%X distance to clear (in Pixels)

 %——Pick edges
    Edges=edge(Grey,'canny',ThreshC);

    %Subtract mask
    Edges(Mask==1)=0; %Replaces subtraction

 %——Determine longest line
 if max(max(Mask))==0 %max mask value is zero —> inactive —> LONGEST
     LINE SEARCH
    imsk = bwmorph(Edges,'thin',Inf);%thinning edges to single line
        thicknss
    bpoints = bwmorph(imsk,'branchpoints',1);%searching for points at
        intercies of lines
    imsk(bpoints)=0;%remove points at intersections − all lines now
        distinct

    %Pull up area properties, sort
    lineStats = regionprops(imsk, {'Area','PixelList'});
    [Stats,indi]=sortrows(struct2table(lineStats),1,'descend');

    %Label matrix
    CCl = bwconncomp(imsk);
    Ll = labelmatrix(CCl);
    %Make new edge matrix from the NL largest lines
    %default(NL)=2;%NL is also saved in TrialThO
    CrdMat=zeros(size(Edges));
    for i=1:NL
        CrdMat(Ll==indi(i))=1;
    end
```

```

else
    CC=bwconncomp(Edges);
    Ll = labelmatrix(CC);
    Stats = regionprops(CC,'area','PixelIdxList');
    idx = find([Stats.Area] > 20); %to update with GUI
    %imshow(ismember(labelmatrix(CC), idx));
    CrdMat=zeros(size(Edges));
    IND=[];
    for i=idx
        IND=[IND;Stats(i).PixelIdxList];%linear indices %Produces a cell array
        CrdMat(Ll==i)=1;
    end
    %New Canny edge (no mask) for SBE
    Edges=edge(Grey,'canny',ThreshC);
    imsk = bwmorph(Edges,'thin',Inf);%thinning edges to single line thicknss
    bpoints = bwmorph(imsk,'branchpoints',1);%searching for points at intercies of lines
    imsk(bpoints)=0;%remove points at intersections - all lines now distinct
    CCl = bwconncomp(imsk);
    Ll = labelmatrix(CCl);
end

%FlagOptns={SBD,WhiteMask(WM),BubMask}
if FlagOptns(1) == 1 %(right)(SBE)
    %...Clear components touching the edge
        %...Canny seems to clear the final row of edge pixels. Isolate top and right facing pixels.
        Edge2=imclearborder(imsk(:,1:end-1));
        Edge3=zeros(size(imsk));
        Edge3(:,1:end-1)=Edge2;

    %...Subtract to Search for lines connected to the right-most edge
        Edge4=imsk;
```

```
        Edge4(Edge3==1)=0;
        %imshow(Edge4+0.5*Edges-0.5);

    %...Region properties - using major axis for length.
        CC = bwconncomp(Edge4);
        if CC.NumObjects>0 %objects exist
            L = labelmatrix(CC);
            Stats=regionprops(L,'PixelList','area','majorAxisLength');
            for i=1:length(Stats);
                Stats(i).Label=i;
            end%add column with label
            t=struct2table(Stats,'AsArray', true);
        %...Identify the largest major axis lengths.
            t2=sortrows(t,'MajorAxisLength','descend');
            if size(t2,1)<N
               Labels=t2.Label(:);
            else
               Labels=t2.Label(1:N);
            end
        %...Working on the longest lengths, clear the left-most (inside
            ) 5 x width
            NewMask=zeros(size(Edge3));%
            for l=Labels
                SegCrd=table2array(t.PixelList(l));
                %SegCrd will be sorted by x-value, ascending.
                for i=1:length(SegCrd)
                    if SegCrd(i,1)>SegCrd(1,1)+C,
                        NewMask(SegCrd(i,2),SegCrd(i,1))=1;
                    end
                end%i=1:length(SegCrd)
            end%for l=Labels

        %...Search again.
            CC = bwconncomp(NewMask);
            L = labelmatrix(CC);
            Stats=regionprops(L,'PixelIdxList','PixelList','Area','
                MajorAxisLength','Orientation');
```

```
            for i=1:length(Stats);
                Stats(i).Label=i;
            end%add column with label
            t=struct2table(Stats,'AsArray', true);
            t2=sortrows(t,'MajorAxisLength','descend');

        %...Entry with largest major axis length, with POSITIVE
            orientation (angles LL to TR **RIGHT SIDE)
            i=0;
            while i<length(Stats)-1 && t2.Orientation(i+1)<=0
                i=i+1;
            end%while

            if i~=length(Stats)%segments matching the critera were
                found.
             %--Remove "longest line" coords overlapping the new x
                 coordinates.
                %--x limit
                CrdB=table2array(t2.PixelList(i+1,:));
                yr = CrdB(1,2);
                %--Add longest line
%                   CrdMat=zeros(size(NewMask));
%                   CrdMat(Ll==index)=1;
                %--Clear SBD area and add SBD
                CrdMat(1:yr,:)=0;
                CrdMat(L==t2.Label(i+1))=1;%t2.Label(i+1) --> component
                    label
                Stats = regionprops(CrdMat, 'PixelIdxList');
                IND=Stats(1).PixelIdxList;%linear indices
            end%if
        end%(if numobj==1)

    elseif FlagOptns ==2 %(left)
    %...Clear components touching the edge
    %...Canny seems to clear the final row of edge pixels. Isolate top
        and right facing pixels.
        %EdgeL=edge(Grey,'canny',ThreshC);
```

```
        Edge2=imclearborder(imsk(:,2:end));
        Edge3=zeros(size(imsk));
        Edge3(:,2:end)=Edge2;

    %...Subtract to Search for lines connected to the right-most edge
        Edge4=imsk;
        Edge4(Edge3==1)=0;
        %imshow(Edge4+0.5*Edges-0.5);

    %...Region properties - using major axis for length.
        CC = bwconncomp(Edge4);
        if CC.NumObjects>0 %objects exist
            L = labelmatrix(CC);
            Stats=regionprops(L,'PixelList','area','majorAxisLength');
            for i=1:length(Stats);
                Stats(i).Label=i;
            end%add column with label
            t=struct2table(Stats,'AsArray', true);
        %...Identify the largest major axis lengths.
            t2=sortrows(t,'MajorAxisLength','descend');
            if size(t2,1)<N
               Labels=t2.Label(:);
            else
               Labels=t2.Label(1:N);
            end
            %...Working on the longest lengths, clear the left-most (inside) 5 x width
            NewMask=zeros(size(Edge2));%smaller
            for l=Labels
                SegCrd=flipud(table2array(t.PixelList(l)));%innermost x = x1
                for i=1:length(SegCrd)
                    if SegCrd(i,1)<SegCrd(1,1)-C,
                        NewMask(SegCrd(i,2),SegCrd(i,1))=1;
                    end
                end%i=1:length(SegCrd)
            end%for l=Labels
```

```

        %...Search again.
            CC = bwconncomp(NewMask);
            L = labelmatrix(CC);
            Stats=regionprops(L,'PixelList','Area','MajorAxisLength','Orientation');
            for i=1:length(Stats);
                Stats(i).Label=i;
            end%add column with label
            t=struct2table(Stats,'AsArray', true);
            t2=sortrows(t,'MajorAxisLength','descend');

        %...Entry with largest major axis length, with NEGAIVE orientation (angles LR to TL **LEFT SIDE)
            i=0;
            while i<length(Stats)-1 && t2.Orientation(i+1)>=0
                i=i+1;
            end%while
            %t2.MajorAxisLength(1);
            if i~=length(Stats)%segments matching the critera were found.
                %--Remove "longest line" coords overlapping the new x coordinates.
                %--x limit
                CrdB=table2array(t2.PixelList(i+1,:));
                yl = CrdB(end,2);
                %--Add longest line
%                   CrdMat=zeros(size(NewMask));
%                   CrdMat(Ll==index)=1;
                %--Clear SBD area (above) and add SBD
                CrdMat(1:yl,:)=0;
                CrdMat(L==t2.Label(i+1))=1;%t2.Label(i+1) --> component label
                Stats = regionprops(CrdMat, 'PixelIdxList');
                IND=Stats(1).PixelIdxList;%linear indices
            end%if
        end%if numobj==1
```

```
 end%(if SBD=)
 %------------------------------------------------------------------------


function ET_Notes_Callback(hObject, eventdata, handles)
%------------------------------------------------------------------------
%------------------------------------------------------------------------

% --- Executes during object creation, after setting all properties.
function ET_Notes_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in CB_Print.
function CB_Print_Callback(hObject, eventdata, handles)
% Hint: get(hObject,'Value') returns toggle state of CB_Print


% --- Executes on selection change in PU_Data.
function PU_Data_Callback(hObject, eventdata, handles)
%------------------------------------------------------------------------
%-- Enable ET_BestNum if call back "best" is used.
%------------------------------------------------------------------------


% --- Executes during object creation, after setting all properties.
function PU_Data_CreateFcn(hObject, eventdata, handles)
%------------------------------------------------------------------------
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%------------------------------------------------------------------------
```

```
% --- Executes on selection change in PU_Side.
function PU_Side_Callback(hObject, eventdata, handles)
% hObject    handle to PU_Side (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns PU_Side contents as cell array
%        contents{get(hObject,'Value')} returns selected item from PU_Side


% --- Executes during object creation, after setting all properties.
function PU_Side_CreateFcn(hObject, eventdata, handles)
% hObject    handle to PU_Side (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in PB_LoadResults.
function PB_LoadResults_Callback(hObject, eventdata, handles)
%-------------------------------------------------------------------------
%-- ALLOW USER TO CALL A SPECIFIC RESULTS FILE
    [ResultFile,Path]=uigetfile('.mat','multiselect','off')
    FullFileName=fullfile(Path,ResultFile);
    load(FullFileName);

%-- Plot
    axes(handles.axes2), hold on
```

```
    plot(Results(:,1),Results(:,21),'+r','markersize',3)%plot gamma (
        aVE)
    xlabel('time (min)'); ylabel('interfacial tension (L,R BEST), mN/mm
        , Temp(C)');
%------------------------------------------------------------------------



function ET_BestNum_Callback(hObject, eventdata, handles)
%------------------------------------------------------------------------

%------------------------------------------------------------------------


% --- Executes during object creation, after setting all properties.
function ET_BestNum_CreateFcn(hObject, eventdata, handles)
%------------------------------------------------------------------------
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%------------------------------------------------------------------------


% --- Executes on button press in CB_MWtog.
function CB_MWtog_Callback(hObject, eventdata, handles)
%------------------------------------------------------------------------
load(fullfile(handles.Path,'MData.mat')); %microwave data
%-- PLOT MICROWAVE RUNNING TIME
MS=MicrStart/60;%frame when microwave is turned on
MO=MicrStart/60+irTime;%frame when microwave is turned off
plot([MS,MS],[0,100],'y',[MO,MO],[0,100],'y');
%------------------------------------------------------------------------


% --- Executes on button press in CB_Temp.
function CB_Temp_Callback(hObject, eventdata, handles)
```

```
%------------------------------------------------------------------------
Tnum=get(handles.DD_Temp,'Value');
load(fullfile(handles.Path,sprintf('TC%i.mat',Tnum)));
TC2=[TC,transpose(1:length(TC))];
TC2(:,2)=TC2(:,2)/60;

axes(handles.axes2)
plot(TC2(:,2),TC2(:,1),'g');
%------------------------------------------------------------------------


% --- Executes on button press in PB_SaveFig.
function PB_SaveFig_Callback(hObject, eventdata, handles)
%------------------------------------------------------------------------
%-- SAVE GUI FIGURE
%------------------------------------------------------------------------
folder=getappdata(0,'folder');
saveas(gcf,fullfile(folder,'ResultsMainGUI'),'fig');
fprintf('figure saved');
%------------------------------------------------------------------------



% --- Executes on button press in PB_Cnr.
function PB_Cnr_Callback(hObject, eventdata, handles)
%------------------------------------------------------------------------
    if isfield(handles,'ImName')==1
        DropMovie = VideoReader(handles.ImName);
        Grey = read(DropMovie, handles.ST);
        if get(handles.CB_Flip,'Value')==1
            Grey=flipud(Grey);
            set(handles.CB_Flip,'value',1);
        end
        hold off, imshow(Grey);

        button=questdlg('Crop images?','Crop?','Yes','No','No');
        if strcmp(button,'Yes')==1
```

```
                fprintf(1,'Select area to crop\n')
                rec=round(getrect());
                Crop=[rec(1),rec(2),rec(1)+rec(3),rec(2)+rec(4)];
                if Crop(3)>size(Grey,2);Crop(3)=size(Grey,2);end
                if Crop(4)>size(Grey,1);Crop(4)=size(Grey,1);end
            else
                Crop=handles.Crop;
            end
            Cnr=Crop;
            Grey=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));
            %update
            handles.Crop=Crop;
            handles.Grey=Grey;
        else
%           ImName=sprintf('%s%04i.%s',handles.Prefix,handles.ST,handles.
    Ext);
%           Grey = imread(ImName);
            Grey=handles.Grey; %already cropped)
        end

%--- REDEFINE FITTING AREAS
        axes(handles.axes1); hold off
        imshow(Grey); hold on

        fprintf(1,'Selecting an area outside of image boundaries will
            return an error\n')
        fprintf(1,'Please select the area to fit for the holm (right)\n(
            click and drag box):\n')
        rec=round(getrect());
        HlmCnrR=[rec(1),rec(2),rec(1)+rec(3),rec(2)+rec(4)];
        if HlmCnrR(3)>size(Grey,2);HlmCnrR(3)=size(Grey,2);end
        if HlmCnrR(4)>size(Grey,1);HlmCnrR(4)=size(Grey,1);end

        fprintf(1,'Please select the area to fit for the holm (left)\n(
            click and drag box):\n')
        rec=round(getrect());
        HlmCnrL=[rec(1),rec(2),rec(1)+rec(3),rec(2)+rec(4)];
```

```
    if HlmCnrL(3)>size(Grey,2);HlmCnrL(3)=size(Grey,2);end
    if HlmCnrL(4)>size(Grey,1);HlmCnrL(4)=size(Grey,1);end

    HlmCnr=[HlmCnrR;HlmCnrL];

    fprintf(1,'Please select the area to fit for the sphere (right) \n(click and drag box):\n')
    rec=round(getrect());
    SphCnr(1,:)=round([rec(1),rec(2),rec(1)+rec(3),rec(2)+rec(4)]);

    fprintf(1,'Please select the area to fit for the sphere (left) \n(click and drag box):\n')
    rec=round(getrect());
    SphCnr(2,:)=[rec(1),rec(2),rec(1)+rec(3),rec(2)+rec(4)];

    %Display boxes
    plot([HlmCnrR(1),HlmCnrR(3),HlmCnrR(3),HlmCnrR(1),HlmCnrR(1)],[HlmCnrR(2),HlmCnrR(2),HlmCnrR(4),HlmCnrR(4),HlmCnrR(2)],'r');
    plot([HlmCnrL(1),HlmCnrL(3),HlmCnrL(3),HlmCnrL(1),HlmCnrL(1)],[HlmCnrL(2),HlmCnrL(2),HlmCnrL(4),HlmCnrL(4),HlmCnrL(2)],'r');
    plot([SphCnr(1,1),SphCnr(1,3),SphCnr(1,3),SphCnr(1,1),SphCnr(1,1)],[SphCnr(1,2),SphCnr(1,2),SphCnr(1,4),SphCnr(1,4),SphCnr(1,2)],'r');
    plot([SphCnr(2,1),SphCnr(2,3),SphCnr(2,3),SphCnr(2,1),SphCnr(2,1)],[SphCnr(2,2),SphCnr(2,2),SphCnr(2,4),SphCnr(2,4),SphCnr(2,2)],'r');

    %Save
    handles.HlmCnr=HlmCnr;
    handles.SphCnr=SphCnr;
    guidata(hObject, handles);
%------------------------------------------------------------
```

Bibliography

[1] Manar El-Sayed Abdel-Raouf. Factors affecting the stability of crude oil emulsions. *Crude Oil Emuls. Stab. Charact.*, pages 183–204, 2012. doi: 10.5772/35018.

[2] Anita Abedi, Nasser Safari, Vahid Amani, and Hamid Reza Khavasi. Synthesis, characterization, mechanochromism and photochromism of $[Fe(dm4bt)_3][Fecl_4]_2$ and $[Fe(dm4bt)_3][Febr_4]_2$, along with the investigation of steric influence on spin state. *Dalt. Trans.*, 40(26):6877, 2011. ISSN 1477-9226. doi: 10.1039/c0dt01508c.

[3] Noam Agmon. The Grotthuss mechanism. *Chem. Phys. Lett.*, 244(5-6):456–462, 1995. ISSN 00092614. doi: 10.1016/0009-2614(95)00905-J.

[4] T. Al-Sahhaf, A. Elkamel, A. Suttar Ahmed, a. R. Khan, and A Suttar Ahmed. The influence of temperature, pressure, salinity, and surfactant concentration on the interfacial tension of the n-octane-water system. *Chem. Eng. Commun.*, 192(5):667–684, 2005. ISSN 0098-6445. doi: 10.1080/009864490510644.

[5] M. Shahrooz Amin, Shihab Elborai, Se Hee Lee, Xiaowei He, and Markus Zahn. Surface tension measurement techniques of magnetic fluids at an interface between different fluids using perpendicular field instability. *J. Appl. Phys.*, 97(10):1–4, 2005. ISSN 00218979. doi: 10.1063/1.1861374.

[6] M C Amiri and Ali A Dadkhah. On reduction in the surface tension of water due to magnetic treatment. *Colloids Surfaces A Physicochem. Eng. Asp.*, 278 (1-3):252–255, 2006. doi: 10.1016/j.colsurfa.2005.12.046.

[7] Masahiro Asada, Yushin Kanazawa, Yusuke Asakuma, Itsuro Honda, and Chi Phan. Surface tension and oscillation of water droplet under microwave radiation. *Chem. Eng. Res. Des.*, 101:107–112, 2015. ISSN 02638762. doi: 10.1016/j.cherd.2015.05.019.

[8] Yusuke Asakuma, Takuya Munenaga, and Ryosuke Nakata. Observation of bubble formation in water during microwave irradiation by dynamic light scattering. *Heat Mass Transf.*, 52(9):1833–1840, 2016. ISSN 0947-7411. doi: 10.1007/s00231-015-1703-3.

[9] C Atae-Allah, M Cabrerizo-Vílchez, J F Gómez-Lopera, J A Holgado-Terriza, R Román-Roldán, and P L Luque-Escamilla. Measurement of surface tension and contact angle using entropic edge detection. *Meas. Sci. Technol.*, 12(3): 288–298, March 2001. ISSN 0957-0233. doi: 10.1088/0957-0233/12/3/307.

[10] G M Ataev. Anomalous temperature dependence of interfacial tension in water-hydrocarbon mixtures. *Russ. J. Phys. Chem. A*, 81(12):2094–2095, 2007. ISSN 0036-0244. doi: 10.1134/S003602440712031X.

[11] Shiva Badban, Anita E. Hyde, and Chi M. Phan. Hydrophilicity of nonanoic acid and its conjugate base at the air/water interface. *ACS Omega*, 2(9):5565–5573, 2017. ISSN 2470-1343. doi: 10.1021/acsomega.7b00960.

[12] Francis Bashforth and John Couch Adams. *An attempt to test the theories of capillary action: by comparing the theoretical and measured forms of drops of fluid, with an explanation of the method of integration employed in constucting the tables.* Cambridge Eng.: University Press, 1883.

[13] A Bateni, S S Susnar, A Amirfazli, and A W Neumann. Development of a new methodology to study drop shape and surface tension in electric fields. *Langmuir*, 20(18):7589–7597, 2004. doi: 10.1021/la0494167.

[14] A Bateni, A Amirfazli, and A.W. Neumann. Effects of an electric field on the surface tension of conducting drops. *Colloids Surfaces A Physicochem. Eng. Asp.*, 289:25–38, 2006.

[15] C. J. Beverung, Clayton J. Radke, and Harvey W. Blanch. Protein adsorption at the oil/water interface: characterization of adsorption kinetics by dynamic interfacial tension measurements. *Biophys. Chem.*, 81(1):59–80, 1999. ISSN 0301-4622. doi: 10.1016/S0301-4622(99)00082-4.

[16] Eleanor R. Binner, John P. Robinson, Sam W. Kingman, Ed H. Lester, Barry J. Azzopardi, Georgios Dimitrakis, and John Briggs. Separation of oil/water emulsions in continuous flow using microwave heating. *Energy & Fuels*, 27 (6):3173–3178, 2013. ISSN 0887-0624. doi: 10.1021/ef400634n.

[17] Olle Björneholm, Martin H. Hansen, Andrew Hodgson, Li-Min Liu, David T. Limmer, Angelos Michaelides, Philipp Pedevilla, Jan Rossmeisl, Huaze Shen, Gabriele Tocci, Eric Tyrode, Marie-Madeleine Walz, Josephina Werner, and Hendrik Bluhm. Water at interfaces. *Chem. Rev.*, 116(13): acs.chemrev.6b00045, 2016. ISSN 0009-2665. doi: 10.1021/acs.chemrev.6b00045.

[18] Vasily A Bolotov, Evgeny I Udalov, Valentin N Parmon, Yuriy Yu Tanashev, and Yuriy D Chernousov. Pyrolysis of heavy hydrocarbons under microwave heating of catalysts and adsorbents. *J. Microw. Power Electromagn. Energy*, 46(1): 39–46, 2012. ISSN 0832-7823.

[19] R. P. Borwankar and D. T. Wasan. Dynamic interfacial tensions in acidic crude oil/caustic systems. Part I: A chemical diffusion-kinetic model for enhanced oil recovery. *AIChE J.*, 32(3):455–466, 1986. ISSN 15475905. doi: 10.1002/aic.690320312.

[20] R. P. Borwankar and D. T. Wasan. Dynamic interfacial tensions in acidic crude oil/caustic systems Part II : Role of dynamic effects in alkaline flooding for. *AIChE J.*, 32(3):467–476, 1986.

[21] E Boucher. Capillary phenomena: Properties of systems with fluid/fluid interfaces. *Reports Prog. Phys.*, 43(4):497, 1980.

[22] E Boucher and T Jones. Capillary phenomena. XVIII. - conditions for the flotation of solid spheres at liquid/liquid and liquid/vapour interfaces in a gravitational field. *J. Chem. Soc. Faraday Trans.*, 78(5):1499–1506, 1982.

[23] E Boucher and H Kent. Capillary phenomena. VI. behaviour associated with the flotation and mechanical manipulation of solid spheres at fluid interfaces. *J. Chem. Soc. Faraday Trans.*, 74:846–857, 1978.

[24] E Boucher, M Evans, and T Jones. The computation of interface shapes for capillary systems in a gravitational field. *Adv. Colloid Interface Sci.*, 27(1-2): 43–79, 1987.

[25] E A Boucher. Capillary phenomena. IV. thermodynamics of rotationally-symmetric fluid bodies in a gravitational field. *Proc. R. Soc. Lond. A. Math. Phys. Sci.*, 358(1695):519–533, 1978. ISSN 00804630.

[26] E A Boucher and H J Kent. Capillary phenomena. III. properties of rotationally symmetric fluid bodies with one asymptote - holms. *Proc. R. Soc. London A Math. Phys. Eng. Sci.*, 356(1684):61–75, 1977. ISSN 0080-4630. doi: 10.1098/rspa.1977.0121.

[27] Ernest A Boucher and Timothy G J Jones. Capillary phenomena. part 8.–bridge meridians for fluid interfaces with solids in cylindrical vessels: computation, Bessel approximations and comparison of properties of finite and infinite systems. *J. Chem. Soc., Faraday Trans. 1*, 75(0):1929–1939, 1979. doi: 10.1039/F19797501929.

[28] P. Brown, T. Alan Hatton, and J. Eastoe. Magnetic surfactants. *Curr. Opin. Colloid Interface Sci.*, 20(3):140–150, August 2015. ISSN 13590294. doi: 10.1016/j.cocis.2015.08.002.

[29] Paul Brown, Craig P. Butts, Jing Cheng, Julian Eastoe, Christopher A. Russell, and Gregory N. Smith. Magnetic emulsions with responsive surfactants. *Soft Matter*, 8(29):7545–7546, 2012. ISSN 1744-683X. doi: 10.1039/c2sm26077h.

[30] Paul Brown, Alexey Bushmelev, Craig P Butts, Jean-Charles Eloi, Isabelle Grillo, Peter J Baker, Annette M Schmidt, and Julian Eastoe. Properties of new magnetic surfactants. *Langmuir*, 29(10):3246–51, March 2013. ISSN 1520-5827. doi: 10.1021/la400113r.

[31] Paul Brown, Craig P Butts, and Julian Eastoe. Stimuli-responsive surfactants. *Soft Matter*, 9(8):2365–2374, 2013. ISSN 1744-683X. doi: 10.1039/C3SM27716J.

[32] K.A Burrill and D.R Woods. Change in interface and film shapes for a deformable drop at a deformable liquid-liquid interface. *J. Colloid Interface Sci.*, 30(4): 511–524, August 1969. ISSN 00219797. doi: 10.1016/0021-9797(69)90420-2.

[33] Hans-Jurgen Butt, Karlheinz Graf, and Michael Kappl. *Physics and chemistry of interfaces*, volume 1. Wiley and Sons, 1 edition, 2003. ISBN 3527406298.

[34] M G Cabezas, A Bateni, J M Montanero, and A W Neumann. A new drop-shape methodology for surface tension measurement. *Appl. Surf. Sci.*, 238 (1-4):480–484, 2004. doi: 10.1016/j.apsusc.2004.05.250.

[35] M Guadalupe Cabezas, Arash Bateni, José M Montanero, and A Wilhelm Neumann. Determination of surface tension and contact angle from the shapes of axisymmetric fluid interfaces without use of apex coordinates. *Langmuir*, 22(24):10053–10060, 2006. doi: 10.1021/la061928t.

[36] John Canny. A computational approach to edge detection. *Pattern Anal. Mach. Intell. IEEE Trans.*, PAMI-8(6):679–698, 1986. doi: 10.1109/TPAMI.1986. 4767851.

[37] Ana Maria Carmona-Ribeiro. Synthetic amphiphile vesicles. *Chem. Soc. Rev.*, 21(3):209–214, 1992.

[38] Mankin Chan and Teh Fu Yen. A chemical equilibrium model for interfacial activity of crude oil in aqueous alkaline solution: The effects of pH, alkali and salt. *Can. J. Chem. Eng.*, 60(2):305–308, 1982. ISSN 1939019X. doi: 10.1002/ cjce.5450600215.

[39] Chien-Hsiang Chang and Elias I. Franses. Adsorption dynamics of surfactants at the air/water interface: a critical review of mathematical models, data, and mechanisms. *Colloids Surfaces A Physicochem. Eng. Asp.*, 100(0):1–45, 1995. ISSN 09277757. doi: 10.1016/0927-7757(94)03061-4.

[40] H. Chen, Jesus L. Muros-Cobos, Juan A. Holgado-Terriza, and A. Amirfazli. Surface tension measurement with a smartphone using a pendant drop. *Colloids Surfaces A Physicochem. Eng. Asp.*, 533(July):213–217, 2017. ISSN 09277757. doi: 10.1016/j.colsurfa.2017.08.019.

[41] P Cheng and A W Neumann. Computational evaluation of axisymmetric drop shape analysis-profile (ADSA-P). *Colloids and Surfaces*, 62:297–305, 1992.

[42] P. Cheng, D. Li, L. Boruvka, Y. Rotenberg, and a.W. W Neumann. Automation of axisymmetric drop shape analysis for measurements of interfacial tensions and contact angles. *Colloids and Surfaces*, 43(2):151–167, January 1990. ISSN 01666622. doi: 10.1016/0166-6622(90)80286-D.

[43] Chris I. Chiwetelu, Vladimir Hornof, and Graham H. Neale. A dynamic model for the interaction of caustic reagents with acidic oils. *AIChE J.*, 36(2):233–241, 1990. ISSN 15475905. doi: 10.1002/aic.690360209.

[44] P G de Gennes and F Brochard-Wyart. *Gouttes, bulles, perles et ondes*. Échelles. Humensis, 2015. ISBN 9782701185583.

[45] Cesar de la Fuente-Nunez, Paul Brown, Marcelo D.T. Torres, Jicong Cao, and Timothy K. Lu. Magnetic surfactant ionic liquids and polymers with tetrahaloferrate (III) anions as antimicrobial agents with low cytotoxicity. *Colloid Interface Sci. Commun.*, 22:11–13, 2018. ISSN 22150382. doi: 10.1016/j.colcom.2017.11.002.

[46] Antonio De Nicola, Toshihiro Kawakatsu, Camillo Rosano, Massimo Celino, Mattia Rocco, and Giuseppe Milano. Self-assembly of Triton X-100 in water solutions: A multiscale simulation study linking mesoscale to atomistic models. *J. Chem. Theory Comput.*, 11(10):4959–4971, 2015. ISSN 15499626. doi: 10.1021/acs.jctc.5b00485.

[47] Shigeru Deguchi and Nao Ifuku. Bottom-up formation of dodecane-in-water nanoemulsions from hydrothermal homogeneous solutions. *Angew. Chemie - Int. Ed.*, 52(25):6409–6412, 2013. ISSN 14337851. doi: 10.1002/anie.201301403.

[48] O del Rio and A W Neumann. Axisymmetric drop shape analysis: computational methods for the measurement of interfacial properties from the shape and dimensions of pendant and sessile drops. *J. Colloid Interface Sci.*, 196(2):136–147, 1997.

[49] Rico E Del Sesto, T Mark McCleskey, Anthony K Burrell, Gary A Baker, Joe D Thompson, Brian L Scott, John S Wilkes, and Peg Williams. Structure and magnetic behavior of transition metal based ionic liquids. *Chem. Commun.*, 1(4):447–449, 2008. doi: 10.1039/B711189D.

[50] Agnes Derecskei-Kovacs, Bela Derecskei, and Zoltan A. Schelly. Atomic-level molecular modeling of the nonionic surfactant Triton X-100: The OPE9 component in vacuum and water. *J. Mol. Graph. Model.*, 16(4-6):206–212, 1998. ISSN 10933263. doi: 10.1016/S1093-3263(99)00011-X.

[51] E A Disalvo and S A Simon. *Permeability and Stability of Lipid Bilayers*. Taylor & Francis, 1995. ISBN 9780849345319.

[52] R O Duda and P E Hart. *Pattern classification*. Pattern Classification and Scene Analysis. John Wiley & Sons, 1973. ISBN 9780471056690.

[53] John Eastman. Colloid stability. *Colloid Sci. Princ. Methods Appl.*, pages 36–49, 2009. ISSN 0196-4321. doi: 10.1002/9781444305395.ch3.

[54] V. B. Fainerman and R. Miller. Simple method to estimate surface tension of mixed surfactant solutions. *J. Phys. Chem. B*, 105(46):11432–11438, 2001. ISSN 10895647. doi: 10.1021/jp004179b.

[55] Merv Fingas. Chapter 15 - oil spill dispersants: A technical summary. In Mervin Fingas, editor, *Oil Spill Sci. Technol.*, pages 435–582. Gulf Professional Publishing, Boston, 2011. ISBN 978-1-85617-943-0. doi: 10.1016/B978-1-85617-943-0.10015-2.

[56] C Flament, S Lacis, A Cebers, S Neveu, and R Perzynski. Measurement of ferrofluid surface tension in confined geometry. *Phys. Rev. E*, 53(5):4801–4806, 1996.

[57] Y Fujimura and M Iino. Magnetic field increases the surface tension of water. *J. Phys. Conf. Ser.*, 156:012028, 2009. ISSN 1742-6596. doi: 10.1088/1742-6596/156/1/012028.

[58] Anilkumar G Gaonkar. Effects of salt, temperature, and surfactants on the interfacial tension behavior of a vegetable oil/water system. *J. Colloid Interface Sci.*, 149(1):256–260, March 1992. ISSN 00219797. doi: 10.1016/0021-9797(92)90412-F.

[59] Abel Garcia-Saiz, Pedro Migowski, Oriol Vallcorba, Javier Junquera, Jesús Angel Blanco, Jesús Antonio González, María Teresa Fernández-Díaz, Jordi Rius, Jairton Dupont, Jesús Rodríguez Fernández, and Imanol de Pedro. A magnetic ionic liquid based on tetrachloroferrate exhibits three-dimensional magnetic ordering: A combined experimental and theoretical study of the magnetic interaction mechanism. *Chem.-A Eur. J.*, 20(1):72–76, 2014. ISSN 1521-3765. doi: 10.1002/chem.201303602.

[60] A Goebel and K Lunkenheimer. Interfacial tension of the water/n-alkane interface. *Langmuir*, 13(2):369–372, 1997. doi: 10.1021/la960800g.

[61] Halil Gökce and Semiha Bahçeli. The molecular structures, vibrational spectroscopies (FT-IR and Raman) and quantum chemical calculations of n-alkyltrimethylammonium bromides. *Opt. Spectrosc.*, 115(5):632–644, 2013. ISSN 0030-400X. doi: 10.1134/S0030400X13110076.

[62] Alexandre Goldszal and Maurice Bourrel. Demulsification of crude oil emulsions: Correlation to microemulsion phase behavior. *Ind. Eng. Chem. Res.*, 39 (8):2746–2751, 2000. ISSN 0888-5885. doi: 10.1021/ie990922e.

[63] Yongan Gu, Dongqing Li, and P. Cheng. A novel contact angle measurement technique by analysis of capillary rise profile around a cylinder (ACRPAC). *Colloids Surfaces A Physicochem. Eng. Asp.*, 122(1-3):135–149, 1997. ISSN 09277757. doi: 10.1016/S0927-7757(96)03853-8.

[64] E A Guggenheim. *Modern Thermodynamics by the Methods of Willard Gibbs*. Methuen & Company Limited, 1933.

[65] S Hartland. The coalescence of a liquid drop at a liquid-liquid interface. Part I: Drop shape. *Trans. Instn Chem Engrs*, 45:T97–T101, 1967.

[66] W.M. Haynes, editor. *CRC Handbook of Chemistry and Physics*. CRC Press Inc, Boca Raton, FL, 91 edition, 2011.

[67] W.M. Haynes, editor. *CRC Handbook of Chemistry and Physics*. CRC Press LLC, Boca Raton, 95 edition, 2015.

[68] P M Heertjes, E C de Smet, and W C Witvoet. The determination of interfacial tensions with the Wilhelmy plate method. *Chem. Eng. Sci.*, 26(9):1479–1480, 1971. doi: 10.1016/0009-2509(71)80068-4.

[69] D M Himmelblau and J B Riggs. *Basic Principles and Calculations in Chemical Engineering*. Prentice Hall International Series in the Physical and Chemical Engineering Sciences. Pearson Education, 7 edition, 2012. ISBN 9780132901086.

[70] J.A Holgado-Terriza, J.F Gómez-Lopera, P.L Luque-Escamilla, C Atae-Allah, and M.A Cabrerizo-Vílchez. Measurement of ultralow interfacial tension with ADSA using an entropic edge-detector. *Colloids Surfaces A Physicochem. Eng. Asp.*, 156(1-3):579–586, October 1999. ISSN 09277757. doi: 10.1016/S0927-7757(99)00122-3.

[71] Krister Holmberg, editor. *Novel Surfactants: Preparation, applications and biodegradability*. Marcel Dekker Inc., 2003.

[72] M Hoorfar and August Neumann. Recent progress in axisymmetric drop shape analysis (ADSA). *Adv. Colloid Interface Sci.*, 121(1-3):25–49, 2006. doi: 10.1016/j.cis.2006.06.001.

[73] M Hoorfar, M A Kurz, and A W Neumann. Evaluation of the surface tension measurement of axisymmetric drop shape analysis (ADSA) using a shape parameter. *Colloids Surfaces A Physicochem. Eng. Asp.*, 260(1-3):277–285, 2005. doi: 10.1016/j.colsurfa.2004.08.080.

[74] Guoyou Huang, Moxiao Li, Qingzhen Yang, Yuhui Li, Hao Liu, Hui Yang, and Feng Xu. Magnetically actuated droplet manipulation and its potential biomedical applications. *ACS Appl. Mater. Interfaces*, 9(2):1155–1166, 2017. ISSN 1944-8244. doi: 10.1021/acsami.6b09017.

[75] Chun Huh and S G Mason. The flotation of axisymmetric particles at horizontal liquid linterfaces. *J. Colloid Interface Sci.*, 47(2):271–289, 1973.

[76] Chun Huh and L E Scriven. Shapes of axisymmetric fluid interfaces of unbounded extent. *J. Colloid Interface Sci.*, 30(3):232–337, 1969.

[77] A. Hyde, C. Phan, and S. Yusa. Dynamic interfacial tension of nonanoic acid/-hexadecane/water system in response to ph adjustment. ([Submitted]), 2017.

[78] Anita Hyde, Chi Phan, and Gordon Ingram. Determining liquid–liquid interfacial tension from a submerged meniscus. *Colloids Surfaces A Physicochem. Eng. Asp.*, 459(1):267–273, October 2014. ISSN 0927-7757. doi: 10.1016/j.colsurfa.2014.07.016.

[79] Anita Hyde, Masahiro Horiguchi, Naoya Minamishima, Yusuke Asakuma, and Chi Phan. Effects of microwave irradiation on the decane-water interface in the presence of triton x-100. *Colloids Surfaces A Physicochem. Eng. Asp.*, 524 (April):178–184, 2017. ISSN 09277757. doi: 10.1016/j.colsurfa.2017.04.044.

[80] Harley Y Jennings. The effect of temperature and pressure on the interfacial tension of benzene-water and normal decane-water. *J. Colloid Interface Sci.*, 24(3):323–329, July 1967. ISSN 00219797. doi: 10.1016/0021-9797(67)90257-3.

[81] Harley Y Jennings Jr. and George H Newman. The effect of temperature and pressure on the interfacial tension of water against methane-normal decane mixtures. *Soc. Pet. Eng. 45th Annu. Fall Meet.*, 1971. doi: 10.2118/3071-PA.

[82] A Kalantarian. *Development of axisymmetrix drop shape analysis - no apex (ADSA-NA).* PhD thesis, 2011.

[83] A. Kalantarian, R. David, and A. W. Neumann. Methodology for high accuracy contact angle measurement. *Langmuir*, 25(24):14146–14154, 2009. ISSN 07437463. doi: 10.1021/la902016j.

[84] Ali Kalantarian, Sameh M I Saad, and a. Wilhelm Neumann. Accuracy of surface tension measurement from drop shapes: The role of image analysis. *Adv. Colloid Interface Sci.*, 199-200:15–22, 2013. ISSN 00018686. doi: 10.1016/j.cis.2013.07.004.

[85] Wanli Kang, Bin Xu, Yongjian Wang, Yuan Li, Xiuhua Shan, Faquan An, and Jiaheng Liu. Stability mechanism of w/o crude oil emulsion stabilized by polymer and surfactant. *Colloids Surfaces A Physicochem. Eng. Asp.*, 384(1-3): 555–560, 2011. doi: 10.1016/j.colsurfa.2011.05.017.

[86] Wimpy Karnanda, M. S. Benzagouta, Abdulrahman AlQuraishi, and M. M. Amro. Effect of temperature, pressure, salinity, and surfactant concentration on ift for surfactant flooding optimization. *Arab. J. Geosci.*, 6(9):3535–3544, 2013. ISSN 18667538. doi: 10.1007/s12517-012-0605-7.

[87] Vikrant Khanna, Phalguni Gupta, and C.J. Hwang. Finding connected components in digital images by aggressive reuse of labels. *Image Vis. Comput.*, 20 (8):557–568, June 2002. ISSN 02628856. doi: 10.1016/S0262-8856(02)00044-6.

[88] Sanghoon Kim, Christine Bellouard, Andreea Pasc, Emmanuel Lamouroux, Jean-Luc Blin, Cedric Carteret, Yves Fort, Melanie Emo, Pierrick Durand, and Marie-Jose Stebe. Nanoparticle-free magnetic mesoporous silica with magneto-responsive surfactants. *J. Mater. Chem. C*, 1(42):6930–6934, 2013. ISSN 2050-7526. doi: 10.1039/c3tc31617c.

[89] F Kreith. *The CRC Handbook of Thermal Engineering*. Mechanical Engineering. Springer Berlin Heidelberg, 2000. ISBN 9783540663492.

[90] B Lautrup. Physics of continuous matter - exotic and everyday phenomena in the macroscopic world, 2005.

[91] Pei Xun Li, Robert K. Thomas, and Jeffrey Penfold. Limitations in the use of surface tension and the Gibbs equation to determine surface excesses of cationic surfactants. *Langmuir*, 30(23):6739–6747, 2014. ISSN 15205827. doi: 10.1021/la501287v.

[92] Jiling Liang, Na Du, Shue Song, and Wanguo Hou. Magnetic demulsification of diluted crude oil-in-water nanoemulsions using oleic acid-coated magnetite nanoparticles. *Colloids Surfaces A Physicochem. Eng. Asp.*, 466:197–202, 2015. ISSN 18734359. doi: 10.1016/j.colsurfa.2014.11.050.

[93] W C Lyons and B S Gary J Plisga. *Standard Handbook of Petroleum and Natural Gas Engineering*. Elsevier Science, 2011. ISBN 9780080481081.

[94] Andrew D. Malec, Deng Guo Wu, Mary Louie, Janusz J. Skolimowski, and Marcin Majda. Gibbs monolayers at the air/water interface: Surface partitioning and lateral mobility of an electrochemically active surfactant. *Langmuir*, 20 (4):1305–1310, 2004. ISSN 07437463. doi: 10.1021/la035754g.

[95] R Massoudi and A D King. Effect of pressure on the surface tension of aqueous solutions. adsorption of hydrocarbon gases, carbon dioxide, and nitrous oxide on aqueous solutions of sodium chloride and tetrabutylammonium bromide at 25 degrees. *J. Phys. Chem.*, 79:1670–1675, 1975. ISSN 0022-3654. doi: 10.1021/j100583a012.

[96] Curtis W. McDonald and Olubunmi A. Ogunkeye. Ion flotation of zinc using ethylhexadecyldimethylammonium bromide. *Microchem. J.*, 26(1):80–85, 1981. ISSN 0026265X. doi: 10.1016/0026-265X(81)90013-8.

[97] D Mecerreyes. *Applications of Ionic Liquids in Polymer Science and Technology.* Springer Berlin Heidelberg, 2015. ISBN 9783662449035.

[98] A. Miquilena, V. Coll, A. Borges, J. Melendez, and S. Zeppieri. Influence of drop growth rate and size on the interfacial tension of Triton X-100 solutions as a function of pressure and temperature. *Int. J. Thermophys.*, 31(11-12): 2416–2424, 2010. ISSN 0195928X. doi: 10.1007/s10765-010-0825-6.

[99] Indrajyoti Mukherjee, Satya P. Moulik, and Animesh K. Rakshit. Tensiometric determination of Gibbs surface excess and micelle point: A critical revisit. *J. Colloid Interface Sci.*, 394(1):329–336, 2013. ISSN 00219797. doi: 10.1016/j.jcis.2012.12.004.

[100] Karol J. Mysels. The maximum bubble pressure method of measuring surface tension, revisited. *Colloids and Surfaces*, 43(2):241–262, 1990. ISSN 01666622. doi: 10.1016/0166-6622(90)80291-B.

[101] Michael J Neeson, Rico F Tabor, Franz Grieser, Raymond R Dagastine, and Derek Y C Chan. Compound sessile drops. *Soft Matter*, 8(43):11042–11050, 2012. doi: 10.1039/C2SM26637G.

[102] Michael J. Neeson, Derek Y C Chan, and Rico F. Tabor. Compound pendant drop tensiometry for interfacial tension measurement at zero Bond number. *Langmuir*, 30(51):15388–15391, 2014. ISSN 15205827. doi: 10.1021/la504406m.

[103] M Nuechter, B Ondruschka, W Bonrath, and A Gum. Microwave assisted synthesis - a critical technology overview. *Green Chem.*, 6(3):128–141, 2004. ISSN 1463-9262. doi: 10.1039/b310502d.

[104] T. Ody, M. Panth, A. D. Sommers, and K. F. Eid. Controlling the motion of ferrofluid droplets using surface tension gradients and magnetoviscous pinning. *Langmuir*, 32(27):6967–6976, 2016. ISSN 15205827. doi: 10.1021/acs.langmuir.6b01030.

[105] Yuki Ohara, Yuuki Kawata, Anita Hyde, Chi Phan, Ryoji Takeda, Yasushi Takemura, and Shin-ichi Yusa. Preparation of a magnetic-responsive polycation with a tetrachloroferrate anion. *Chem. Lett.*, 46(10):1473–1475, 2017. ISSN 0366-7022. doi: 10.1246/cl.170621.

[106] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Trans Syst Man Cybern*, SMC-9(1):62–66, 1979. ISSN 00189472.

[107] J F Padday and A Pitt. Axisymmetric meniscus profiles. *J. Colloid Interface Sci.*, 38(2):323–334, 1972. ISSN 0021-9797. doi: https://doi.org/10.1016/0021-9797(72)90249-4.

[108] J.F Padday and D.R Russell. The measurement of the surface tension of pure liquids and solutions. *J. Colloid Sci.*, 15(6):503–511, 1960. ISSN 00958522. doi: 10.1016/0095-8522(60)90054-4.

[109] Harisinh Parmar, Masahiro Asada, Yushin Kanazawa, Yusuke Asakuma, Chi M Phan, Vishnu Pareek, and Geoffrey M Evans. Influence of microwaves on the water surface tension. *Langmuir*, 30(33):9875–9879, 2014. doi: dx.doi.org/10.1021/la5019218|.

[110] Paul Brown, Alexey Bushmelev, Craig P. Butts, Jing Cheng, Julian Eastoe, Isabelle Grillo, Richard K. Heenan, and Annette M. Schmidt. Magnetic control over liquid surface properties with responsive surfactants. *Angewandte*, 51 (10):2414–2416, 2012. doi: 10.1002/anie.201108010.

[111] I De Pedro, Dp Rojas, and J Albo. Long-range magnetic ordering in magnetic ionic liquid: Emim. *J. Phys.: Condensed Matter*, 22(29):296006, 2010. ISSN 0953-8984. doi: 10.1088/0953-8984/22/29/296006.

[112] F Peters and D Arabali. Interfacial tension between oil and water measured with a modified contour method. *Colloids Surfaces A Physicochem. Eng. Asp.*, 426(0):1–5, 2013. doi: 10.1016/j.colsurfa.2013.03.010.

[113] Chi M. Phan, Thu N. Le, and Shin-ichi Yusa. A new and consistent model for dynamic adsorption of CTAB at air/water interface. *Colloids Surfaces A Physicochem. Eng. Asp.*, 406:24–30, July 2012. ISSN 18734359. doi: 10.1016/j.colsurfa.2012.04.044.

[114] Chi M Phan, Thu N Le, Cuong V Nguyen, and Shin-ichi Yusa. Modeling adsorption of cationic surfactants at air/water interface without using the Gibbs equation. *Langmuir*, 29:4743–4749, 2013. doi: 10.1021/la3046302.

[115] J. M. S. Prewitt. Object enhancement and extraction. In B. Lipkin and A. Rosenfeld, editors, *Pict. Process. Psychopictorics*, pages 75–149. Academic Press, New York, 1970.

[116] H.M Princen. Shape of a fluid drop at a liquid-liquid interface. *J. Colloid Sci.*, 18:178–195, 1963. ISSN 00958522. doi: 10.1016/0095-8522(63)90008-4.

[117] H.M Princen, I.Y.Z Zia, and S.G Mason. Measurement of interfacial tension from the shape of a rotating drop. *J. Colloid Interface Sci.*, 23(1):99–107, January 1967. ISSN 00219797. doi: 10.1016/0021-9797(67)90090-2.

[118] R M Prokop, A Jyoti, M Eslamian, A Garg, M Mihaila, O I del Río, S S Susnar, Z Policova, and A W Neumann. A study of captive bubbles with axisymmetric drop shape analysis. *Colloids Surfaces A Physicochem. Eng. Asp.*, 131(1-3):231–247, 1998. doi: 10.1016/S0927-7757(96)03940-4.

[119] T.S. Ramakrishnan and D.T. Wasan. A model for interfacial activity of acidic crude oil/caustic systems for alkaline flooding. *Soc. Pet. Eng. J.*, 23(04):602–612, 1983. ISSN 0197-7520. doi: 10.2118/10716-PA.

[120] Hermann Rau, Gerhard Greiner, and Hugo Hämmerle. Temperature dependence of number and size of Triton X-100 micelles in aqueous solution. *Berichte der Bunsengesellschaft für Phys. Chemie*, 88(2):116–121, 1984. ISSN 00059021. doi: 10.1002/bbpc.19840880208.

[121] J.A. Riddick, W.B. Bunger, and Sakano T.K. *Techniques of Chemistry*, volume II. John Wiley & Sons, New York, 4th edition, 1985.

[122] Milton J Rosen and Joy T Kunjappu. *Surfactants and Interfacial Phenomena.* John Wiley & Sons, Inc., 4th edition, 2012. ISBN 9781118228920. doi: 10.1002/9781118228920.fmatter.

[123] R E Rosensweig. *Ferrohydrodynamics.* Dover Books on Physics. Dover Publications, 2013. ISBN 9780486783000.

[124] Y Rotenberg, L Boruvka, and a.W W Neumann. Determination of surface tension and contact angle from the shapes of axisymmetric fluid interfaces. *J. Colloid Interface Sci.*, 93(1):169–183, 1983. ISSN 00219797. doi: 10.1016/0021-9797(83)90396-X.

[125] E Ruckenstein and I.V Rao. Interfacial tension of oil-brine systems in the presence of surfactant and cosurfactant. *J. Colloid Interface Sci.*, 117(1):104–119, May 1987. ISSN 00219797. doi: 10.1016/0021-9797(87)90173-1.

[126] Sameh M I Saad and A Wilhelm Neumann. Total Gaussian curvature, drop shapes and the range of applicability of drop shape techniques. *Adv. Colloid Interface Sci.*, 204(0):1–14, 2014. doi: 10.1016/j.cis.2013.12.001.

[127] Sameh M I Saad, Zdenka Policova, Edgar J Acosta, and A Wilhelm Neumann. Axisymmetric drop shape analysis–constrained sessile drop (ADSA-CSD): A film balance technique for high collapse pressures. *Langmuir*, 24(19):10843–10850, 2008. doi: 10.1021/la801683q.

[128] Sameh M I Saad, Zdenka Policova, and A Wilhelm Neumann. Design and accuracy of pendant drop methods for surface tension measurement. *Colloids Surfaces A Physicochem. Eng. Asp.*, 384(1-3):442–452, 2011. doi: 10.1016/j.colsurfa.2011.05.002.

[129] Sameh M.I. Saad and A. Wilhelm Neumann. Axisymmetric drop shape analysis (ADSA): An outline. *Adv. Colloid Interface Sci.*, 238:62–87, 2016. ISSN 00018686. doi: 10.1016/j.cis.2016.11.001.

[130] Antonio Manzolillo Sanseverino. Microondas em síntese orgânica. *Quim. Nova*, 25(4):660–667, 2002. ISSN 01004042. doi: 10.1590/S0100-40422002000400022.

[131] E Schmidt and U Grigull. *Properties of water and steam in SI-units.* Springer, 1981. ISBN 9780387046761.

[132] L L Schramm. *Surfactants: Fundamentals and Applications in the Petroleum Industry.* Surfactants: Fundamentals and Applications in the Petroleum Industry. Cambridge University Press, 2000. ISBN 9780521640671.

[133] James R T Seddon, Detlef Lohse, William A. Ducker, and Vincent S J Craig. A deliberation on nanobubbles at surfaces and in bulk. *ChemPhysChem*, 13(8): 2179–2187, 2012. ISSN 14394235. doi: 10.1002/cphc.201100900.

[134] Andreas M Seifert. Spinning drop tensiometry. In D Mobius and R Miller, editors, *Stud. Interface Sci.*, volume Volume 6, pages 187–238. Elsevier, 1998. ISBN 1383-7303. doi: 10.1016/S1383-7303(98)80021-5.

[135] M M Sharma, L K Jang, and T F Yen. Transient interfacial tension behavior of crude-oil / caustic interfaces. *SPE Reserv. Eng.*, 4(2):228–236, 1989. doi: 10.2118/12669-PA.

[136] Mukul M Sharma and T F Yen. A thermodynamic model for low interfacial tensions in alkaline flooding. *Soc. Pet. Eng. J.*, 23(1):125–134, 1983. ISSN 01977520. doi: 10590-PA.

[137] F a Siddiqui and E I Franses. Surface tension and adsorption synergism for solutions of binary surfactants. *Ind Eng Chem Res*, 35(9):3223–3232, 1996. ISSN 0888-5885. doi: 10.1021/ie960044+.

[138] I M N Smallwood. *Solvent Recovery Handbook*. Blackwell Science, 2nd edition, 2002. ISBN 9780849316029.

[139] Stephen M Smith and J Michael Brady. Susan–a new approach to low level image processing. *Int. J. Comput. Vis.*, 23(1):45–78, 1997. ISSN 1573-1405. doi: 10.1023/A:1007963824710.

[140] Aurélien F Stalder, Tobias Melchior, Michael Müller, Daniel Sage, Thierry Blu, and Michael Unser. Low-bond axisymmetric drop shape analysis for surface tension and contact angle measurements of sessile drops. *Colloids Surfaces A Physicochem. Eng. Asp.*, 364(1-3):72–81, 2010. doi: 10.1016/j.colsurfa.2010.04.040.

[141] H. a. Stone, J. R. Lister, and M. P. Brenner. Drops with conical ends in electric and magnetic fields. *Proc. R. Soc. A Math. Phys. Eng. Sci.*, 455(1981):329–347, 1999. ISSN 1364-5021. doi: 10.1098/rspa.1999.0316.

[142] Kiril Streletzky and George D J Phillies. Temperature dependence of triton x-100 micelle size and hydration. *Langmuir*, 11(1):42–47, 1995. doi: 10.1021/la00001a011.

[143] Reinhard Strey, Yrjo Viisanen, Makoto Aratono, Josip P. Kratohvil, Qi Yin, and Stig E. Friberg. On the necessity of using activities in the gibbs equation. *J. Phys. Chem. B*, 103(43):9112–9116, 1999. ISSN 1520-6106. doi: 10.1021/jp990306w.

[144] Didier Stuerga. Microwave-materials interactions and dielectric properties: From molecules and macromolecules to solids and colloidal suspensions. In *Microwaves Org. Synth.*, chapter 1, pages 1–56. Wiley-VCH Verlag GmbH & Co. KGaA, 2012. ISBN 9783527651313. doi: 10.1002/9783527651313.ch1.

[145] S.S. S Susnar, H.A. A Hamza, and A.W. W Neumann. Pressure dependence of interfacial tension of hydrocarbon-water systems using axisymmetric drop shape analysis. *Colloids and Surfaces*, 89(2-3):169–180, September 1994. ISSN 09277757. doi: 10.1016/0927-7757(94)80116-9.

[146] Motohiko Tanaka and Motoyasu Sato. Microwave heating of water, ice, and saline solution: molecular dynamics study. *J. Chem. Phys.*, 126(3):034509, 2007. ISSN 0021-9606. doi: 10.1063/1.2403870.

[147] M trujillo Edward and SPE Marathon oil Co. The static and dynamic interfacial tension between crude oil and caustic solutions. *Soc. Pet. Eng. J.*, 23 (August):647–657, 1983. ISSN 0197-7520.

[148] P J Tummino and A Gafni. Determination of the aggregation number of detergent micelles using steady-state fluorescence quenching. *Biophys. J.*, 64(5): 1580–1587, 1993. ISSN 00063495. doi: 10.1016/S0006-3495(93)81528-5.

[149] Josefina Viades-Trejo and Jesús Gracia-Fadrique. Spinning drop method: From Young-Laplace to Vonnegut. *Colloids Surfaces A Physicochem. Eng. Asp.*, 302(1-3):549–552, 2007. doi: 10.1016/j.colsurfa.2007.03.033.

[150] J. K. Vij. Microwave and far-infra-red dielectric absorption in n-alkanes. *Nuovo Cim. D*, 2(3):751–762, 1983. ISSN 03926737. doi: 10.1007/BF02453215.

[151] A. F. H. Ward and L. Tordai. Time-dependence of boundary tensions of solutions i. the role of diffusion in time-effects. *J. Chem. Phys.*, 14(7):453–461, 1946. ISSN 0021-9606. doi: 10.1063/1.1724167.

[152] Ludwig Wilhelmy. Ueber die abhaenigkeit der capillaritaets-constanten des akohols von substanz und gestalt des benetzten festen koerpers. *Ann. der Phys. und Chemie*, 119(6):177–217, 1863.

[153] Qian Zhou, William D. Ristenpart, and Pieter Stroeve. Magnetically induced decrease in droplet contact angle on nanostructured surfaces. *Langmuir*, 27 (19):11747–11751, 2011. ISSN 07437463. doi: 10.1021/la2024633.

[154] A V Zhukov. Structure of the fluid interface in an external magnetic field in the presence of a magnetizable surfactant. *Fluid Dyn.*, 51(4):463–468, 2016.

[155] Y Y Zuo, M Ding, A Bateni, M Hoorfar, and A W Neumann. Improvement of interfacial tension measurement using a captive bubble in conjunction with axisymmetric drop shape analysis (ADSA). *Colloids Surfaces A Physicochem. Eng. Asp.*, 250(1-3):233–246, 2004. doi: 10.1016/j.colsurfa.2004.04.081.

[156] Yi Y Zuo, Chau Do, and A Wilhelm Neumann. Automatic measurement of surface tension from noisy images using a component labeling method. *Colloids Surfaces A Physicochem. Eng. Asp.*, 299(1-3):109–116, 2007. doi: 10.1016/j.colsurfa.2006.11.027.

Every reasonable effort has been made to acknowledge the owners of copywrite material. I would be pleased to hear from any copywrite owner who has been omitted or incorrectly acknowledged.

In-situ Investigation of the Oil-Water Interface Under Dynamic Conditions

Anita Hyde

School of Chemical and Petroleum Engineering

Extended appendices
Presented for the degree of
Doctor of Philosophy
of
Curtin University

January 2018

Contents

A Overview of code structure **3**

B Matlab code **4**

B.1 Holm main GUI (controller GUI) 4
B.2 Holm_Main (back-end analysis) 90
B.3 ThreshGUI (interactive thresholding and edge identification) 102
B.4 AngleGUI (automated angle adjustment) 134
B.5 Timeline (formatting of temperature files) 156
B.6 SaveOutputGUI (export output `.txt` file to Excel template) 175

C Secondary filtering in Excel **179**

C.1 Excel template 179
C.2 First-pass filtering 179
C.3 Second-pass filtering 184
C.4 Additional formatting 189

Listings

B.1 HolmGUImainV11d.m 4
B.2 Holm_MAIN_3.m 90
B.3 GUIthreshHv11c.m 102
B.4 HolmGUIangle12b.m 134
B.5 Timeline10.m 156
B.6 SaveOutputGUI.m 175
C.1 VBA code for first-pass filtering. 179
C.2 VBA code for second-pass filtering. 184
C.3 VBA code for additional formatting. 189

Appendix A

Overview of code structure

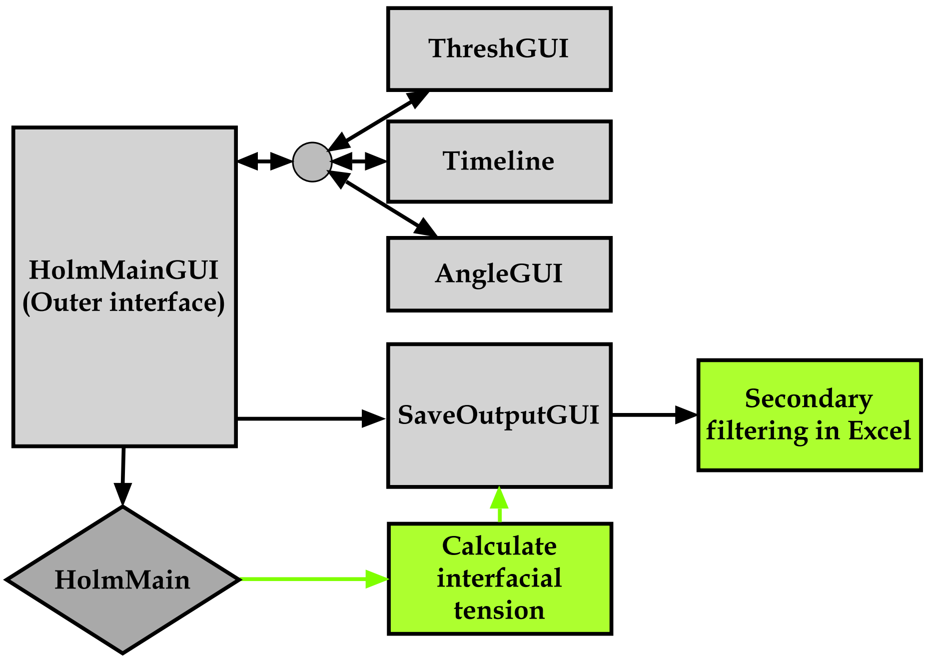

Figure A.1: Main program GUI.

Access to all of the GUIs is through the MainGUI.

Appendix B

Matlab code

B.1 Holm main GUI (controller GUI)

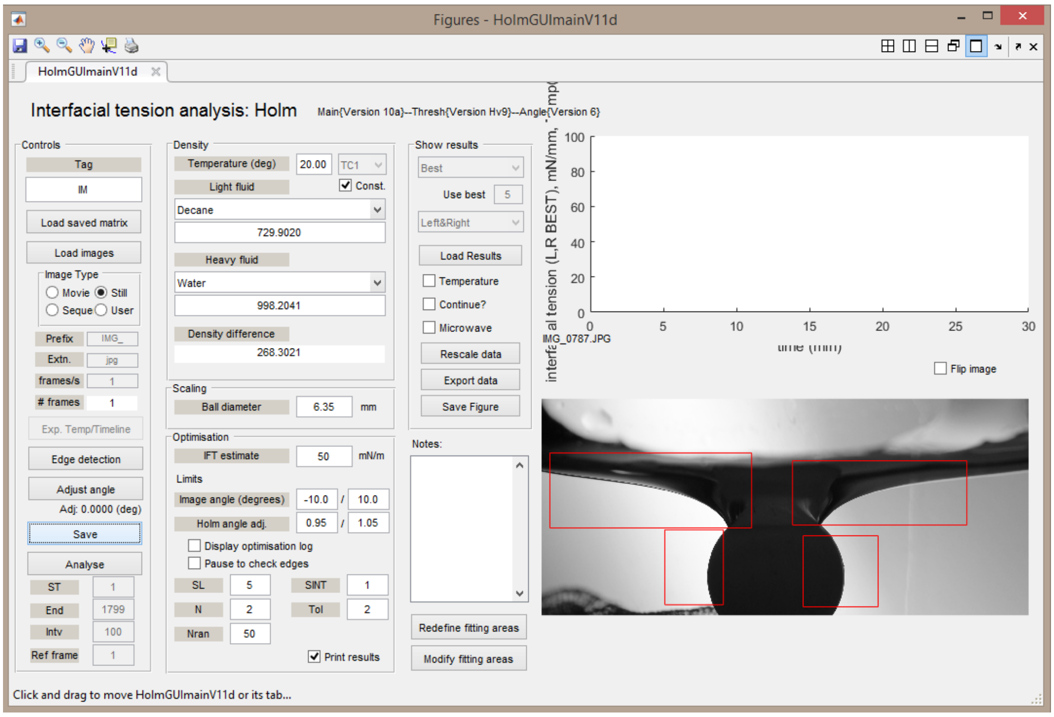

Figure B.1: Main program GUI.

Listing B.1: HolmGUImainV11d.m

```
function varargout = HolmGUImainV11d(varargin)
% HOLMGUIMAINV9 MATLAB code for HolmGUImainV9.fig
%——————————————————————————————————————
%      HOLMGUIMAINV9, by itself, creates a new HOLMGUIMAINV9 or raises the
```

```
%       existing-````````````````````````````````````````
%       singleton*.
%
%      H = HOLMGUIMAINV9 returns the handle to a new HOLMGUIMAINV9 or the handle to
%       the existing singleton*.
%
%--------------------------------------------------------------------
%-- HolmGUImainV* is the main GUI code to run ADSA-style analysis using the
%holm meridian.
%--------------------------------------------------------------------
%
% See also: Holm_Main*, GUIthreshHv*, Holm_2sides*, HolmGUIangle*, Timeline


% Last Modified by GUIDE v2.5 25-May-2017 23:23:19

%--------------------------------------------------------------------
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @HolmGUImainV9_OpeningFcn, ...
                   'gui_OutputFcn',  @HolmGUImainV9_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
%--------------------------------------------------------------------


% --- Executes just before HolmGUImainV9 is made visible.
function HolmGUImainV9_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
%--------------------------------------------------------------------
% Choose default command line output for HolmGUImainV9
handles.output = hObject;

```

```
% Update handles structure
guidata(hObject, handles);
%--------------------------------------------------------------------------
%.. update version information
    set(handles.ST_Version,'String','Main{Version 10a}--Thresh{Version Hv9}--Angle{Version 6}');%%%VERSION INFO
%.. Save handles data to figure
    GUI_Hmain=gcf;
    setappdata(0,'GUI_Hmain',GUI_Hmain);
    setappdata(GUI_Hmain,'hImMask',@ImMask);
    setappdata(GUI_Hmain,'hWIM',@WholeImageMask);
    setappdata(GUI_Hmain,'hTri',@Tri);
    setappdata(GUI_Hmain,'hTriSphere',@TriSphere);
    setappdata(GUI_Hmain,'hRotateCoords',@RotateCoords);
    %setappdata(GUI_Hmain,'hOrderCoord',@OrderCoord);
    set(0,'DefaultFigureWindowStyle','docked')%dock all figures this session

%.. Suppress warnings:
    warning('off','images:initSize:adjustingMag')%image too large for screen - will resize
    warning('off','images:imshow:magnificationMustBeFitForDockedFigure')%image docked
    warning('off','MATLAB:colon:nonIntegerIndex')%"Integer operands are required for colon operator when used as index"
    warning('off','MATLAB:polyfit:RepeatedPointsOrRescale');%interpolating on holmcoord-not smooth function..will give warning
    warning('off','MATLAB:hg:uicontrol:ParameterValuesMustBeValid');%suppress slider warning until initialisation is finished

%.. Set angles
    handles.Alpha=0;
    handles.Theta=0;

%.. Update handles structure
    guidata(hObject, handles);
%--------------------------------------------------------------------------

% --- Outputs from this function are returned to the command line.
function varargout = HolmGUImainV9_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```


% --- Executes on selection change in DD_LightFluid.
function DD_LightFluid_Callback(hObject, eventdata, handles)
%------------------------------------------------------------------
%--- Select fluid type, calculate density
%.. Get fluid type
contents = cellstr(get(hObject,'String')); %returns DD_LightFluid contents as cell array
light=contents{get(hObject,'Value')}; %returns selected item from DD_LightFluid

%.. Get temperature
T=str2double(get(handles.ET_Temp,'String'));

%.. Calculate denstiy
switch light
    case 'Decane'
        RhoL=228.2*0.247^(-(1-(T+273.13)/616)^(2/7));%from Himmenbleau and Riggs (original gives g/cm3, *1000
    case 'Dodecane'
        msgbox('not done yet')
    case 'Air'
        msgbox('not done yet. using 1.2')
        RhoL=1.2;
    case 'Water'
        RhoL=(999.83952+T*(16.945176+T*(-7.9870401e-3+T*(-46.170461e-6+T*(105.56302e-9-280.54253e-12*T)))))/(1+T*16.87985e-3);%SysCad steam properties: http://help.syscad.net/index.php/Water_and_Steam_Properties
    case 'Other'
        RhoL=inputdlg('Enter value manually (light fluid density).');
        RhoL=str2double(RhoL);
end

%.. Update static density
if exist('RhoL','var')==1
    handles.RhoL=RhoL;
    set(handles.ET_StaticLight,'String',sprintf('%0.4f',RhoL));

    %.. Density difference
    handles.dRho=handles.RhoH-RhoL;
    set(handles.ST_dRho,'String',sprintf('%0.4f',handles.dRho));
    handles.light=light;
end
guidata(hObject, handles);
%------------------------------------------------------------------
```

```matlab


% --- Executes during object creation, after setting all properties.
function DD_LightFluid_CreateFcn(hObject, eventdata, handles)
%--------------------------------------------------------------
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%--------------------------------------------------------------
%--- LIGHT FLUID DEFAULT
%--------------------------------------------------------------
 handles.light='Decane';
 guidata(hObject, handles);
%--------------------------------------------------------------


function ET_StaticLight_Callback(hObject, eventdata, handles)
%--------------------------------------------------------------
%--- INPUT LIGHT DENSITY MANUALLY
%--------------------------------------------------------------
handles.RhoL=str2double(get(hObject,'String'));% returns contents of ET_StaticLight as a double

%.. Update dRho
handles.dRho = handles.RhoH - handles.RhoL;
set(handles.ST_dRho,'String',sprintf('%0.4f',handles.dRho));

guidata(hObject, handles);
%--------------------------------------------------------------


% --- Executes during object creation, after setting all properties.
function ET_StaticLight_CreateFcn(hObject, eventdata, handles)
%--------------------------------------------------------------
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%--- SET DEFAULT LIGHT DENSITY
%.. Use default temperature
T=20;

%.. Calculate denstiy (decane)
handles.RhoL=228.2*0.247^(-(1-(T+273.13)/616)^(2/7));%from Himmenbleau and Riggs (original gives g/cm3, *1000
```

```
set(hObject,'String',sprintf('%0.4f',handles.RhoL));

guidata(hObject, handles);
%------------------------------------------------------------



% --- Executes on selection change in DD_DenseFluid.
function DD_DenseFluid_Callback(hObject, eventdata, handles)
 %------------------------------------------------------------
%--- Select fluid type, calculate density
%.. Get fluid type
contents = cellstr(get(hObject,'String')); %returns DD_LightFluid contents as cell
     array
dense=contents{get(hObject,'Value')}; %returns selected item from DD_LightFluid

%.. Get temperature
T=str2double(get(handles.ET_Temp,'String'));

%.. Calculate denstiy
switch dense
    case 'Water' %pure water
        RhoH=(999.83952+T*(16.945176+T*(-7.9870401e-3+T*(-46.170461e-6+T
            *(105.56302e-9-280.54253e-12*T)))))/(1+T*16.87985e-3);%SysCad steam
            properties: http://help.syscad.net/index.php/
            Water_and_Steam_Properties
    case 'Brine-0.1 NaCl' %0.1 mol/l
        RhoH=(-5.3478e-6*T^2+4.9629e-6*T+1.0048)*1000;
    case 'Brine-0.01 NaCl'%0.01mol/l
        RhoH=(-5.5488e-6*T^2+2.0921e-5*T+1.001)*1000;
    case 'Brine-0.001 NaCl'%0.001mol/l
        RhoH=(-5.5692e-6*T^2+2.2535e-5*T+1.000)*1000;
    case 'Other'
        RhoH=inputdlg('Enter value manually (heavy fluid density).');
        RhoH=str2double(RhoH);
end

%.. Update static density
if exist('RhoH','var')==1
    handles.RhoH=RhoH;
    set(handles.ET_StaticDense,'String',sprintf('%0.4f',RhoH));
    handles.dRho = handles.RhoH - handles.RhoL;
    set(handles.ST_dRho,'String',sprintf('%0.4f',handles.dRho));
    handles.dense=dense;
end

```

```
guidata(hObject, handles);
%----------------------------------------------------------------


% --- Executes during object creation, after setting all properties.
function DD_DenseFluid_CreateFcn(hObject, eventdata, handles)
%----------------------------------------------------------------
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%----------------------------------------------------------------
%--- DENSE FLUID DEFAULT
%----------------------------------------------------------------
 handles.dense='Water';
 guidata(hObject, handles);
%----------------------------------------------------------------




function ET_StaticDense_Callback(hObject, eventdata, handles)
%----------------------------------------------------------------
%--- INPUT LIGHT DENSITY MANUALLY
%----------------------------------------------------------------
handles.RhoH=str2double(get(hObject,'String'));% returns contents of ET_StaticLight as a double

%.. Update dRho
handles.dRho = handles.RhoH - handles.RhoL;
set(handles.ST_dRho,'String',sprintf('0.4%f',handles.dRho));

guidata(hObject, handles);
%----------------------------------------------------------------


% --- Executes during object creation, after setting all properties.
function ET_StaticDense_CreateFcn(hObject, eventdata, handles)
%----------------------------------------------------------------
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%----------------------------------------------------------------
%--- SET DEFAULT HEAVY DENSITY
%----------------------------------------------------------------
```

```
%.. Use default temperature
T=20;

%.. Calculate denstiy (decane)
handles.RhoH=(999.83952+T*(16.945176+T*(-7.9870401e-3+T*(-46.170461e-6+T*(105.56302e-9-280.54253e-12*T)))))/(1+T*16.87985e-3);%SysCad steam properties: http://help.syscad.net/index.php/Water_and_Steam_Properties
set(hObject,'String',sprintf('%0.4f',handles.RhoH));

guidata(hObject, handles);
%-------------------------------------------------------------------------


% --- Executes on button press in CB_Const.
function CB_Const_Callback(hObject, eventdata, handles)
%-------------------------------------------------------------------------
%-- ENABLE/DISABLE TEMPERATURE DROP DOWN MENU
%-------------------------------------------------------------------------
if get(hObject,'Value')==1 %Disable temp menu
    set(handles.DD_Temp,'enable','off');
else %Re-enable temp menu
    set(handles.DD_Temp,'enable','on');
end
%-------------------------------------------------------------------------



function ET_Temp_Callback(hObject, eventdata, handles)
%-------------------------------------------------------------------------
%-- UPDATE STATIC TEMPERATURE
%-------------------------------------------------------------------------
T=str2double(get(hObject,'String'));
[RhoH,RhoL,dRho,Flag]=DensDiff(T,handles.light,handles.dense,[],[]);
set(handles.ET_StaticLight,'String',sprintf('%0.4f',RhoL));
set(handles.ET_StaticDense,'String',sprintf('%0.4f',RhoH));
set(handles.ST_dRho,'String',sprintf('%0.4f',dRho));
disp('Density updated')
%-------------------------------------------------------------------------


% --- Executes during object creation, after setting all properties.
function ET_Temp_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_Temp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in PB_LoadIm.
function PB_LoadIm_Callback(hObject, eventdata, handles)
%--------------------------------------------------------------------
%--- ASK FOR AND LOAD IMAGES, ASK FOR FITTING AREAS
%.. Image type
    ImType = handles.ImType;

%.. Load the image
    switch ImType
        case 'Movie'
            [ImName,Path] = uigetfile('*.*');
            %use full file names for deployed application.
            ImName = fullfile(Path, ImName);
            handles.ImName=ImName;
            DropMovie = VideoReader(ImName);
            nFrames = DropMovie.NumberOfFrames;
            set(handles.ST_TotalFrames,'String',sprintf('%d',nFrames));
            set(handles.ET_End,'String',sprintf('%d',nFrames));
            ST = str2double(get(handles.ET_ST,'String'));
            Grey = read(DropMovie, ST);
        case 'Still'
            [ImName,Path] = uigetfile('*.*');
            handles.ImName = fullfile(Path, ImName);
            Grey = imread(handles.ImName);

        case 'Sequence'
            disp('Specify image path - select an image from the set');
            [~,Path]=uigetfile(sprintf('.%s',get(handles.ET_Ext,'String')),'multiselect','off');
            ImName=sprintf('%s%04i.%s',get(handles.ET_Prefix,'String'),str2double(get(handles.ET_ST,'String')),get(handles.ET_Ext,'String'));
            Grey = imread(fullfile(Path, ImName));
            disp(ImName);
        otherwise
            Msg=sprintf('Opps, it looks like the image type was not defined. Returning you to the main GUI.');
            h=msgbox(Msg); uiwait(h);
            Path=[];
```

```
                return
        end
        setappdata(getappdata(0,'GUI_Hmain'),'Path',Path);
        handles.Path=Path;
%.. Convert to greyscale
        sz=size(size(Grey));
        if sz(2)>2
            Grey=rgb2gray(Grey);
        end
%.. Flip raised holm images
        if get(handles.CB_Flip,'value')==1
           Grey=flipud(Grey);
        end
%.. Display filename
        set(handles.ST_ImName,'String',ImName);

%.. Display image
        axes(handles.axes1); hold off,
        imshow(Grey);

%-----------------------------------------------------------------
%--- ASK FOR FITTING AREAS
%-----------------------------------------------------------------
        button=questdlg('Crop images?','Crop?','Yes','No','No');
        if strcmp(button,'Yes')==1
            fprintf(1,'Select area to crop\n')
            rec=round(getrect());
            Crop=[rec(1),rec(2),rec(1)+rec(3),rec(2)+rec(4)];
            if Crop(3)>size(Grey,2);Crop(3)=size(Grey,2);end
            if Crop(4)>size(Grey,1);Crop(4)=size(Grey,1);end
        else
            Crop=[1,1,size(Grey,2),size(Grey,1)];
        end
        Cnr=Crop;
        Grey=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));
        handles.Crop=Cnr;
        imshow(Grey);
        hold on

        fprintf(1,'Selecting an area outside of image boundaries will return an error\
            n')
        fprintf(1,'Please select the area to fit for the holm (right)\n(click and drag
             box):\n')
        rec=round(getrect());
        HlmCnrR=[rec(1),rec(2),rec(1)+rec(3),rec(2)+rec(4)];
        if HlmCnrR(3)>size(Grey,2);HlmCnrR(3)=size(Grey,2);end
```

```
    if HlmCnrR(4)>size(Grey,1);HlmCnrR(4)=size(Grey,1);end

    fprintf(1,'Please select the area to fit for the holm (left)\n(click and drag box):\n')
    rec=round(getrect());
    HlmCnrL=[rec(1),rec(2),rec(1)+rec(3),rec(2)+rec(4)];
    if HlmCnrL(3)>size(Grey,2);HlmCnrL(3)=size(Grey,2);end
    if HlmCnrL(4)>size(Grey,1);HlmCnrL(4)=size(Grey,1);end

    HlmCnr=[HlmCnrR;HlmCnrL];

    fprintf(1,'Please select the area to fit for the sphere (right) \n(click and drag box):\n')
    rec=round(getrect());
    SphCnr(1,:)=[rec(1),rec(2),rec(1)+rec(3),rec(2)+rec(4)];

    fprintf(1,'Please select the area to fit for the sphere (left) \n(click and drag box):\n')
    rec=round(getrect());
    SphCnr(2,:)=[rec(1),rec(2),rec(1)+rec(3),rec(2)+rec(4)];

    %Display boxes
    plot([HlmCnrR(1),HlmCnrR(3),HlmCnrR(3),HlmCnrR(1),HlmCnrR(1)],[HlmCnrR(2),HlmCnrR(2),HlmCnrR(4),HlmCnrR(4),HlmCnrR(2)],'r');
    plot([HlmCnrL(1),HlmCnrL(3),HlmCnrL(3),HlmCnrL(1),HlmCnrL(1)],[HlmCnrL(2),HlmCnrL(2),HlmCnrL(4),HlmCnrL(4),HlmCnrL(2)],'r');
    plot([SphCnr(1,1),SphCnr(1,3),SphCnr(1,3),SphCnr(1,1),SphCnr(1,1)],[SphCnr(1,2),SphCnr(1,2),SphCnr(1,4),SphCnr(1,4),SphCnr(1,2)],'r');
    plot([SphCnr(2,1),SphCnr(2,3),SphCnr(2,3),SphCnr(2,1),SphCnr(2,1)],[SphCnr(2,2),SphCnr(2,2),SphCnr(2,4),SphCnr(2,4),SphCnr(2,2)],'r');

    %Save
    handles.Grey=Grey;
    handles.HlmCnr=HlmCnr;
    handles.SphCnr=SphCnr;
    %handles.Path=Path;
    guidata(hObject, handles);
%------------------------------------------------------------------------


% --- Executes on button press in PB_LoadMat.
function PB_LoadMat_Callback(hObject, eventdata, handles)
%------------------------------------------------------------------------
%--- LOAD SAVED MATRIX
% %.. Read Tag
%     Tag = char(get(handles.ET_Tag,'String'));
```

```
%     file=fullfile(handles.Path,sprintf('%s-Data.mat',Tag));
%
% %.. If the file does not exist, display error.
%     if exist(file,'file')==0
%         Msg=sprintf('File does not exist. (%s)\r\n',file);
%         fprintf(1,Msg);
%         h=msgbox(Msg); uiwait(h);
%     else
%.. Else, load the file and update the GUI & handles.
        fprintf(1,'Select an existing datafile.');
        [ResultFile,Path]=uigetfile('.mat','multiselect','off');
        FullFileName=fullfile(Path,ResultFile);
        disp(FullFileName);
        load(FullFileName);
        set(handles.ET_Tag,'String',Data.Tag);
        handles.Path=Path;

        setappdata(getappdata(0,'GUI_Hmain'),'Path',Path);
        setappdata(getappdata(0,'GUI_Hmain'),'Tag',Data.Tag);

        set(handles.ST_adj,'String',sprintf('Adj: %0.4f (deg)',Data.Theta));
        handles.Theta=Data.Theta;
        handles.Alpha=Data.Alpha

        %.. Update handles
        handles.ImType=Data.ImType;
        set(get(handles.ImTypePanel,'SelectedObject'),'Value',1);
        handles.SphCnr=Data.SphCnr;
        handles.HlmCnr=Data.HlmCnr;
        handles.Crop=Data.Crop;
                Cnr=handles.Crop;
        %.. Physical properies
        handles.RhoL=Data.RhoL;
        set(handles.ET_StaticLight,'String',sprintf('%0.4f',handles.RhoL));
        handles.RhoH=Data.RhoH;
        set(handles.ET_StaticDense,'String',sprintf('%0.4f',handles.RhoH));
        handles.dRho=Data.dRho;
        set(handles.ST_dRho,'String',sprintf('%0.4f',handles.dRho));
        handles.Temp=Data.Temp;
        set(handles.ET_Temp,'String',sprintf('%0.2f',handles.Temp));
        %.. Analysis info
        handles.ST=Data.ST;
        set(handles.ET_ST,'String',sprintf('%d',handles.ST));
        handles.End=Data.End;
        set(handles.ET_End,'String',sprintf('%d',handles.End));
        handles.Int=Data.Int;
```

```
        set(handles.ET_Intv,'String',sprintf('%d',handles.Int));
        handles.Ref=Data.Ref;
        set(handles.ET_Ref,'String',sprintf('%d',handles.Ref));
        %.. Optimisation info
        handles.AngleLow=Data.AngleLow;
        set(handles.ET_AngleLow,'String',sprintf('%0.1f',handles.AngleLow));
        handles.AngleHigh=Data.AngleHigh;
        set(handles.ET_AngleHigh,'String',sprintf('%0.1f',handles.AngleHigh));
        handles.ThetaAdjLow=Data.ThetaAdjLow;
        set(handles.ET_ThetaAdjLow,'String',sprintf('%0.2f',handles.ThetaAdjLow));
        handles.ThetaAdjHigh=Data.ThetaAdjHigh;
        set(handles.ET_ThetaAdjHigh,'String',sprintf('%0.2f',handles.ThetaAdjHigh));

        if exist('Notes')==1
            set(handles.ET_Notes,'String',Notes);
        end

        %.. Update image
%.. Load the image
        switch handles.ImType
            case 'Sequence'
                set(handles.ET_Prefix,'String',Data.Prefix);
                set(handles.ET_Ext,'String',Data.Ext);
                ImName=sprintf('%s%04i.%s',Data.Prefix,Data.ST,Data.Ext);
                Grey = imread(fullfile(handles.Path,ImName));
            case 'User'
                set(handles.ET_Prefix,'String',Data.Prefix);
                set(handles.ET_Ext,'String',Data.Ext);
                ImName=sprintf('%s%04i.%s',Data.Prefix,Data.ST,Data.Ext);
                ImName=fullfile(handles.Path,ImName);
                Grey = imread(ImName);
            case 'Movie'
                handles.ImName=Data.ImName;
                ImName=handles.ImName;
                set(handles.ST_ImName,'String',handles.ImName);
                DropMovie = VideoReader(handles.ImName);
                nFrames = DropMovie.NumberOfFrames;
                set(handles.ST_TotalFrames,'String',sprintf('%d',nFrames));
                Grey = read(DropMovie, Data.ST);
            case 'Still'
                handles.ImName=Data.ImName;
                ImName=handles.ImName;
                set(handles.ST_ImName,'String',handles.ImName);
                Grey = imread(ImName);
            otherwise
```

```
                printf('Interesting...issue loading ImType.')
            end
%.. Flip raised holm images
    if Data.Flip==1;
        Grey=flipud(Grey);
        set(handles.CB_Flip,'value',1);
    end

%.. Crop image
    Grey=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));
%.. Display filename
    set(handles.ST_ImName,'String',ImName);

%.. Display image
    axes(handles.axes1); hold off
    imshow(Grey); hold on
    HlmCnr=Data.HlmCnr; SphCnr=Data.SphCnr;
    %Display boxes
    plot([HlmCnr(1,1),HlmCnr(1,3),HlmCnr(1,3),HlmCnr(1,1),HlmCnr(1,1)],[HlmCnr
        (1,2),HlmCnr(1,2),HlmCnr(1,4),HlmCnr(1,4),HlmCnr(1,2)],'r');
    plot([HlmCnr(2,1),HlmCnr(2,3),HlmCnr(2,3),HlmCnr(2,1),HlmCnr(2,1)],[HlmCnr
        (2,2),HlmCnr(2,2),HlmCnr(2,4),HlmCnr(2,4),HlmCnr(2,2)],'r');
    plot([SphCnr(1,1),SphCnr(1,3),SphCnr(1,3),SphCnr(1,1),SphCnr(1,1)],[SphCnr
        (1,2),SphCnr(1,2),SphCnr(1,4),SphCnr(1,4),SphCnr(1,2)],'r');
    plot([SphCnr(2,1),SphCnr(2,3),SphCnr(2,3),SphCnr(2,1),SphCnr(2,1)],[SphCnr
        (2,2),SphCnr(2,2),SphCnr(2,4),SphCnr(2,4),SphCnr(2,2)],'r');

    %Save
    handles.Grey=Grey;
    %end

    guidata(hObject, handles);

    fprintf(1,'...complete \n');
%--------------------------------------------------------------------------

% --- Executes on button press in PB_Temp.
function PB_Temp_Callback(hObject, eventdata, handles)
%--------------------------------------------------------------------------
%--- CALL TIMELINE GUI
%.. Export fps
    GUI_Hmain = getappdata(0,'GUI_Hmain');
    setappdata(GUI_Hmain,'Fps',str2double(get(hObject,'String')));

%.. Call timeline GUI
    h=Timeline10;
```

```
    uiwait(h)
%------------------------------------------------------------------

% --- Executes on button press in PB_Edge.
function PB_Edge_Callback(hObject, eventdata, handles)
disp('Opening edge GUI')
%------------------------------------------------------------------
%--- VIEW EDGE DETECTION
%------------------------------------------------------------------
% Show the detected edge  prior to analysis - show if there are issues with
% the image quality.
% Allow user to modify the threshold for the mask.
%------------------------------------------------------------------
    SphCnr=handles.SphCnr;
    HlmCnr=handles.HlmCnr;
    ST=str2double(get(handles.ET_ST,'String'));
    End=str2double(get(handles.ET_End,'String'));
    Intv=str2double(get(handles.ET_Intv,'String'));
    Prefix=get(handles.ET_Prefix,'String');
    Extn=get(handles.ET_Ext,'String');
    Crop=handles.Crop;
    Flip=get(handles.CB_Flip,'Value');

%.. Update grey
    %ImType=handles.ImType;ImName=handles.ImName;
    if strcmp(handles.ImType,'Sequence')==1
        ImName=sprintf('%s%04i.%s',Prefix,ST,Extn);
        ImName=fullfile(handles.Path,ImName);
        Grey=imread(ImName);
        if Flip==1;
            Grey=flipud(Grey);
        end
        Cnr=Crop;
        Grey=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));
    elseif strcmp(handles.ImType,'Movie')==1
        DropMovie = VideoReader(handles.ImName);
        Grey = read(DropMovie, ST);
        %Grey = readframe(DropMovie, ST);
        if Flip==1;
            Grey=flipud(Grey);
        end
        Cnr=Crop;
        Grey=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));
    else
        Grey=handles.Grey;
    end
```

```

%.. Show image
    axes(handles.axes1); imshow(Grey); hold on

%.. Save a matrix to store threshold info
    Tag = char(get(handles.ET_Tag,'String'));
    FileName=fullfile(handles.Path,sprintf('%sThI',Tag));
    save(FileName,'Grey','SphCnr','HlmCnr','ST','End','Intv','Prefix','Extn','Crop
        ','Flip');
     %.. Set tag
    GUI_Hmain=getappdata(0,'GUI_Hmain');
    setappdata(GUI_Hmain,'Tag',Tag);
    setappdata(GUI_Hmain,'Path',handles.Path);

%.. Call Thresh GUI
    h=GUIthreshHv11b;
    disp('Loading....')
    uiwait(h)
    load(fullfile(handles.Path,sprintf('%sThO',Tag)));

%.. Upload data
    if exist('BWadj','var')==0
        fprintf(1,'Threshold data not saved. Using defaults.\n')
        handles.BWadj = 1;
        handles.CANadjL = [];
        handles.CANadjH = [];
        handles.wd = 5;
        handles.SBD=0;
            MorphProps.Size=15;
            MorphProps.Type='disk';
            MorphProps.FilLoc=[];
        handles.MorphProps=MorphProps;
        handles.NL=2;
        handles.BMask=1;
    else
        handles.BWadj = BWadj;
        handles.CANadjL = CANadjL;
        handles.CANadjH = CANadjH;
        handles.wd = wd;
        handles.SBD=SBD;
        handles.MorphProps=MorphProps;
        handles.SphCnr=SphCnr;
        handles.HlmCnr=HlmCnr;
        handles.NL=NL;
        try handles.BMask=BMask;
        catch
```

```
            handles.BMask=1; BMask =1;
        end
    end

    FileName=fullfile(handles.Path,'ThreshTemp1');
    save(FileName,'BWadj','CANadjL','CANadjH','wd','SBD','MorphProps','NL','BMask'
        );

    %GUI_Hmain=getappdata(0,'GUI_Hmain');
    %ImType = getappdata(GUI_Hmain,'ImType');
    disp('Thresh info updated.\n')
    guidata(hObject, handles);
%--------------------------------------------------------------------------

% --- Executes on button press in PB_Analyse.
function PB_Analyse_Callback(hObject, eventdata, handles)
%--------------------------------------------------------------------------
%-- ANALYSE -- PROGRAM FRONT END
%--------------------------------------------------------------------------
%.. Parameters
ST=str2double(get(handles.ET_ST,'String'));%starting frame
FTA=str2double(get(handles.ET_End,'String'));%ending frame
Intv=str2double(get(handles.ET_Intv,'String'));%Interval
Ref=str2double(get(handles.ET_Ref,'String'));%Interval

SL=str2double(get(handles.ET_SL,'String'));%search length
SINT=uint8(str2double(get(handles.ET_SINT,'String')));%interval for checkpoint
FPS=str2double(get(handles.ET_Fps,'String'));%frames per second
tol=str2double(get(handles.ET_Tol,'String')); %Search tolerance
N=str2double(get(handles.ET_N,'String')); %Number of iterations per SL

PrintYN=get(handles.CB_Print,'value');
try
    HL=handles.HL;
catch
    HL='low'
end

if get(handles.CB_Cnt,'value')~=1
    cla(handles.axes2);%clear axes
end

%--------------------------------------------------------------------------------------
%---Call Image properties------------------------------------------------------------
%   - Imcase determines which movie/Density to use, loads temp matrix TC
%--------------------------------------------------------------------------------------
```

```
%.. Load properties file
Identi=char(get(handles.ET_Tag,'String'));
load(fullfile(handles.Path,sprintf('%s-Data.mat',Identi)));

%.. Load edge detection file
if exist(fullfile(handles.Path,'ThreshTemp2.mat'),'file')==0
    msgbox('Please confirm the edge detection. Routing to main GUI.')
    return
else
    load(fullfile(handles.Path,'ThreshTemp1'))
    load(fullfile(handles.Path,'ThreshTemp2'))
    %Loads wd, BWadj, CANadjH, CANadjL, NL, SBD and BMask
end

%.. Unpack
Dense=Data.dense;%Dense fluid identifier
Light=Data.light;%Light fluid identifier
disp('get paragraph from GUI later')
Thta=Data.Theta;
nran=Data.nRan;
NumError=uint16(0);%NumPoor=uint16(0);num=uint16(0);Gamma_Sum=0;
%=========================================================================
% Determine simulation status
    % - New simulation
    % - Continue simulation
%=========================================================================
folder=sprintf('%s-%s',Identi,datestr(now,'ddmmyy-HHSS'));
setappdata(0,'folder',folder);
mkdir(fullfile(handles.Path,folder));
section=uint8(1);

%Diary for error logs.
DiaryName=fullfile(handles.Path,folder, sprintf('%s-%s-Diary.txt',Identi,datestr(now,'ddmmyy-HHSS')));
%diary(DiaryName)
LogID=fopen(DiaryName,'a');

fig=handles.axes1;
FDisp=1;

switch Data.ImType
    case 'Movie'
    FileName=Data.ImName;%movie file
    %-------------------------------------------------------------------------
    %---LOAD MOVIES-------------------------------------------------------
    %-------------------------------------------------------------------------
```

```
%Read initial frame
DropMovie = VideoReader(FileName);
nFrames = DropMovie.NumberOfFrames;

fprintf('Total frames: %i \n', nFrames);

k=ST;Rot='Yes';ok=false;
while ok == false && k <= FTA %cycle through until a readable frame is found
    try
        GREY = read(DropMovie, k);
        k=k+1;
        ok=true;
    catch err0
        k=k+1;
        erString0 = getReport(err0);
        fprintf(LogID, '...................... L0: failed reading frame %i. \r
            \n %s \r\n ',k,erString0);
        fprintf(1, '...................... L0: failed reading frame %i. \r\n %
            s \r\n ',k,erString0);
    end
end
if k==nFrames
    h=warndlg('L0: Unable to find a readable frame. Closing program.');
    fprintf(LogID, '************************Unable to find a readable frame.
        Program terminates.');
    fprintf(1, '************************Unable to find a readable frame.
        Program terminates.');
    return
end

%CFN=fullfile(folder,sprintf('Check%i.mat',section));
HFN=fullfile(handles.Path,folder,'Head.mat');
setappdata(0,'Head',HFN);
save(HFN, '-regexp', '^(?!(ST|FTA|Intv|HOLMhandle|ADJhandle|BROKENhandle|
    EDGEhandle|DropMovie|eventdata|fig|hObject)$).');%doesn't save ST,FTA or
    Intv so new parameters can be given.
FileName=fullfile(handles.Path,sprintf('HFN-%s',Identi));
save(FileName,'HFN','folder');
disp(FileName);

%--------------------------------------------------------------------------
%---Find a frame which can be read (frame 1)-------------------------------
%--------------------------------------------------------------------------
flg='Retry';
while strcmp(flg,'Cnt')==0
    try
```

```
            Grey = read(DropMovie, ST);
            flg='Cnt';
        catch err00
            ST=ST+1; %increment ST
            if ST>nFrames
                fprintf(LogID, '\r\n*****************Could not read any frames \r\n');
                fprintf(1, '\r\n*****************Could not read any frames \r\n');
                return
            end
            msgString = getReport(err00);
            fprintf(LogID, 'Error reading frame %i. Report: %s, ',k,msgString);
            fprintf(1, 'Error reading frame %i. Report: %s, ',k,msgString);
        end

    end
    if size(Grey,3)>1%convert colour images to greyscale
        Grey=rgb2gray(Grey);
    end
    if Data.Flip==1;
      Grey=flipud(Grey);
      set(handles.CB_Flip,'value',1);
    end
    %--------------------------------------------------------------------
    %--- FITTING AREAS
    Crop=Data.Crop;
    HlmCnr=Data.HlmCnr;
    SphCnr=Data.SphCnr;
    %--------------------------------------------------------------------

    %--------------------------------------------------------------------
    %--- ROTATION ANGLE
    Alpha=-Thta*pi/180;

    %--------------------------------------------------------------------

    %--------------------------------------------------------------------
    %--- TEMPERATURE FILE
    %--------------------------------------------------------------------
    TempFile=Data.TempFile;
    e=msgbox(sprintf('Temp feed: %s, Movie %i frames per second.  Click <OK> to
        continue or close this dialogue box to return to the main GUI',TempFile,
        FPS),'modal');
    try
        uiwait(e,60);
        fprintf(sprintf('Temp feed: %s.\r\n',TempFile));
```

```
    catch
        disp('Returning to main GUI');
        return
    end
    load(fullfile(handles.Path,sprintf('%s.mat',TempFile)));

    %----------------------------------------------------------------------
    %---- RESULTS FILE
    %----------------------------------------------------------------------
        Result_file=sprintf('%s-Section%iframes%i-%i.txt',folder,section,ST,FTA);
        fullFileNameA = fullfile(handles.Path,folder, Result_file);
        [fileIDa,errmsg]=fopen(fullFileNameA,'a');%use a+ for reading & writing, append. Open a new file each instance

        %--abbrv file header
        fprintf(fileIDa,'Identifier:, %s, folder:, %s \r\n ,%s: , results for file ,%s, right then left, %s\r\n',Identi,folder,FileName,datestr(now),FileName);
        fprintf(fileIDa,'Program version information: %s \r\n',get(handles.ST_Version,'String'));
        fprintf(fileIDa,',Adjustment:,%i , degrees\r\n Dense fluid:, %s, Light fluid:, %s \r\n',Thta,Dense,Light);
        fprintf(fileIDa,',frame , dRho , RhoH , RhoL , T ,tens(mN/mm)(best, ave), Shape (best, ave) , error (H, S, ave F), tens(mN/mm)(best, ave), Shape (best, ave) , error (H, S, ave F) , rR, rL \r\n');

        Results=zeros(int32((FTA-ST+1)/Intv),22);%reset each time
        i=uint8(1);%assign as integer

        CFN=fullfile(handles.Path,folder,sprintf('Check%i.mat',section));%new file
        save(CFN, '-regexp', '^(?!(ST|FTA|Intv|HOLMhandle|ADJhandle|EDGEhandle|DropMovie)$).')%doesn't save ST,FTA or Intv so new parameters can be given.

        %===============Checkpoint log==========
        %LogName=sprintf('Log%s-%i',folder,section);
        %logID=fopen(sprintf('C:\Users\14291160\Dropbox\PhD-ChemEng\%s.txt',LogName),'a');
        %===============Cleanup=================
        finishup = onCleanup(@() myCleanupFun(HFN, CFN, LogID));

        %===============Back End================
        im=1;itr=1;
        for k = ST : Intv : FTA %subsequent images
            h=msgbox(sprintf('Analyzing frame %i (%i/%i/%i)',k,ST,Intv,FTA));
           fprintf('--- processing interval %i: %d of %d',k, i, FTA-ST+1);
```

```
%------------------------------------------------------------------------
%---Find a frame which can be read---------------------------------------
%------------------------------------------------------------------------
CntFlag=false;ki=k-1;%find a frame which can be read
while CntFlag==false && ki<=k+Intv,%if flag is false, have not found an
     image to read
    ki=ki+1;  %(on first entry, ki=k-1+1=k
    try
        Grey = read(DropMovie, ki);
        CntFlag=true;
    catch errR
        msgString = getReport(errR);
        fprintf(1,'---------Unable to read frame %i.  \r\n', ki);
        fprintf(LogID,'Frame %i failed. (line 194). Report: %s',ki,
            msgString);
    end
end
ImName=sprintf('%s-%04i',Identi,ki);

%------------------------------------------------------------------------
%---Temperature & fluid densities----------------------------------------
%------------------------------------------------------------------------
T=TC(int16(ki/FPS)+1);%C.  Temp matrix is per second, from t=0s. Movie
     is FPS frames/s.
[RhoH,RhoL,dRho,Flag]=DensDiff(T,Light,Dense,Data.RhoL,Data.RhoH);
if strcmp(Flag, 'Stop')==1
    fprintf(fileIDa, 'Code terminated by user...');
    fprintf(LogID, 'Code terminated by user...');
    fprintf(1, 'Code terminated by user...');
    return %do not continue with code - error in TEMP file.
end
Results(im,1:5)=[ki,dRho,RhoH,RhoL,T]; % Keep temp/density info even if
     all frames failed
%------------------------------------------------------------------------
%---Return if no frame could be read-------------------------------------
%------------------------------------------------------------------------
if ki>=k+Intv && CntFlag==false,%no frame has been found
   fprintf(LogID,'*****Could not read a frame from this interval');
   im=im+1;itr=itr+1;
   continue %Go to next iteration of outer FOR loop
end
%------------------------------------------------------------------------
%---Analysis (Holm_MAIN*)------------------------------------------------
%------------------------------------------------------------------------

try
```

```
    %------------------------------------------------------------------------
    % GREY IMAGE
    %------------------------------------------------------------------------
       axes(handles.axes1);
        sz=size(size(Grey));
       if sz(2)>2,
            Grey=rgb2gray(Grey);
       end
       if Data.Flip==1;
          Grey=flipud(Grey);
          set(handles.CB_Flip,'value',1);
       end

       Cnr=Crop;
       Grey=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));
    %------------------------------------------------------------------------
    % GENERATE WHOLE IMAGE MASK ON BW IMAGE
    %------------------------------------------------------------------------
       [WIM,ThreshC,Thr]=WholeImageMask(Grey,BWadj,CANadjL,CANadjH,
           MorphProps,0);

    %------------------------------------------------------------------------
    % SPHERE EDGE DETECTION AND FIT
    %------------------------------------------------------------------------
        %--Edge detection, Left & Right sides
        [SphereCoordL,SphereCoordR]=ImMask(SphCnr(2,:),SphCnr(1,:),Grey,
            ThreshC,Thr,0,wd,0,WIM,[],NL,BMask);%'0' for flag - needed in
            ThreshGUI only.
        SphereCoord=[transpose(SphereCoordR),transpose(SphereCoordL)];%One
             layer - fit together

        %--Fit circle
        [SphereOpt,fvalS]=TriSphere(SphereCoordR(1,1)-SphereCoordL(1,1),
            SphereCoord);%first argument is a VERY rough guess of the
            width
        %--Optimised sphere coordinates
        x0=SphereOpt(1);
        y0=SphereOpt(2);%adjusting x,y to main image
        plot(x0,y0,'+r')
        R=SphereOpt(3);
        if R==0; disp('Error fitting sphere profile (R==0)');end

    %-----------------------------------------------------------------------------
    % HOLM EDGE DETECTION
    %-----------------------------------------------------------------------------
```

```
        [HolmCrdL,HolmCrdR]=ImMask(HlmCnr(2,:),HlmCnr(1,:),Grey,ThreshC,
            Thr,0,wd,SBD,WIM,[],NL,BMask);
        HolmCoordL=fliplr(transpose(HolmCrdL)); HolmCoordR=fliplr(
            transpose(HolmCrdR));

    %---------------------------------------------------------------
    % DISPLAY - EDGES PRIOR TO ROTATION
    %---------------------------------------------------------------
    Show=false;
    if Show==true
        imshow(Grey);
        %..Sphere edges
        plot(SphereCoordL(:,1),SphereCoordL(:,2),'r','linewidth',2)
        plot(SphereCoordR(:,1),SphereCoordR(:,2),'r','linewidth',2)
        %..Sphere fit
        SphIde=zeros(3,int32(R/5));
        k=0;i=int16(0);
        for i=x0+R:-2:x0-R
            k=k+1;
            SphIde(1,k)=i;
            SphIde(2,k)=sqrt(R^2-(i-x0)^2)+y0;
            SphIde(3,k)=-sqrt(R^2-(i-x0)^2)+y0;
        end
        plot(SphIde(1,:),SphIde(2,:),'y',SphIde(1,:),SphIde(3,:),'y');
        %..Holm
        plot(HolmCoordL(1,:),HolmCoordL(2,:),'.r')
        plot(HolmCoordR(1,:),HolmCoordR(2,:),'.r')

        drawnow
        g=msgbox('Edges ok?');
        uiwait(g);
    end

    %---------------------------------------------------------------
    % ROTATE COORDINATES
    %---------------------------------------------------------------
        if Alpha==0
            x0a=x0;
            y0a=y0;
            GreyA=Grey;
        else
            [HolmCoordL,HolmCoordR,x0a,y0a]=RotateCoords(Alpha,x0,y0,R,
                HolmCoordR,HolmCoordL,Grey,HL);
            GreyA=imrotate(Grey,Thta);%for plotting only
        end
```

```
                    HolmCoordLr=HolmCoordL;
                    HolmCoordLr(1,:)=size(GreyA,2)-HolmCoordL(1,:);

                    xL=size(GreyA,2)-x0a;

                    figure(FDisp), imshow(GreyA), hold on
                    axes(handles.axes1); hold off, imshow(GreyA), hold on
                    plot(HolmCoordL(1,:),HolmCoordL(2,:),'.-r');
                    plot(HolmCoordR(1,:),HolmCoordR(2,:),'.-r');
                    plot(x0a,y0a,'r+');

                    %--Theoretical points
                    SphIde=zeros(3,int32(R/5));
                    m=0;
                    for n=x0a+R:-2:x0a-R
                        m=m+1;
                        SphIde(1,m)=n;
                        SphIde(2,m)=sqrt(R^2-(n-x0a)^2)+y0a;
                        SphIde(3,m)=-sqrt(R^2-(n-x0a)^2)+y0a;
                    end
                    %--plot ideal circle for comparison
                    plot(SphIde(1,:),SphIde(2,:),':r',SphIde(1,:),SphIde(3,:),':r')
figure(1),
                    plot(HolmCoordL(1,:),HolmCoordL(2,:),'.-r');
                    plot(HolmCoordR(1,:),HolmCoordR(2,:),'.-r');
                    plot(x0a,y0a,'r+');
                    plot(SphIde(1,:),SphIde(2,:),':r',SphIde(1,:),SphIde(3,:),':r')
drawnow;
                %----------------------------------------------------------------
                % Analyse RIGHT
                %----------------------------------------------------------------
                    [GammaR,Shape,fval,OptStore]=Holm_MAIN_3('Right',ki,ImName,PrintYN
                        ,x0a,y0a,R,HolmCoordR,GreyA,fullfile(handles.Path,folder),fig,
                        FDisp,Data);
                        %Output::Gamma=[Gamma,GammaAveF];shape=aBF=[aBest,aAveF];fval
                            =[fvalH,fvalS,fHave];
                    Results(im,6:12)=[GammaR,Shape,fval];
                    fprintf(1,'Frame %d, Right - complete\n',ki);
                %----------------------------------------------------------------
                % Analyse LEFT
                %----------------------------------------------------------------
                    %Left -->code will flip image (fliplr)
                    figure(2); imshow(fliplr(GreyA)); hold on
                    plot(xL,y0a,'+r');%circle centre, flipped
                    plot(HolmCoordLr(1,:),HolmCoordLr(2,:),'r');

```

```
                [GammaL,Shape,fval,OptStore]=Holm_MAIN_3('Left',ki,ImName,PrintYN,
                    xL,y0a,R,HolmCoordLr,fliplr(GreyA),fullfile(handles.Path,
                    folder),fig,FDisp,Data);
                Results(im,13:19)=[GammaL,Shape,fval];
                Results(im,20)=R;
                fprintf(1,'Frame %d, Left - complete\n',ki);
                %abbreviated results file
                fprintf(fileIDa,'% 3.6f,',Results(im,:));
                fprintf(fileIDa,'\r\n');

                axes(handles.axes2), hold on
                plot(ki/FPS/60,GammaL(1),'+b', ki/FPS/60,GammaR(1),'+m','
                    markersize',3)%min
            %----------------------------------------------------------------------
            % Analyse PAIR for SINT intervals
            %----------------------------------------------------------------------
%                   if mod(im,SINT)==0 %multiple of SINT
%                       axes(handles.axes1);
%                       a2=0.5*Results(im,8)+0.5*Results(im,15);
%                       [HolmOpt,Er2S]=Holm_2sides_FixAngle(dRho,Data.Size,a2,x0a,
    y0a,R,HolmCoordL,HolmCoordR,GreyA,nran);
%                       Scale=Data.Size/2/R;%image scale mm/p.
%                       Gam2=dRho*9.81*1000/(HolmOpt(1)/Scale*1000)^2;
%                       Results(im,21:22)=[Gam2,HolmOpt(1)];
%                           %---image
%                       figname=sprintf('%i-Edge',k);
%                       fullFileName = fullfile(folder, figname);
%                       saveas(FDisp, [fullFileName '.png'])
%                   end
                  close(h)
            %----------------------------------------------------------------------
            % Analyse ENDS (if no error)
            %----------------------------------------------------------------------
            catch err1 %error in analysis
                loopEr1 = getReport(err1);
                fprintf(LogID, '.frame %i, failed Level 1, Report: %s \r\n',ki,
                    loopEr1);
                fprintf(1, '.frame %i, failed Level 1, Report: %s \r\n',ki,loopEr1
                    );
                NumError=NumError+1;
                close(h)
            end
            im=im+1;
            itr=itr+1;
        end
        Fullfilename=fullfile(handles.Path,folder, 'Results.mat');
```

```
        save(Fullfilename,'Results');

        axes(handles.axes2), hold on
            plot(Results(:,1)/FPS/60,Results(:,21),'+r','markersize',3)%plot gamma (aVE)
            xlabel('time (min)'); ylabel('interfacial tension (L,R BEST), mN/mm, Temp(C)');

        %--Microwave data

        load(fullfile(handles.Path,'MData.mat')); %microwave data
        %MS=(MicrStart)*FPS;%frame when microwave is turned on, frames
        %MO=(MicrStart+irTime*60)*FPS;%frame when microwave is turned off, Frames
        %--Results
        figure(4); hold on;
        subplot(2,3,1);hold on%plotting surface tension against frame
            plot(Results(:,1),Results(:,6),'-om','markersize',3)%plot gamma (BEST)
            plot(Results(:,1),Results(:,13),'-ob','markersize',3)%plot gamma (BEST)
            plot(Results(:,1),Results(:,21),'+r','markersize',3)%plot gamma (aVE)
            xlabel('frame'); ylabel('interfacial tension (BEST), mN/mm');
            %plot([MS,MS],[0,100],'y',[MO,MO],[0,100],'y');

        subplot(2,3,2); hold on%error
            plot(Results(:,1),Results(:,10),'-om','markersize',3)%plot error (full)
            plot(Results(:,1),Results(:,12),'+m','markersize',3)%plot error (ave)
            plot(Results(:,1),Results(:,17),'-ob','markersize',3)%plot error (full)
            plot(Results(:,1),Results(:,19),'+b','markersize',3)%plot error (ave)
            plot(Results(:,1),Results(:,6)-Results(:,13),'-*r','markersize',3)%plot LR diff
            xlabel('frame'); ylabel('error');
            %plot([MS,MS],[0,5],'y',[MO,MO],[0,5],'y');

        subplot(2,3,4); hold on%plot shape factor against frame
            plot(Results(:,1),Results(:,8),'-om','markersize',3)%plot error (full)
            plot(Results(:,1),Results(:,9),'+m','markersize',3)%plot error (ave)
            plot(Results(:,1),Results(:,15),'-ob','markersize',3)%plot error (full)
            plot(Results(:,1),Results(:,16),'+b','markersize',3)%plot error (ave)
            xlabel('frame'); ylabel('shape factor');
            %plot([MS,MS],[0,0.005],'y',[MO,MO],[0,0.005],'y');

        subplot(2,3,3); hold on %plot ave tension
            plot(Results(:,1),Results(:,7),'-+m','markersize',3)%plot gamma
```

```
            plot(Results(:,1),Results(:,14),'-+b','markersize',3)%plot gamma
            xlabel('frame'); ylabel('Mean tension');
            %plot([MS,MS],[0,100],'y',[MO,MO],[0,100],'y');

        subplot(2,3,5); hold on%plot density
            plot(Results(:,1),Results(:,2),'om','markersize',3)%plot rhoDiff
            plot(Results(:,1),Results(:,3),'oc','markersize',3)%plot RhoH
            plot(Results(:,1),Results(:,4),'om','markersize',3)%plot RhoL
            xlabel('frame'); ylabel('density difference');
            %plot([MS,MS],[0,1000],'y',[MO,MO],[0,1000],'y');

        subplot(2,3,6); hold on
            plot(Results(:,1),Results(:,5),'oc','markersize',3)%plot temp
            plot(Results(:,1),Results(:,5),'-b')%plot temp
            xlabel('frame'); ylabel('Temperature');
            %plot([MS,MS],[0,100],'y',[MO,MO],[0,100],'y');

    case 'Still'
        FileName=Data.ImName;%image file
        %-------------------------------------------------------------
        %---- LOAD IMAGE
        Grey=imread(Data.ImName);
        if Data.Flip==1;
          Grey=flipud(Grey);
          set(handles.CB_Flip,'value',1);
        end

        sz=size(size(Grey));
        if sz(2)>2,
            Grey=rgb2gray(Grey);
        end
        Cnr=Data.Crop;
        Grey=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));

        %-------------------------------------------------------------
        %---- RESULTS FILE
        %-------------------------------------------------------------
        Result_file=sprintf('%s-Section%iframes%i-%i.txt',folder,section,ST,FTA);
        fullFileNameA = fullfile(handles.Path,folder, Result_file);
        [fileIDa,errmsg]=fopen(fullFileNameA,'a');%use a+ for reading & writing,
            append. Open a new file each instance

        %--abbrv file header
        fprintf(fileIDa,'Identifier:, %s, folder:, %s \r\n,%s: , results for file
             ,%s, right then left, %s\r\n',Identi,folder,FileName,datestr(now),
            FileName);
```

```
fprintf(fileIDa,'Program version information: %s \r\n',get(handles.ST_Version,'String'));
fprintf(fileIDa,',Adjustment:,%i , degrees\r\n Dense fluid:, %s, Light fluid:, %s \r\n',Thta,Dense,Light);
fprintf(fileIDa,',frame , dRho , RhoH , RhoL , T ,tens(mN/mm)(best, ave), Shape (best, ave) , error (H, S, ave F), tens(mN/mm)(best, ave), Shape (best, ave) , error (H, S, ave F) , rR, rL \r\n');

Results=zeros(1,22);%reset each time
ki=1; im=1;
%-----------------------------------------------------------------

%-----------------------------------------------------------------
%---- FITTING AREAS
HlmCnr=Data.HlmCnr;
SphCnr=Data.SphCnr;
%-----------------------------------------------------------------

%-----------------------------------------------------------------
%---- ROTATION ANGLE
Alpha=-Thta*pi/180;
%-----------------------------------------------------------------
%-----------------------------------------------------------------
% GENERATE WHOLE IMAGE MASK ON BW IMAGE
%-----------------------------------------------------------------
[WIM,ThreshC,Thr]=WholeImageMask(Grey,BWadj,CANadjL,CANadjH,MorphProps,0);

%-----------------------------------------------------------------
% SPHERE EDGE DETECTION AND FIT
%-----------------------------------------------------------------
%--Edge detection, Left & Right sides
[SphereCoordL,SphereCoordR]=ImMask(SphCnr(2,:),SphCnr(1,:),Grey,ThreshC,Thr,0,wd,SBD,WIM,[],NL,BMask);%'0' for flag - needed in ThreshGUI only.
SphereCoord=[transpose(SphereCoordR),transpose(SphereCoordL)];%One layer - fit together

%--Fit circle
[SphereOpt,fvalS]=TriSphere(SphereCoordR(1,1)-SphereCoordL(1,1),SphereCoord);%first argument is a VERY rough guess of the width
%--Optimised sphere coordinates
x0=SphereOpt(1);
y0=SphereOpt(2);%adjusting x,y to main image
plot(x0,y0,'+r')
R=SphereOpt(3);
if R==0; disp('Error fitting sphere profile (R==0)');end
```

```

%--Theoretical points
SphIde=zeros(3,int32(R/5));
k=0;i=int16(0);
for i=x0+R:-2:x0-R
    k=k+1;
    SphIde(1,k)=i;
    SphIde(2,k)=sqrt(R^2-(i-x0)^2)+y0;
    SphIde(3,k)=-sqrt(R^2-(i-x0)^2)+y0;
end

%------------------------------------------------------------------------
% HOLM EDGE DETECTION
%------------------------------------------------------------------------
[HolmCrdL,HolmCrdR]=ImMask(HlmCnr(2,:),HlmCnr(1,:),Grey,ThreshC,Thr,0,wd,
    SBD,WIM,[],NL,BMask);
HolmCoordL=fliplr(transpose(HolmCrdL)); HolmCoordR=fliplr(transpose(
    HolmCrdR));

%------------------------------------------------------------------------
% DISPLAY - EDGES PRIOR TO ROTATION
%------------------------------------------------------------------------
%    imshow(Grey);
%    %..Sphere edges
%    plot(SphereCoordL(:,1),SphereCoordL(:,2),'r','linewidth',2)
%    plot(SphereCoordR(:,1),SphereCoordR(:,2),'r','linewidth',2)
%    %..Sphere fit
%    plot(SphIde(1,:),SphIde(2,:),'r',SphIde(1,:),SphIde(3,:),'r');
%    %..Holm
%    plot(HolmCoordL(1,:),HolmCoordL(2,:),'r','linewidth',2)
%    plot(HolmCoordR(1,:),HolmCoordR(2,:),'r','linewidth',2)

%------------------------------------------------------------------------
% ROTATE COORDINATES
%------------------------------------------------------------------------
[HolmCoordL,HolmCoordR,x0a,y0a]=RotateCoords(Alpha,x0,y0,R,HolmCoordR,
    HolmCoordL,Grey,HL);
GreyA=imrotate(Grey,Thta);%for plotting only

HolmCoordLr=HolmCoordL;
HolmCoordLr(1,:)=size(GreyA,2)-HolmCoordL(1,:);

xL=size(GreyA,2)-x0a;

figure(FDisp), imshow(GreyA), hold on
axes(handles.axes1); hold off, imshow(GreyA), hold on
```

```
            plot(HolmCoordL(1,:),HolmCoordL(2,:),'r');
            plot(HolmCoordR(1,:),HolmCoordR(2,:),'r');
            plot(x0a,y0a,'ro');

            %--Theoretical points
            SphIde=zeros(3,int32(R/5));
            m=0;
            for n=x0a+R:-2:x0a-R
                m=m+1;
                SphIde(1,m)=n;
                SphIde(2,m)=sqrt(R^2-(n-x0a)^2)+y0a;
                SphIde(3,m)=-sqrt(R^2-(n-x0a)^2)+y0a;
            end
            %--plot ideal circle for comparison
            plot(SphIde(1,:),SphIde(2,:),':r',SphIde(1,:),SphIde(3,:),':r')

            %-----------------------------------------------------------------------
            % Analyse RIGHT
            %-----------------------------------------------------------------------
            [GammaR,Shape,fval,OptStoreR]=Holm_MAIN_3('Right',1,Data.ImName,PrintYN,
                x0a,y0a,R,HolmCoordR,GreyA,fullfile(handles.Path,folder),fig,FDisp,
                Data);

            %Output::Gamma=[Gamma,GammaAveF];shape=aBF=[aBest,aAveF];fval=[fvalH,fvalS
                ,fHave];
            %[RhoH,RhoL,dRho,Flag]=DensDiff(T,Light,Dense,Data.RhoL,Data.RhoH);
            Results(im,1:5)=[ki,handles.dRho,handles.RhoH,handles.RhoL,str2double(get(
                handles.ET_Temp,'String'))];
            Results(im,6:12)=[GammaR,Shape,fval];
            fprintf(1,'Right - complete\n');
            %-----------------------------------------------------------------------
            % Analyse LEFT
            %-----------------------------------------------------------------------
            %Left -->code will flip image (fliplr)
            [GammaL,Shape,fval,OptStoreL]=Holm_MAIN_3('Left',1,Data.ImName,PrintYN,xL,
                y0a,R,HolmCoordLr,fliplr(GreyA),fullfile(handles.Path,folder),fig,
                FDisp,Data);
            Results(im,13:19)=[GammaL,Shape,fval];
            Results(im,20)=R;
            fprintf(1,'Left - complete\n');
            %abbreviated results file
            fprintf(fileIDa,'% 3.7f,',Results(im,:));
            fprintf(fileIDa,'\r\n');

            %-----------------------------------------------------------------------
            % Display
```

```
        %---------------------------------------------------------------------------
        %Data.ImName includes PATH: This is Path/IMG_XXXX.xxx
        figname=sprintf('%s-Edge',Data.ImName(end-11:end-4));%this removes the file extension
        fullFileName = fullfile(handles.Path,folder, figname);
        saveas(FDisp, [fullFileName '.png'])

    %---------------------------------------------------------------------------
    % Analyse ENDS (Still)
    %---------------------------------------------------------------------------
    case 'Sequence'
        %-----------------------------------------------------------------
        %--- FITTING AREAS
        Crop=Data.Crop;
        HlmCnr=Data.HlmCnr;
        SphCnr=Data.SphCnr;
        %-----------------------------------------------------------------

        %-----------------------------------------------------------------
        %--- ROTATION ANGLE
        Alpha=-Thta*pi/180;

        %-----------------------------------------------------------------

        %-----------------------------------------------------------------
        %--- TEMPERATURE FILE
        %-----------------------------------------------------------------
        if get(handles.CB_Const,'value')==1
            T=str2double(get(handles.ET_Temp,'string'));
            Tc=true;
        else
            TempFile=Data.TempFile;
            load(fullfile(handles.Path,sprintf('%s.mat',TempFile)));
            Tc=false;
        end
        %-----------------------------------------------------------------
        %--- RESULTS FILE
        %-----------------------------------------------------------------
        Result_file=sprintf('%s-Section%iframes%i-%i.txt',folder,section,ST,FTA);
        fullFileNameA = fullfile(handles.Path,folder, Result_file);
        [fileIDa,errmsg]=fopen(fullFileNameA,'a');%use a+ for reading & writing, append. Open a new file each instance

        %--abbrv file header
        fprintf(fileIDa,'Identifier:, %s, folder:, %s \r\n,%s: , results for file right then left\r\n',Identi,folder,datestr(now));
```

```
        fprintf(fileIDa,'Program version information: %s \r\n',get(handles.
            ST_Version,'String'));
        fprintf(fileIDa,',Adjustment:,%i , degrees\r\n Dense fluid:, %s, Light
            fluid:, %s \r\n',Thta,Dense,Light);
        fprintf(fileIDa,',frame , dRho , RhoH , RhoL , T ,tens(mN/mm)(best, ave),
            Shape (best, ave) , error (H, S, ave F), tens(mN/mm)(best, ave),
            Shape (best, ave) , error (H, S, ave F) , rR, rL \r\n');

        Results=zeros(int32((FTA-ST+1)/Intv),22);%reset each time
        i=uint8(1);%assign as integer

        CFN=fullfile(folder,sprintf('Check%i.mat',section));%new file
        HFN=fullfile(handles.Path,folder,'Head.mat');
        setappdata(0,'Head',HFN);
        save(HFN, '-regexp', '^(?!(ST|FTA|Intv|HOLMhandle|ADJhandle|BROKENhandle|
            EDGEhandle|eventdata|fig|hObject)$).');%doesn't save ST,FTA or Intv so
             new parameters can be given.
        FileName=fullfile(handles.Path,sprintf('HFN-%s',Identi));
        save(FileName,'HFN','folder');
        disp(FileName);
        %===============Cleanup=================
        finishup = onCleanup(@() myCleanupFun(HFN, CFN, LogID));

        %===============Back End================
        im=1;itr=1;t0=Ref;NumError=0;%May change manually
        for k = ST : Intv : FTA %subsequent images
            h=msgbox(sprintf('Analyzing frame %i (%i/%i/%i)',k,ST,Intv,FTA));
            fprintf('--- processing interval %i: %d of %d',k, i, FTA-ST+1);
            %---------------------------------------------------------------------
            %---Find a frame which can be read------------------------------------
            %---------------------------------------------------------------------
            CntFlag=false;ki=k-1;%find a frame which can be read
            while CntFlag==false && ki<=k+Intv,%if flag is false, have not found an
                 image to read
                ki=ki+1;  %(on first entry, ki=k-1+1=k
                try
                    ImFile=fullfile(handles.Path,sprintf('%s%04i.%s',get(handles.
                        ET_Prefix,'String'),ki,get(handles.ET_Ext,'String')));
                    disp(ImFile);
                    Grey = imread(ImFile);
                    CntFlag=true;
                catch errR
                    msgString = getReport(errR);
                    fprintf(1,'----------Unable to read frame %i. \r\n', ki);
                    fprintf(LogID,'Frame %i failed. (line 194). Report: %s',ki,
                        msgString);
```

```
            end
        end
        %--------------------------------------------------------------------------
        %---Temperature & fluid densities------------------------------------------
        %--------------------------------------------------------------------------
        if Tc==false
%           t=(ki-t0+1)*5;
            T=TC(int16((ki-t0)/FPS)+1);%C.  Temp matrix is per second, from t=0 s. Movie is FPS frames/s.
        %else use constant T
        end
        [RhoH,RhoL,dRho,Flag]=DensDiff(T,Light,Dense,Data.RhoL,Data.RhoH);
        if strcmp(Flag, 'Stop')==1
            fprintf(fileIDa, 'Code terminated by user...');
            fprintf(LogID, 'Code terminated by user...');
            fprintf(1, 'Code terminated by user...');
            return %do not continue with code - error in TEMP file.
        end
        Results(im,1:5)=[ki,dRho,RhoH,RhoL,T]; % Keep temp/density info even if all frames failed
        %--------------------------------------------------------------------------
        %---Return if no frame could be read----------------------------------------
        %--------------------------------------------------------------------------
        if ki>=k+Intv && CntFlag==false,%no frame has been found
           fprintf(LogID,'*****Could not read a frame from this interval');
           im=im+1;itr=itr+1;
           continue %Go to next iteration of outer FOR loop
        end
        %--------------------------------------------------------------------------
        %---Analysis (Holm_MAIN*)---------------------------------------------------
        %--------------------------------------------------------------------------

        try
         %---------------------------------------------------------------
         % GREY IMAGE
         %---------------------------------------------------------------
            axes(handles.axes1);
             sz=size(size(Grey));
            if sz(2)>2,
                Grey=rgb2gray(Grey);
            end
            if Data.Flip==1;
               Grey=flipud(Grey);
               set(handles.CB_Flip,'value',1);
            end
            Cnr=Crop;
```

```
        Grey=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));

    %------------------------------------------------------------
    % GENERATE WHOLE IMAGE MASK ON BW IMAGE
    %------------------------------------------------------------
        [WIM,ThreshC,Thr]=WholeImageMask(Grey,BWadj,CANadjL,CANadjH,
            MorphProps,0);

    %------------------------------------------------------------
    % SPHERE EDGE DETECTION AND FIT
    %------------------------------------------------------------
        %--Edge detection, Left & Right sides
        [SphereCoordL,SphereCoordR]=ImMask(SphCnr(2,:),SphCnr(1,:),Grey,
            ThreshC,Thr,0,wd,0,WIM,[],NL,BMask);%'0' for flag - needed in
            ThreshGUI only.
        SphereCoord=[transpose(SphereCoordR),transpose(SphereCoordL)];%One
            layer - fit together

        %--Fit circle
        [SphereOpt,fvalS]=TriSphere(SphereCoordR(1,1)-SphereCoordL(1,1),
            SphereCoord);%first argument is a VERY rough guess of the
            width
        %--Optimised sphere coordinates
        x0=SphereOpt(1);
        y0=SphereOpt(2);%adjusting x,y to main image
        plot(x0,y0,'+r')
        R=SphereOpt(3);
        if R==0; disp('Error fitting sphere profile (R==0)');end

    %------------------------------------------------------------------------
    % HOLM EDGE DETECTION
    %------------------------------------------------------------------------
        [HolmCrdL,HolmCrdR]=ImMask(HlmCnr(2,:),HlmCnr(1,:),Grey,ThreshC,
            Thr,0,wd,SBD, WIM,[],NL,BMask);
        HolmCoordL=fliplr(transpose(HolmCrdL)); HolmCoordR=fliplr(
            transpose(HolmCrdR));

    %------------------------------------------------------------------------
    % ROTATE COORDINATES
    %------------------------------------------------------------------------
        if Alpha==0
            x0a=x0;
            y0a=y0;
            GreyA=Grey;
        else
```

```
                [HolmCoordL,HolmCoordR,x0a,y0a]=RotateCoords(Alpha,x0,y0,R,HolmCoordR,HolmCoordL,Grey,HL);
                GreyA=imrotate(Grey,Thta);%for plotting only
            end

            HolmCoordLr=HolmCoordL;
            HolmCoordLr(1,:)=size(GreyA,2)-HolmCoordL(1,:);

            xL=size(GreyA,2)-x0a;

            figure(FDisp), imshow(GreyA), hold on
            axes(handles.axes1); hold off, imshow(GreyA), hold on
            plot(HolmCoordL(1,:),HolmCoordL(2,:),'.-r');
            plot(HolmCoordR(1,:),HolmCoordR(2,:),'.-r');
            plot(x0a,y0a,'r+');

            %--Theoretical points
            SphIde=zeros(3,int32(R/5));
            m=0;
            for n=x0a+R:-2:x0a-R
                m=m+1;
                SphIde(1,m)=n;
                SphIde(2,m)=sqrt(R^2-(n-x0a)^2)+y0a;
                SphIde(3,m)=-sqrt(R^2-(n-x0a)^2)+y0a;
            end
            %--plot ideal circle for comparison
            plot(SphIde(1,:),SphIde(2,:),':r',SphIde(1,:),SphIde(3,:),':r')
figure(1),
            plot(HolmCoordL(1,:),HolmCoordL(2,:),'.-r');
            plot(HolmCoordR(1,:),HolmCoordR(2,:),'.-r');
            plot(x0a,y0a,'r+');
            plot(SphIde(1,:),SphIde(2,:),':r',SphIde(1,:),SphIde(3,:),':r')
drawnow;
        %--------------------------------------------------------------------------
        % Analyse RIGHT
        %--------------------------------------------------------------------------
            [GammaR,Shape,fval,OptStore]=Holm_MAIN_3('Right',ki,ImFile,PrintYN,x0a,y0a,R,HolmCoordR,GreyA,fullfile(handles.Path,folder),fig,FDisp,Data);
                %Output::Gamma=[Gamma,GammaAveF];shape=aBF=[aBest,aAveF];fval=[fvalH,fvalS,fHave];
            Results(im,6:12)=[GammaR,Shape,fval];
            fprintf(1,'Frame %d, Right - complete\n',ki);
        %--------------------------------------------------------------------------
        % Analyse LEFT
        %--------------------------------------------------------------------------
```

```
                    %Left -->code will flip image (fliplr)
                    figure(2); imshow(fliplr(GreyA)); hold on
                    plot(xL,y0a,'+r');%circle centre, flipped
                    plot(HolmCoordLr(1,:),HolmCoordLr(2,:),'r');

                    [GammaL,Shape,fval,OptStore]=Holm_MAIN_3('Left',ki,ImFile,PrintYN,
                        xL,y0a,R,HolmCoordLr,fliplr(GreyA),fullfile(handles.Path,
                        folder),fig,FDisp,Data);
                    Results(im,13:19)=[GammaL,Shape,fval];
                    Results(im,20)=R;
                    fprintf(1,'Frame %d, Left - complete\n',ki);
                    %abbreviated results file
                    fprintf(fileIDa,'% 3.7f,',Results(im,:));
                    fprintf(fileIDa,'\r\n');

                    axes(handles.axes2), hold on
                    plot((((ki-t0)/FPS)+1)/60,GammaL(1),'+b', (((ki-t0)/FPS)+1)/60,
                        GammaR(1),'+m','markersize',3)%min


                    figname=sprintf('%s-Edge',Data.ImName);
                    fullFileName = fullfile(folder, figname);
                    saveas(FDisp, [fullFileName '.png'])

                    close(h)
                %------------------------------------------------------------------
                % Analyse ENDS (if no error)
                %------------------------------------------------------------------
                catch err1 %error in analysis
                    loopEr1 = getReport(err1);
                    fprintf(LogID, '.frame %i, failed Level 1, Report: %s \r\n',ki,
                        loopEr1);
                    fprintf(1, '.frame %i, failed Level 1, Report: %s \r\n',ki,loopEr1
                        );
                    if isempty(h)==0
                        close(h)
                    end
                    NumError=NumError+1;
                end
                im=im+1;
                itr=itr+1;
            end
            Fullfilename=fullfile(folder, 'Results.mat');
try         save(Fullfilename,'Results');
catch       save('Results','Results');
end
```

```
        axes(handles.axes2), hold on
            plot(Results(:,1),Results(:,21),'+r','markersize',3)%plot gamma (aVE)
            xlabel('time (min)'); ylabel('interfacial tension (L,R BEST), mN/mm, Temp(C)');

        figure(4); hold on;
        subplot(2,3,1);hold on%plotting surface tension against frame
            plot(Results(:,1),Results(:,6),'om','markersize',3)%plot gamma (BEST)
            plot(Results(:,1),Results(:,13),'ob','markersize',3)%plot gamma (aVE)
            plot(Results(:,1),Results(:,21),'+r','markersize',3)%plot gamma (aVE)
            xlabel('frame'); ylabel('interfacial tension (BEST), mN/mm');

        subplot(2,3,2); hold on%error
            plot(Results(:,1),Results(:,10),'om','markersize',3)%plot error (full)
            plot(Results(:,1),Results(:,12),'+m','markersize',3)%plot error (ave)
            plot(Results(:,1),Results(:,17),'ob','markersize',3)%plot error (full)
            plot(Results(:,1),Results(:,19),'+b','markersize',3)%plot error (ave)
            xlabel('frame'); ylabel('error');

        subplot(2,3,4); hold on%plot shape factor against frame
            plot(Results(:,1),Results(:,8),'om','markersize',3)%plot error (full)
            plot(Results(:,1),Results(:,9),'+m','markersize',3)%plot error (ave)
            plot(Results(:,1),Results(:,15),'ob','markersize',3)%plot error (full)
            plot(Results(:,1),Results(:,16),'+b','markersize',3)%plot error (ave)
            xlabel('frame'); ylabel('shape factor');

        subplot(2,3,3); hold on %plot ave tension
            plot(Results(:,1),Results(:,7),'om','markersize',3)%plot gamma
            plot(Results(:,1),Results(:,14),'ob','markersize',3)%plot gamma
            xlabel('frame'); ylabel('Mean tension');
        subplot(2,3,5); hold on%plot density
            plot(Results(:,1),Results(:,2),'om','markersize',3)%plot rhoDiff
            plot(Results(:,1),Results(:,3),'oc','markersize',3)%plot RhoH
            plot(Results(:,1),Results(:,4),'om','markersize',3)%plot RhoL
            xlabel('frame'); ylabel('density difference');

        subplot(2,3,6); hold on
            plot(Results(:,1),Results(:,5),'oc','markersize',3)%plot temp
            plot(Results(:,1),Results(:,5),'-b')%plot temp
            xlabel('frame'); ylabel('Temperature');

    %-----------------------------------------------------------------------
    % Analyse ENDS (Sequence)
    %-----------------------------------------------------------------------


```

```
    case 'User'
        %----------------------------------------------------------------
        %--- FITTING AREAS
        Crop=Data.Crop;
        HlmCnr=Data.HlmCnr;
        SphCnr=Data.SphCnr;
        %----------------------------------------------------------------

        %----------------------------------------------------------------
        %--- ROTATION ANGLE
        Alpha=-Thta*pi/180;
        %----------------------------------------------------------------

        %----------------------------------------------------------------
        %--- TEMPERATURE FILE
        %----------------------------------------------------------------
        if get(handles.CB_Const,'value')==1
            T=str2double(get(handles.ET_Temp,'string'));
            Tc=true;
        else
            TempFile=Data.TempFile;
            load(fullfile(handles.Path,sprintf('%s.mat',TempFile)));
            Tc=false;
        end
        %----------------------------------------------------------------
        %--- LOAD SAVED EDGES
        %----------------------------------------------------------------
        if exist(fullfile(handles.Path,'UserEdges.mat'))==0
            msgbox('No pre-defined edges.  Please use "Edge GUI" to define edges.');
            return
        end

        %----------------------------------------------------------------
        %--- RESULTS FILE
        %----------------------------------------------------------------
        Result_file=sprintf('%s-Section%iframes%i-%i.txt',folder,section,ST,FTA);
        fullFileNameA = fullfile(handles.Path,folder, Result_file);
        [fileIDa,errmsg]=fopen(fullFileNameA,'a');%use a+ for reading & writing, append. Open a new file each instance

        %--abbrv file header
        fprintf(fileIDa,'Identifier:, %s, folder:, %s \r\n ,%s: , results for file right then left\r\n',Identi,folder,datestr(now));
        fprintf(fileIDa,'Program version information: %s \r\n',get(handles.ST_Version,'String'));
```

```
        fprintf(fileIDa,',Adjustment:,%i , degrees\r\n Dense fluid:, %s, Light fluid:, %s \r\n',Thta,Dense,Light);
        fprintf(fileIDa,',frame , dRho , RhoH , RhoL , T ,tens(mN/mm)(best, ave), Shape (best, ave) , error (H, S, ave F), tens(mN/mm)(best, ave), Shape (best, ave) , error (H, S, ave F) , rR, rL \r\n');

        Results=zeros(int32((FTA-ST+1)/Intv),22);%reset each time
        i=uint8(1);%assign as integer

        CFN=fullfile(folder,sprintf('Check%i.mat',section));%new file

        %===============Cleanup=================
        finishup = onCleanup(@() myCleanupFun(HFN, CFN, LogID));

        %===============Back End================
        im=1;itr=1;t0=ST;NumError=0;%May change manually
        for k = ST : Intv : FTA %subsequent images
            h=msgbox(sprintf('Analyzing frame %i (%i/%i/%i)',k,ST,Intv,FTA));
            fprintf('--- processing interval %i: %d of %d',k, i, FTA-ST+1);
            %---------------------------------------------------------------
            %---Find a frame which can be read------------------------------
            %---------------------------------------------------------------
            CntFlag=false;ki=k-1;%find a frame which can be read
            while CntFlag==false && ki<=k+Intv,%if flag is false, have not found an image to read
                ki=ki+1; %(on first entry, ki=k-1+1=k
                try
                    ImFile=fullfile(handles.Path,sprintf('%s%04i.%s',get(handles.ET_Prefix,'String'),ki,get(handles.ET_Ext,'String')));
                    disp(ImFile);
                    Grey = imread(ImFile);
                    CntFlag=true;
                catch errR
                    msgString = getReport(errR);
                    fprintf(1,'----------Unable to read frame %i. \r\n', ki);
                    fprintf(LogID,'Frame %i failed. (line 194). Report: %s',ki,msgString);
                end
            end
            %---------------------------------------------------------------
            %---Temperature & fluid densities-------------------------------
            %---------------------------------------------------------------
            if Tc==false
%               t=(ki-t0+1)*5;
                T=TC(int16((ki-t0)/FPS)+1);%C. Temp matrix is per second, from t=0 s. Movie is FPS frames/s.
```

```
        %else use constant T
        end
        [RhoH,RhoL,dRho,Flag]=DensDiff(T,Light,Dense,Data.RhoL,Data.RhoH);
        if strcmp(Flag, 'Stop')==1
            fprintf(fileIDa, 'Code terminated by user...');
            fprintf(LogID, 'Code terminated by user...');
            fprintf(1, 'Code terminated by user...');
            return %do not continue with code - error in TEMP file.
        end
        Results(im,1:5)=[ki,dRho,RhoH,RhoL,T]; % Keep temp/density info even if all frames failed
        %-------------------------------------------------------------------------
        %---Return if no frame could be read--------------------------------------
        %-------------------------------------------------------------------------
        if ki>=k+Intv && CntFlag==false,%no frame has been found
           fprintf(LogID,'*****Could not read a frame from this interval');
           im=im+1;itr=itr+1;
           continue %Go to next iteration of outer FOR loop
        end
        %-------------------------------------------------------------------------
        %---Analysis (Holm_MAIN*)-------------------------------------------------
        %-------------------------------------------------------------------------

        try
         %-------------------------------------------------------------
         % GREY IMAGE
         %-------------------------------------------------------------
            axes(handles.axes1);
             sz=size(size(Grey));
            if sz(2)>2,
                Grey=rgb2gray(Grey);
            end
            if Data.Flip==1;
                 Grey=flipud(Grey);
                 set(handles.CB_Flip,'value',1);
            end
            Cnr=Crop;
            Grey=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));

         %-------------------------------------------------------------
         % LOAD SAVED EDGES
         %-------------------------------------------------------------
            load(fullfile(handles.Path,'UserEdges'),sprintf('UsrEdge%04i',ki));
            mn=eval(genvarname(sprintf('UsrEdge%04i',ki)));

         %-------------------------------------------------------------
```

```
        % SPHERE FIT
        %---------------------------------------------------------------
            SphereCoordR=mn.SphCrdR;
            SphereCoordL=mn.SphCrdL;

            SphereCoord=[transpose(SphereCoordR),transpose(SphereCoordL)];%One layer - fit together
            [SphereOpt,fvalS]=TriSphere(SphereCoordR(1,1)-SphereCoordL(1,1),SphereCoord);%first argument is a VERY rough guess of the width
            %--Optimised sphere coordinates
            x0=SphereOpt(1);
            y0=SphereOpt(2);%adjusting x,y to main image
            plot(x0,y0,'+r')
            R=SphereOpt(3);
            if R==0; disp('Error fitting sphere profile (R==0)');end

        %---------------------------------------------------------------------------
        % HOLM EDGE
        %---------------------------------------------------------------------------
            HolmCoordL=fliplr(transpose(mn.HolmCrdL));
            HolmCoordR=fliplr(transpose(mn.HolmCrdR));

        %---------------------------------------------------------------------------
        % ROTATE COORDINATES
        %---------------------------------------------------------------------------
            if Alpha==0
                x0a=x0;
                y0a=y0;
                GreyA=Grey;
            else
                [HolmCoordL,HolmCoordR,x0a,y0a]=RotateCoords(Alpha,x0,y0,R,HolmCoordR,HolmCoordL,Grey,HL);
                GreyA=imrotate(Grey,Thta);%for plotting only
            end

            HolmCoordLr=HolmCoordL;
            HolmCoordLr(1,:)=size(GreyA,2)-HolmCoordL(1,:);

            xL=size(GreyA,2)-x0a;

            figure(FDisp), imshow(GreyA), hold on
            axes(handles.axes1); hold off, imshow(GreyA), hold on
            plot(HolmCoordL(1,:),HolmCoordL(2,:),'.-r');
            plot(HolmCoordR(1,:),HolmCoordR(2,:),'.-r');
            plot(x0a,y0a,'r+');
```

```
                    %--Theoretical points
                    SphIde=zeros(3,int32(R/5));
                    m=0;
                    for n=x0a+R:-2:x0a-R
                        m=m+1;
                        SphIde(1,m)=n;
                        SphIde(2,m)=sqrt(R^2-(n-x0a)^2)+y0a;
                        SphIde(3,m)=-sqrt(R^2-(n-x0a)^2)+y0a;
                    end
                    %--plot ideal circle for comparison
                    plot(SphIde(1,:),SphIde(2,:),':r',SphIde(1,:),SphIde(3,:),':r')
figure(1),
                    plot(HolmCoordL(1,:),HolmCoordL(2,:),'.-r');
                    plot(HolmCoordR(1,:),HolmCoordR(2,:),'.-r');
                    plot(x0a,y0a,'r+');
                    plot(SphIde(1,:),SphIde(2,:),':r',SphIde(1,:),SphIde(3,:),':r')
drawnow;
               %------------------------------------------------------------------------
               % Analyse RIGHT
               %------------------------------------------------------------------------
                    [GammaR,Shape,fval,OptStore]=Holm_MAIN_3('Right',ki,ImFile,PrintYN
                         ,x0a,y0a,R,HolmCoordR,GreyA,fullfile(handles.Path,folder),fig,
                         FDisp,Data);
                         %Output::Gamma=[Gamma,GammaAveF];shape=aBF=[aBest,aAveF];fval
                              =[fvalH,fvalS,fHave];
                    Results(im,6:12)=[GammaR,Shape,fval];
                    fprintf(1,'Frame %d, Right - complete\n',ki);
               %------------------------------------------------------------------------
               % Analyse LEFT
               %------------------------------------------------------------------------
                    %Left -->code will flip image (fliplr)
                    figure(2); imshow(fliplr(GreyA)); hold on
                    plot(xL,y0a,'+r');%circle centre, flipped
                    plot(HolmCoordLr(1,:),HolmCoordLr(2,:),'r');

                    [GammaL,Shape,fval,OptStore]=Holm_MAIN_3('Left',ki,ImFile,PrintYN,
                         xL,y0a,R,HolmCoordLr,fliplr(GreyA),fullfile(handles.Path,
                         folder),fig,FDisp,Data);
                    Results(im,13:19)=[GammaL,Shape,fval];
                    Results(im,20)=R;
                    fprintf(1,'Frame %d, Left - complete\n',ki);
                    %abbreviated results file
                    fprintf(fileIDa,'% 3.7f,',Results(im,:));
                    fprintf(fileIDa,'\r\n');
```

```
                    axes(handles.axes2), hold on
                    plot(ki,GammaL(1),'+b', ki,GammaR(1),'+m','markersize',3)%min
                %-----------------------------------------------------------------------
                % Analyse PAIR for SINT intervals
                %-----------------------------------------------------------------------
%                   if mod(im,SINT)==0 %multiple of SINT
%                       axes(handles.axes1);
%                       a2=0.5*Results(im,8)+0.5*Results(im,15);
%                       [HolmOpt,Er2S]=Holm_2sides_FixAngle(dRho,Data.Size,a2,x0a,y0a,R,HolmCoordL,HolmCoordR,GreyA,nran);
%                       Scale=Data.Size/2/R;%image scale mm/p.
%                       Gam2=dRho*9.81*1000/(HolmOpt(1)/Scale*1000)^2;
%                       Results(im,21:22)=[Gam2,HolmOpt(1)];
%                           %—image
%                       figname=sprintf('%i-Edge',k);
%                       fullFileName = fullfile(folder, figname);
%                       saveas(FDisp, [fullFileName '.png'])
%                   end
                    close(h)
                    clear(sprintf('UsrEdge%04i',ki));
                %-----------------------------------------------------------------------
                % Analyse ENDS (if no error)
                %-----------------------------------------------------------------------
                catch err1 %error in analysis
                    loopEr1 = getReport(err1);
                    fprintf(LogID, '.frame %i, failed Level 1, Report: %s \r\n',ki,loopEr1);
                    fprintf(1, '.frame %i, failed Level 1, Report: %s \r\n',ki,loopEr1);
                    close(h)
                    NumError=NumError+1;
                end
                im=im+1;
                itr=itr+1;
            end
            Fullfilename=fullfile(folder, 'Results.mat');
            save(Fullfilename,'Results');

            axes(handles.axes2), hold on
                plot(Results(:,1),Results(:,21),'+r','markersize',3)%plot gamma (aVE)
                xlabel('time (min)'); ylabel('interfacial tension (L,R BEST), mN/mm, Temp(C)');

            figure(4); hold on;
            subplot(2,3,1);hold on%plotting surface tension against frame
                plot(Results(:,1),Results(:,6),'om','markersize',3)%plot gamma (BEST)
```

```
            plot(Results(:,1),Results(:,13),'ob','markersize',3)%plot gamma (aVE)
            plot(Results(:,1),Results(:,21),'+r','markersize',3)%plot gamma (aVE)
            xlabel('frame'); ylabel('interfacial tension (BEST), mN/mm');

        subplot(2,3,2); hold on%error
            plot(Results(:,1),Results(:,10),'om','markersize',3)%plot error (full)
            plot(Results(:,1),Results(:,12),'+m','markersize',3)%plot error (ave)
            plot(Results(:,1),Results(:,17),'ob','markersize',3)%plot error (full)
            plot(Results(:,1),Results(:,19),'+b','markersize',3)%plot error (ave)
            xlabel('frame'); ylabel('error');

        subplot(2,3,4); hold on%plot shape factor against frame
            plot(Results(:,1),Results(:,8),'om','markersize',3)%plot error (full)
            plot(Results(:,1),Results(:,9),'+m','markersize',3)%plot error (ave)
            plot(Results(:,1),Results(:,15),'ob','markersize',3)%plot error (full)
            plot(Results(:,1),Results(:,16),'+b','markersize',3)%plot error (ave)
            xlabel('frame'); ylabel('shape factor');

        subplot(2,3,3); hold on %plot ave tension
            plot(Results(:,1),Results(:,7),'om','markersize',3)%plot gamma
            plot(Results(:,1),Results(:,14),'ob','markersize',3)%plot gamma
            xlabel('frame'); ylabel('Mean tension');
        subplot(2,3,5); hold on%plot density
            plot(Results(:,1),Results(:,2),'om','markersize',3)%plot rhoDiff
            plot(Results(:,1),Results(:,3),'oc','markersize',3)%plot RhoH
            plot(Results(:,1),Results(:,4),'om','markersize',3)%plot RhoL
            xlabel('frame'); ylabel('density difference');

        subplot(2,3,6); hold on
            plot(Results(:,1),Results(:,5),'oc','markersize',3)%plot temp
            plot(Results(:,1),Results(:,5),'-b')%plot temp
            xlabel('frame'); ylabel('Temperature');

    %------------------------------------------------------------------------
    % Analyse ENDS (Sequence with user edge)
    %------------------------------------------------------------------------
end
Fullfilename=fullfile(handles.Path,folder, 'Results.mat');
save(Fullfilename,'Results');

%SAVE results
%--mat
MatName=sprintf('Workspace%d.mat',section);
MatNameFull=fullfile(handles.Path,folder,MatName);
save(MatNameFull);%save workspace
%Gamma_Dev=standarddev(Results(:,2));
```

```

fprintf(fileIDa, 'Code terminated normally...Number of failer frames: %i',NumError
    );
fprintf(1, 'Code terminated normally...Number of failer frames: %i. \r\n Folder: %
    s',NumError, folder);
fclose(fileIDa);%close results file

beep
beep
%----------------------------------------------------------------------------


% --- Executes on button press in PB_Results.
function PB_Results_Callback(hObject, eventdata, handles)
%----------------------------------------------------------------------------
%-- SHOW RESULTS
%----------------------------------------------------------------------------


% --- Executes on button press in PB_Save.
function PB_Save_Callback(hObject, eventdata, handles)
%----------------------------------------------------------------------------
%-- SAVE MATRIX TO RECALL LATER
%----------------------------------------------------------------------------
Tag=char(get(handles.ET_Tag,'String'));
%.. Image properties
Data.ImType=handles.ImType;
switch(handles.ImType)
    case 'Sequence'
        Data.Prefix=get(handles.ET_Prefix,'String');
        Data.Ext=get(handles.ET_Ext,'String');
        Data.ImName=sprintf('%s%04i.%s',Data.Prefix,str2double(get(handles.ET_ST,'
            String')),Data.Ext);
        Data.Path=handles.Path;
    case 'User'
        Data.Prefix=get(handles.ET_Prefix,'String');
        Data.Ext=get(handles.ET_Ext,'String');
        Data.ImName=sprintf('%s%04i.%s',Data.Prefix,str2double(get(handles.ET_ST,'
            String')),Data.Ext);
        Data.Path=handles.Path;
    otherwise
        Data.ImName=handles.ImName;
end
Data.Tag=get(handles.ET_Tag,'String');
Data.SphCnr=handles.SphCnr;
Data.HlmCnr=handles.HlmCnr;
```

```
Data.Flip=get(handles.CB_Flip,'value');

Data.Crop=handles.Crop;
%.. Physical properies
Data.light=handles.light;
Data.RhoL=handles.RhoL;
Data.dense=handles.dense;
Data.RhoH=handles.RhoH;
Data.dRho=handles.dRho;
Data.Temp=str2double(get(handles.ET_Temp,'String'));
Data.Size=str2double(get(handles.ET_BallWidth,'String'));
Data.GamEst=str2double(get(handles.ET_GamEst,'String'));
Data.TempFile=handles.TempFile;
%.. Analysis info
Data.ST=str2double(get(handles.ET_ST,'String'));
Data.End=str2double(get(handles.ET_End,'String'));
Data.Int=str2double(get(handles.ET_Intv,'String'));
Data.Ref=str2double(get(handles.ET_Ref,'String'));
%.. Optimisation info
Data.AngleLow=str2double(get(handles.ET_AngleLow,'String'));
Data.AngleHigh=str2double(get(handles.ET_AngleHigh,'String'));
Data.ThetaAdjLow=str2double(get(handles.ET_ThetaAdjLow,'String'));
Data.ThetaAdjHigh=str2double(get(handles.ET_ThetaAdjHigh,'String'));
Data.SL=str2double(get(handles.ET_SL,'String'));
Data.N=str2double(get(handles.ET_N,'String'));
Data.nRan=str2double(get(handles.ET_nRan,'String'));
Data.SINT=str2double(get(handles.ET_SINT,'String'));
Data.Tol=str2double(get(handles.ET_Tol,'String'));
%.. Angles - from GUI
Data.Theta=handles.Theta;
Data.Alpha=handles.Alpha;


%..Notes
Notes=get(handles.ET_Notes,'String');

file=fullfile(handles.Path,sprintf('%s-Data.mat',Tag));
Grey=handles.Grey;
save(file,'Data','Notes','Grey');
disp('Data saved');
%----------------------------------------------------------------------


% --- Executes on button press in PB_Angle.
function PB_Angle_Callback(hObject, eventdata, handles)
%----------------------------------------------------------------------
```

```
% DETERMINE THE REQUIRED ANGLE ADJUSTMENT
%--------------------------------------------------------------------------
Tag=char(get(handles.ET_Tag,'Value'));
%.. Call angle GUI
h=HolmGUIangle12b; uiwait(h);

%.. Update main GUI
GUI_Hmain=getappdata(0,'GUI_Hmain');
handles.Alpha=getappdata(GUI_Hmain,'Alpha');
handles.Theta=getappdata(GUI_Hmain,'Theta');
handles.HL=getappdata(GUI_Hmain,'HL');

set(handles.ST_adj,'String',sprintf('Adj: %0.4f (deg)',handles.Theta));
%load(File);
guidata(hObject, handles);
%--------------------------------------------------------------------------


function ET_Tag_Callback(hObject, eventdata, handles)
%--------------------------------------------------------------------------
%--- TAG
%.. Set tag
    GUI_Hmain=getappdata(0,'GUI_Hmain');
    setappdata(GUI_Hmain,'Tag',char(get(handles.ET_Tag,'String')));
%--------------------------------------------------------------------------

% --- Executes during object creation, after setting all properties.
function ET_Tag_CreateFcn(hObject, eventdata, handles)
%--------------------------------------------------------------------------
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%--------------------------------------------------------------------------


function ET_GamEst_Callback(hObject, eventdata, handles)
% hObject    handle to ET_GamEst (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_GamEst as text
%        str2double(get(hObject,'String')) returns contents of ET_GamEst as a double


```

```
% --- Executes during object creation, after setting all properties.
function ET_GamEst_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_GamEst (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_AngleLow_Callback(hObject, eventdata, handles)
% hObject    handle to ET_AngleLow (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_AngleLow as text
%        str2double(get(hObject,'String')) returns contents of ET_AngleLow as a
    double


% --- Executes during object creation, after setting all properties.
function ET_AngleLow_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_AngleLow (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_AngleHigh_Callback(hObject, eventdata, handles)
% hObject    handle to ET_AngleHigh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_AngleHigh as text
```

```
%        str2double(get(hObject,'String')) returns contents of ET_AngleHigh as a
     double


% --- Executes during object creation, after setting all properties.
function ET_AngleHigh_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_AngleHigh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_ThetaAdjLow_Callback(hObject, eventdata, handles)
% hObject    handle to ET_ThetaAdjLow (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_ThetaAdjLow as text
%        str2double(get(hObject,'String')) returns contents of ET_ThetaAdjLow as a
      double


% --- Executes during object creation, after setting all properties.
function ET_ThetaAdjLow_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_ThetaAdjLow (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_ThetaAdjHigh_Callback(hObject, eventdata, handles)
% hObject    handle to ET_ThetaAdjHigh (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_ThetaAdjHigh as text
%        str2double(get(hObject,'String')) returns contents of ET_ThetaAdjHigh as
    a double


% --- Executes during object creation, after setting all properties.
function ET_ThetaAdjHigh_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_ThetaAdjHigh (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_BallWidth_Callback(hObject, eventdata, handles)
% hObject    handle to ET_BallWidth (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_BallWidth as text
%        str2double(get(hObject,'String')) returns contents of ET_BallWidth as a
    double


% --- Executes during object creation, after setting all properties.
function ET_BallWidth_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_BallWidth (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

```

```

% --- Executes on button press in checkbox2.
function checkbox2_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox2



function ET_End_Callback(hObject, eventdata, handles)
handles.End=str2double(get(hObject,'String'));

% --- Executes during object creation, after setting all properties.
function ET_End_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_End (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in checkbox3.
function checkbox3_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox3



function ET_Intv_Callback(hObject, eventdata, handles)
handles.Intv=str2double(get(hObject,'String'));


% --- Executes during object creation, after setting all properties.
function ET_Intv_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_Intv (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_ST_Callback(hObject, eventdata, handles)
handles.ST=str2double(get(hObject,'String'));
disp('ST updated');
guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function ET_ST_CreateFcn(hObject, eventdata, handles)
%------------------------------------------------------------
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%------------------------------------------------------------
%------------------------------------------------------------

function ET_Fps_Callback(hObject, eventdata, handles)
%------------------------------------------------------------
%------------------------------------------------------------

% --- Executes during object creation, after setting all properties.
function ET_Fps_CreateFcn(hObject, eventdata, handles)
% hObject    handle to et_fps (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes when selected object is changed in ImTypePanel.
```

```
function ImTypePanel_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to the selected object in ImTypePanel
%------------------------------------------------------------------------
%--- SELECT IMAGE TYPE (Movie/Single image)
%.. Get image type
    ImType = get(hObject,'String');

%.. If Single, disable movie parameters (Fps, ST, End, Int) & set TotalFrames = 1. If Movie, enable them.
    switch(ImType)
        case 'Sequence'
            set(handles.ET_Prefix,'enable','on')
            set(handles.ET_Ext,'enable','on')
            set(handles.ET_ST,'enable','on');
            set(handles.ET_End,'enable','on');
            set(handles.ET_Intv,'enable','on');
            set(handles.ET_Fps,'enable','on');%most likely a fraction
            set(handles.PB_Temp,'enable','on');
            set(handles.DD_Temp,'enable','on');
            set(handles.ET_Ref,'enable','on');
            set(handles.CB_Const,'Value',0);
            set(handles.CB_Print,'Value',0);

        case 'User'
            set(handles.ET_Prefix,'enable','on')
            set(handles.ET_Ext,'enable','on')
            set(handles.ET_ST,'enable','on');
            set(handles.ET_End,'enable','on');
            set(handles.ET_Intv,'enable','on');
            set(handles.ET_Fps,'enable','on');%most likely a fraction
            set(handles.PB_Temp,'enable','on');
            set(handles.DD_Temp,'enable','on');
            set(handles.ET_Ref,'enable','off');
            set(handles.CB_Const,'Value',0);
            set(handles.CB_Print,'Value',0);

        case 'Still'
            set(handles.ET_Fps,'enable','off');
            set(handles.ET_ST,'enable','off');
            set(handles.ET_End,'enable','off');
            set(handles.ET_Intv,'enable','off');
            set(handles.PB_Temp,'enable','off');
            set(handles.DD_Temp,'enable','off');
            set(handles.ET_Ref,'enable','off');
            set(handles.CB_Const,'Value',1);
            set(handles.ST_TotalFrames,'String','1');
```

```
                set(handles.ET_Prefix,'enable','off')
                set(handles.ET_Ext,'enable','off')
                set(handles.CB_Print,'Value',1);

            case 'Movie'
                set(handles.ET_Fps,'enable','on');
                set(handles.ET_ST,'enable','on');
                set(handles.ET_End,'enable','on');
                set(handles.ET_Intv,'enable','on');
                set(handles.PB_Temp,'enable','on');
                set(handles.DD_Temp,'enable','on');
                set(handles.ET_Ref,'enable','off');
                set(handles.CB_Const,'Value',0);
                set(handles.ST_TotalFrames,'String','Load...');
                set(handles.ET_Prefix,'enable','off')
                set(handles.ET_Ext,'enable','off')
                set(handles.CB_Print,'Value',0);
    end
%.. Save ImType
    handles.ImType = ImType;
    guidata(hObject, handles);
%------------------------------------------------------------


% --- Executes during object creation, after setting all properties.
function ImTypePanel_CreateFcn(hObject, eventdata, handles)
%------------------------------------------------------------
    ImType = get(get(hObject,'SelectedObject'),'String');

    handles.ImType = ImType;
    guidata(hObject, handles);
%------------------------------------------------------------


% --- Executes on button press in PB_Cnrs.
function PB_Cnrs_Callback(hObject, eventdata, handles)
%------------------------------------------------------------
Grey=handles.Grey;
%--- REDEFINE FITTING AREAS
    %..Select axis
    axes(handles.axes1); hold off
    %..Call new image (use new ST)
    if exists(handles.ImName)
        DropMovie = VideoReader(handles.ImName);
        Grey = read(DropMovie, ST);
    else %sequence
```

```
        ImName=sprintf('%s%04i.%s',handles.Prefix,handles.ST,handles.Ext);
        ImName=fullfile(handles.Path,ImName);
        Grey = imread(ImName);
    end
    imshow(Grey); hold on

    fprintf(1,'Selecting an area outside of image boundaries will return an error\n')
    fprintf(1,'Please select the area to fit for the holm (right)\n(click and drag box):\n')
    rec=round(getrect());
    HlmCnrR=[rec(1),rec(2),rec(1)+rec(3),rec(2)+rec(4)];
    if HlmCnrR(3)>size(Grey,2);HlmCnrR(3)=size(Grey,2);end
    if HlmCnrR(4)>size(Grey,1);HlmCnrR(4)=size(Grey,1);end

    fprintf(1,'Please select the area to fit for the holm (left)\n(click and drag box):\n')
    rec=round(getrect());
    HlmCnrL=[rec(1),rec(2),rec(1)+rec(3),rec(2)+rec(4)];
    if HlmCnrL(3)>size(Grey,2);HlmCnrL(3)=size(Grey,2);end
    if HlmCnrL(4)>size(Grey,1);HlmCnrL(4)=size(Grey,1);end

    HlmCnr=[HlmCnrR;HlmCnrL];

    fprintf(1,'Please select the area to fit for the sphere (right) \n(click and drag box):\n')
    rec=round(getrect());
    SphCnr(1,:)=[rec(1),rec(2),rec(1)+rec(3),rec(2)+rec(4)];

    fprintf(1,'Please select the area to fit for the sphere (left) \n(click and drag box):\n')
    rec=round(getrect());
    SphCnr(2,:)=[rec(1),rec(2),rec(1)+rec(3),rec(2)+rec(4)];

    %Display boxes
    plot([HlmCnrR(1),HlmCnrR(3),HlmCnrR(3),HlmCnrR(1),HlmCnrR(1)],[HlmCnrR(2),HlmCnrR(2),HlmCnrR(4),HlmCnrR(4),HlmCnrR(2)],'r');
    plot([HlmCnrL(1),HlmCnrL(3),HlmCnrL(3),HlmCnrL(1),HlmCnrL(1)],[HlmCnrL(2),HlmCnrL(2),HlmCnrL(4),HlmCnrL(4),HlmCnrL(2)],'r');
    plot([SphCnr(1,1),SphCnr(1,3),SphCnr(1,3),SphCnr(1,1),SphCnr(1,1)],[SphCnr(1,2),SphCnr(1,2),SphCnr(1,4),SphCnr(1,4),SphCnr(1,2)],'r');
    plot([SphCnr(2,1),SphCnr(2,3),SphCnr(2,3),SphCnr(2,1),SphCnr(2,1)],[SphCnr(2,2),SphCnr(2,2),SphCnr(2,4),SphCnr(2,4),SphCnr(2,2)],'r');

    %Save
    handles.HlmCnr=HlmCnr;
```

```
    handles.SphCnr=SphCnr;
    guidata(hObject, handles);
%--------------------------------------------------------------------
%--------------------------------------------------------------------


function [RhoH,RhoL,dRho,Flag]=DensDiff(T,Light,Dense,RhoL,RhoH)
%--------------------------------------------------------------------
% CALCULATE DENSITY DIFFERENCE
%--------------------------------------------------------------------
%calculates the density difference based on temperature.
%Temperature in Celsius
%Salt-water (NaCl) density data provided by Yusuke Asakuma."NaCL_density.xlsx)
%   --NaCL density data,0-40 degrees C
Flag = 'Continue';

%check temp OK :: flag if T is out of bounds and ask user if continuing.
if isnan(T)==1 || T==0;
    Flag = questdlg(sprintf('Check temperature! \n T = %f \n I recommend that you
        stop.',T),'WARNING','Continue','Stop','Stop');
end
%--------------------------------------------------------------------
%--- Light fluid
switch Light
    case 'Decane'
        RhoL=228.2*0.247^(-(1-(T+273.13)/616)^(2/7));%from Himmenbleau and Riggs (
            original gives g/cm3, *1000
    case 'Dodecane'
        h=msgbox('not done yet')
        uiwait(h);
    case 'Air'
        RhoL=1.2;
    case 'Water'
        RhoL=(999.83952+T*(16.945176+T*(-7.9870401e-3+T*(-46.170461e-6+T
            *(105.56302e-9-280.54253e-12*T)))))/(1+T*16.87985e-3);%SysCad steam
            properties: http://help.syscad.net/index.php/
            Water_and_Steam_Properties
    case 'Other'
        %no change
end
%--------------------------------------------------------------------
%--- Heavy fluid
switch Dense
    case 'Water' %pure water
```

```
            RhoH=(999.83952+T*(16.945176+T*(-7.9870401e-3+T*(-46.170461e-6+T*(105.56302e-9-280.54253e-12*T)))))/(1+T*16.87985e-3);%SysCad steam properties: http://help.syscad.net/index.php/Water_and_Steam_Properties
        case 'Brine-0.1 NaCl' %0.1 mol/l
            RhoH=(-5.3478e-6*T^2+4.9629e-6*T+1.0048)*1000;
        case 'Brine-0.01 NaCl'%0.01mol/l
            RhoH=(-5.5488e-6*T^2+2.0921e-5*T+1.001)*1000;
        case 'Brine-0.001 NaCl'%0.001mol/l
            RhoH=(-5.5692e-6*T^2+2.2535e-5*T+1.000)*1000;
end
dRho=RhoH-RhoL;
%-----------------------------------------------------------------


% --- Executes during object creation, after setting all properties.
function ST_dRho_CreateFcn(hObject, eventdata, handles)
%-----------------------------------------------------------------
%--- SET DENSITY DIFFERENCE
%-----------------------------------------------------------------
T=20; %default
RhoL=228.2*0.247^(-(1-(T+273.13)/616)^(2/7));%from Himmenbleau and Riggs (original gives g/cm3, *1000
RhoH=(999.83952+T*(16.945176+T*(-7.9870401e-3+T*(-46.170461e-6+T*(105.56302e-9-280.54253e-12*T)))))/(1+T*16.87985e-3);%SysCad steam properties: http://help.syscad.net/index.php/Water_and_Steam_Properties

set(hObject,'String',sprintf('%0.4f',RhoH-RhoL));
handles.dRho=RhoH-RhoL;
guidata(hObject, handles);
%-----------------------------------------------------------------



function ET_N_Callback(hObject, eventdata, handles)
%-----------------------------------------------------------------
%..Check to ensure that N>=2.  N<2 will throw an error.
%-----------------------------------------------------------------
N=str2double(get(hObject,'String'));% returns contents of ET_N as a double
if N<2,
    disp('N must be greater than 2. Setting as default (2).');
    set(hObject,'String','2');
end
%-----------------------------------------------------------------
```

```
% --- Executes during object creation, after setting all properties.
function ET_N_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_N (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_nRan_Callback(hObject, eventdata, handles)
% hObject    handle to ET_nRan (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_nRan as text
%        str2double(get(hObject,'String')) returns contents of ET_nRan as a double


% --- Executes during object creation, after setting all properties.
function ET_nRan_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_nRan (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_SL_Callback(hObject, eventdata, handles)
% hObject    handle to ET_SL (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_SL as text
%        str2double(get(hObject,'String')) returns contents of ET_SL as a double
```

```



% --- Executes during object creation, after setting all properties.
function ET_SL_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_SL (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_Tol_Callback(hObject, eventdata, handles)
% hObject    handle to ET_Tol (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_Tol as text
%        str2double(get(hObject,'String')) returns contents of ET_Tol as a double


% --- Executes during object creation, after setting all properties.
function ET_Tol_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_Tol (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit19_Callback(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

```

```
% Hints: get(hObject,'String') returns contents of edit19 as text
%        str2double(get(hObject,'String')) returns contents of edit19 as a double


% --- Executes during object creation, after setting all properties.
function edit19_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_SINT_Callback(hObject, eventdata, handles)
% hObject    handle to ET_SINT (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_SINT as text
%        str2double(get(hObject,'String')) returns contents of ET_SINT as a double


% --- Executes during object creation, after setting all properties.
function ET_SINT_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_SINT (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in DD_Temp.
function DD_Temp_Callback(hObject, eventdata, handles)
%-------------------------------------------------------------------------
%--- UPDATE TEMPERATURE FILE
```

```
%------------------------------------------------------------
contents = cellstr(get(hObject,'String'));
handles.TempFile=contents{get(hObject,'Value')};
guidata(hObject, handles);
%------------------------------------------------------------


% --- Executes during object creation, after setting all properties.
function DD_Temp_CreateFcn(hObject, eventdata, handles)
%------------------------------------------------------------
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%------------------------------------------------------------
%--- INITIAL - TEMPERATURE TYPE
%------------------------------------------------------------
handles.TempFile='TC1';
guidata(hObject, handles);
%------------------------------------------------------------


%% TRI SPHERE
function [SphereOpt,fvalS]=TriSphere(guessW,SphereCoord)
%------------------------------------------------------------------------------%
%FIT CIRCLE (OPTIMISATION)
%------------------------------------------------------------------------------%
    Guess=[guessW,guessW,guessW];%x0,y0,R
    Opts = optimset('Display','final','TolFun',1e-8,'Algorithm','active-set','LargeScale','on','MaxFunEval',1000);
    %                   x = fmincon(fun,                          x0,    A,b,Aeq,beq,lb,ub,nonlcon,options)
    [SphereOpt,fvalS,exitFlag,~]= fmincon(@(Guess)objTRI(Guess,SphereCoord),Guess,[],[],[],[],[-inf,-inf,0.01],[],[],Opts);
    switch exitFlag
        case 1
            disp('fmincon has converged properly.')
        case 0
            disp('fmincon has reached the max. number of iterations. Function may not have converged.')
        case -2
            disp('fmincon was unable to find a solution.')
    end
%------------------------------------------------------------------------------%


```

```
% SPHERE OBJECTIVE
function eMin=objTRI(Guess,SphereCoord)
%------------------------------------------------------------------------%
%Objective function for circle profile oftimisation
%------------------------------------------------------------------------%
    x0=Guess(1);
    y0=Guess(2);
    R=Guess(3);

    Error2=zeros(1,size(SphereCoord,2));

    for j=1:size(SphereCoord,2)
        Error2(j)=sqrt(((SphereCoord(1,j)-x0)^2+(SphereCoord(2,j)-y0)^2-R^2)^2);
    end
    eMin=sqrt(sum(Error2))/size(SphereCoord,2);
%------------------------------------------------------------------------%


%% TRI-CLEAN
function [Coord]=triClean(Edges,min)
%------------------------------------------------------------------------%
% PICK EDGES using Canny edge detection method
% -- Edges = CLEAN edge matrix (white edges on black)
% -- min   = minimum area to count (pixles) (discard with lower area)
%------------------------------------------------------------------------%

 %--Label matrix and stats.
    [L,num]=bwlabel(Edges);%repeat?
    Stats=regionprops(L,'Area','PixelIdxList');

 %--Sort by area (descending)
    Area=zeros(num,2);
    for R=1:num %region
        Area(R,:)=[R,Stats(R).Area];
    end %R
    Area=flipud(sortrows(Area,2));%sort based on area.  flipup(ascending)=descending.

 %--Find cut-off for minimum area (min)
    R=1;
    while R<=num&&Area(R,2)>=min,
        R=R+1;
    end
    R=R-1;
    RegionsOfInterest=Area(1:R,1);

```

```
 %--Combine all as linear indices
    IND=[];
    for k=1:R
        R=RegionsOfInterest(k,1);
        IND=[IND;Stats(R).PixelIdxList];%linear indices
    end %K = R plot

 %--Convert Linear indices, order
    s=size(Edges);%size of 'grey' image for converting linear indices
    [y,x]=ind2sub(s,IND);
    Coord=[transpose(x);transpose(y)];
%--------------------------------------------------------------------------%


%--------------------------------------------------------------

function [CoordL,CoordR,x0a,y0a]=RotateCoords(Alpha,x0,y0,R,HolmCoordR,HolmCoordL,Grey,HL)
    %---------------------------------------------------------------------
    % rotate Coordinates
    %---------------------------------------------------------------------
    %--Main image and angles
    Thta=-Alpha*180/pi;
    imcentre=size(Grey)/2;
    GreyA=imrotate(Grey,Thta);%for plotting only
    imcentre2=size(GreyA)/2;


    %--Rotate sphere centre (x0,y0);
    xt=x0-imcentre(2);
    yt=y0-imcentre(1);
    Rs=sqrt(xt^2+yt^2);

    if xt<0
        th=atan(yt/xt);
        x0a=-Rs*cos(th+Alpha)+imcentre2(2);
    elseif xt>0
        th=atan(yt/xt);
        x0a=Rs*cos(th+Alpha)+imcentre2(2);
    else %xt=0 >> phi = 90 deg
        th=pi/2;
        x0a=Rs*cos(th+Alpha)+imcentre2(2);
    end

    if strcmp(HL,'Low')==1
        y0a=-Rs*sin(th+Alpha)+imcentre2(1);
```

```
    else
        y0a=Rs*sin(th+Alpha)+imcentre2(1);
    end


    %--Rotate holm coordinates
    HCRA=zeros(4,size(HolmCoordR,2));
    HCLA=zeros(4,size(HolmCoordL,2));

    %...right
    transMat=HolmCoordR;
    transMat(1,:)=HolmCoordR(1,:)-imcentre(2);%translate to centre
    transMat(2,:)=HolmCoordR(2,:)-imcentre(1);
    for c=1:size(HolmCoordR,2)
        HCRA(3,c)=sqrt(transMat(1,c)^2+transMat(2,c)^2);%R
        HCRA(4,c)=atan(transMat(2,c)/transMat(1,c));%Thta
        HCRA(1,c)=HCRA(3,c)*cos(HCRA(4,c)+Alpha);%x'
        HCRA(2,c)=HCRA(3,c)*sin(HCRA(4,c)+Alpha);%y'
    end
    HCRA(1,:)=HCRA(1,:)+imcentre2(2);%translate to centre
    HCRA(2,:)=HCRA(2,:)+imcentre2(1);

    %...left
    alphaL=Alpha-pi;

    transMat=HolmCoordL;
    transMat(1,:)=HolmCoordL(1,:)-imcentre(2);%translate to centre
    transMat(2,:)=HolmCoordL(2,:)-imcentre(1);

    for c=1:size(HolmCoordL,2)
        HCLA(3,c)=sqrt(transMat(1,c)^2+transMat(2,c)^2);%R
        HCLA(4,c)=atan(transMat(2,c)/transMat(1,c));%Thta
        HCLA(1,c)=HCLA(3,c)*cos(HCLA(4,c)+alphaL);%x'
        HCLA(2,c)=HCLA(3,c)*sin(HCLA(4,c)+alphaL);%y'
    end

    HCLA(1,:)=HCLA(1,:)+imcentre2(2);%translate to centre
    HCLA(2,:)=HCLA(2,:)+imcentre2(1);

CoordL=HCLA(1:2,:); CoordR=HCRA(1:2,:);
%     figure(1)
%     imshow(GreyA); hold on
%     plot(HCLA(1,:),HCLA(2,:),'r');
%     plot(HCRA(1,:),HCRA(2,:),'r');
%     plot(x0a,y0a,'ob');
%--------------------------------------------------------------
```

```


% --- Executes during object creation, after setting all properties.
function ST_Version_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ST_Version (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called



%--------------------------------------------------------------------
% Handles for ImMask and Tri are sent to ThreshGUI
%--------------------------------------------------------------------
function [WIM,ThreshC,Thr]=WholeImageMask(Grey,BWadj,CANadjL,CANadjH,MorphProps,
    Flag,Path)
%MorphProps.size
%MorphProps.Type
%MorphProps.FilLoc
%--------------------------------------------------------------------
    %GENERATE A BW MASK OVER THE WHOLE IMAGE
%--------------------------------------------------------------------
%.. Thresholds
    Thr=BWadj*graythresh(Grey);
    if isempty(CANadjL)==1||isempty(CANadjH)==1 %if canny parameters aren't set
        [Edg,ThreshC]=edge(Grey,'canny');
        CANadjL=ThreshC(1);
        CANadjH=ThreshC(2);
    else
        if CANadjL>=CANadjH
            ThreshC=[0.95*CANadjH,CANadjH];
        else
            ThreshC=[CANadjL,CANadjH];
        end
    end

%--------------------------------------------------------------------
%--- If the threshold is too low, no edges will be detected. Matrix will be
    %blank causing the mask to obscure all edges.
    % imshow(im2bw(Grey,0)) == while matrix (all ones)
    %The "don't use mask" toggle where Thresh=0 will also trigger this and
    %will send a blank matrix to fn(Tri).
%--- If BW matrix is blank, generate blank mask.
%--- If not blank, generate mask
%--- dual fitting areas (sphere or holm)
%--------------------------------------------------------------------

```

```
%======================================================================
%-- Whole image mask
%...The drop and sphere should be the largest foreground object
%...Could have a problem with poor lighting if the drop and sphere turn out
%...as separate objects.
%======================================================================
%.. BW image
    BW1=im2bw(Grey,Thr);
    BW2=imfill(~BW1,'holes');%fill holes
    SE3 = strel(MorphProps.Type,MorphProps.Size);%Create structural element
    BW3=imclose(BW2,SE3);%close image
    if isempty(MorphProps.FilLoc)==1
        BW4=BW3;
    else %flood fill
        BW4=imfill(BW3,MorphProps.FilLoc);%flood fill from user's selected points.
    end
%.. Identify the largest object
    CC=bwconncomp(BW4);
    L=labelmatrix(CC);%Create label matrix
    %imshow(label2rgb(L))
    Stats = regionprops(L,'Area');%get region properties
    if isempty(Stats)==1 %use default thresholds
        disp('Mask failed. Using default thresholds.')
        %.. BW image
        BW1=im2bw(Grey);
        BW2=imfill(~BW1,'holes');%fill holes
        BW3=imclose(BW2,SE3);%close image
        if isempty(MorphProps.FilLoc)==1
            BW4=BW3;
        else %flood fill
            BW4=imfill(BW3,MorphProps.FilLoc);%flood fill from user's selected
                points.
        end
        %.. Identify the largest object
        CC=bwconncomp(BW4);
        L=labelmatrix(CC);%Create label matrix
        %imshow(label2rgb(L))
        Stats = regionprops(L,'Area');%get region properties
    end
    [~,iMax] = max([Stats.Area]);%Identify largest region

    WIM=zeros(size(BW4));%create blank mask
    WIM(L==iMax)=1;%Add largest forground object to mask
    WIM=imfill(WIM,'holes');%close any remaining holes.
    %.. For Thresh GUI
    if Flag==1
```

```
        save(fullfile(Path,'ThreshTemp1'));
    end


%% MASK (generate mask for edge detection) %%% Call function handle from main GUI
function [CrdL,CrdR]=ImMask(CnrL,CnrR,Grey,ThreshC,Thr,SvFlag,wd,SBD,WIM,Path,NL,
    BMask)

    [W3]=WhiteMask(Thr,Grey,3);
    ThrB=0.05; Aug=1.5;

%--------------------------------------------------------------------
% LEFT MASK & CORDINATES
%--------------------------------------------------------------------
    Cnr=CnrL;
    try
        GreyL=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));
    catch
        disp('Size misspatch, LHS')
        Cnr=[CnrL(1),size(Grey,2),CnrL(3),size(Grey,1)];
    end
%MASK EDGES/SIDES (boundaries already defined)

%.. If empty matrix, 1 -> 0
    if Thr==0
        maskL=zeros(size(GreyL));%black
    else
%.. If BW is not empty:
    %---Find perimeter of mask
        PerimL=bwperim(WIM(Cnr(2):Cnr(4),Cnr(1):Cnr(3)));
    %---remove border
        PerimL(:,1)=0;
        PerimL(1,:)=0;
        PerimL(:,end)=0;
        PerimL(end,:)=0;
    %---Dilate perimeter
        maskL = ~imdilate(PerimL,strel('disk',wd),'same');%
    end

%.. Edge detection
[EdgeL]=Tri(GreyL,maskL,ThreshC,2*SBD,NL);%Canny edge detection (Edges, y1,x1)

%.. Bubble mask
if BMask == 1
    [BubMask]=CircMask(GreyL,ThrB,Aug);%BW mask, bubbles are white
    CleanL=EdgeL;
```

```
        CleanL(BubMask(:,1:end-1)==1)=0;
        %Red=GreyL;
        %Red(BubMask==1)=0.5;
        %imshow(cat(3, GreyL,Red,Red));
 else
     CleanL=EdgeL;
end
%.. White mask
W=W3(Cnr(2):Cnr(4),Cnr(1):Cnr(3));
CleanL(W(:,1:end-1)==1)=0;

%-------------------------------------------------------
%--Display matrix for bubble and white mask.
% figure(5);
% DispM=CleanL;
% DispM(W(:,1:end-1)==1)=2;
% DispM(BubMask(:,1:end-1)==1)=3;
% DispM(EdgeL==1)=1;
% imshow(label2rgb(DispM));
%-------------------------------------------------------

 if max(max(CleanL))==0
        disp('No edge found.')
end

CleanSt = regionprops(CleanL,'PixelIdxList');
[y1,x1]=ind2sub(size(CleanL),CleanSt.PixelIdxList);
CrdL=sortrows([x1,y1],2);%sort rows
 if isempty(y1)==1
        fprintf('Edge detection has failed (left) - no edge detected. Continuing to next frame.')
end     %....Debugging
        %imshow(maskL+EdgeL)
        %plot(CrdL(:,1),CrdL(:,2),'r')

%.. Coordinates
CrdL(:,1)=CrdL(:,1)+CnrL(1);
CrdL(:,2)=CrdL(:,2)+CnrL(2);
%plot(CrdL(:,1),CrdL(:,2),'r')

%------------------------------------------------------------------------
% RIGHT MASK & CORDINATES
%------------------------------------------------------------------------
Cnr=CnrR;
GreyR=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));

```

```
%.. If empty matrix, 1 -> 0
if Thr==0
    maskR=zeros(size(GreyR));
else
%.. If BW is not empty:
    %---Find perimeter of mask
        PerimR=bwperim(WIM(Cnr(2):Cnr(4),Cnr(1):Cnr(3)));
    %---remove border
        PerimR(:,1)=0;
        PerimR(1,:)=0;
        PerimR(:,end)=0;
        PerimR(end,:)=0;
    %---Dilate perimeter
        maskR = ~imdilate(PerimR,strel('disk',wd),'same');%
end
%.. Edge detection
[EdgeR]=Tri(GreyR,maskR,ThreshC,SBD,NL);%Canny edge detection (Edges, y1,x1)

%.. Bubble mask
if BMask == 1
    [BubMask]=CircMask(GreyR,ThrB,Aug);%BW mask, bubbles is white

    CleanR=EdgeR;
    CleanR(BubMask(:,1:end-1)==1)=0;
else
    CleanR=EdgeR;
end
%.. White mask
W=W3(Cnr(2):Cnr(4),Cnr(1):Cnr(3));
CleanR(W(:,1:end-1)==1)=0;

%.. Coordinates
CleanSt = regionprops(CleanR,'PixelIdxList');
[y2,x2]=ind2sub(size(GreyR),CleanSt.PixelIdxList);
CrdR=sortrows([x2,y2],2);%sort rows
CrdR(:,1)=CrdR(:,1)+CnrR(1);
CrdR(:,2)=CrdR(:,2)+CnrR(2);
%plot(CrdR(:,1),CrdR(:,2),'r')

if isempty(y2)==1
    fprintf('Edge detection has failed (right) - no edge detected. Continuing to
        next frame.')
end

%.. For Thresh GUI
    if SvFlag==1
```

```
        save(fullfile(Path,'ThreshTemp2'));
    end
%------------------------------------------------------------------
%% Generate circle mask
%Bubbles are circular and dark
function [BubMask]=CircMask(Grey,Thr,Aug)
%[centers, radii] = imfindcircles(Grey,[10 100],'ObjectPolarity','dark');
%Thr=0.1; Aug=2;
[centers, radii] = imfindcircles(Grey,[10 30],'ObjectPolarity','dark','
    EdgeThreshold',Thr);
[centers2, radii2] = imfindcircles(Grey,[30 90],'ObjectPolarity','dark','
    EdgeThreshold',Thr);
[centers3, radii3] = imfindcircles(Grey,[90 150],'ObjectPolarity','dark','
    EdgeThreshold',Thr);

centers=[centers;centers2;centers3];
radii=[radii;radii2;radii3];

BubMask=zeros(size(Grey));

if isempty(radii)==0
    T=table(centers,radii);
    %imshow(Grey);
    %h = viscircles(centers,radii);

    for i=1:size(T,1)
        %(x-x0)2+(y-y0)2=R2 -> y=sqrt(R2 -(x-x0)2)+-y0
        xMin=int32(max(T.centers(i,1)-Aug*T.radii(i),1));
        xMax=int32(min(T.centers(i,1)+Aug*T.radii(i),size(Grey,2)));
        for j=xMin+1:xMax-1
            a=sqrt((Aug*T.radii(i))^2-(double(j)-T.centers(i,1))^2);
            yMin=int32(max(T.centers(i,2)-a,1));
            yMax=int32(min(T.centers(i,2)+a,size(Grey,1)));
            BubMask(yMin:yMax,j)=1;
        end%for x range of circle
    end
end

%% White Mask
function [W3]=WhiteMask(Th,Grey,n)
%Create a "white" mask
%...Th is the adjusted BW threshold (Thresh*BWadj)
W1=~im2bw(Grey,Th);
imshow(W1);

W2 = ~bwmorph(W1,'close');
```

```
%.. Identify the largest object
CC=bwconncomp(W2);
L=labelmatrix(CC);%Create label matrix
imshow(label2rgb(L));

 Stats=regionprops(L,'Area');
 [~,iMax] = max([Stats.Area]);
W2(L==iMax)=0;
W3=W2;
 for i=1:n
     W3 = bwmorph(W3,'dilate');
end
%------------------------------------------------------------------------------------%

%% TRI (%%%Call function handle from main GUI)
%Called from ImMask
 function [CrdMat]=Tri(Grey,Mask,ThreshC,FlagOptns,NL)
  %FlagOptns={SBD,WhiteMask(WM),BubMask}

%------------------------------------------------------------------------------------%
% PICK EDGES using Canny edge detection method
%------------------------------------------------------------------------------------%
  %--Parameters
     N=2;%number of lines (units/components) to process
     C=10;%X distance to clear (in Pixels)

  %--Pick edges
     Edges=edge(Grey,'canny',ThreshC);

     %Subtract mask
     Edges(Mask==1)=0; %Replaces subtraction

  %--Determine longest line
   if max(max(Mask))==0 %max mask value is zero -> inactive -> LONGEST LINE SEARCH
     imsk = bwmorph(Edges,'thin',Inf);%thinning edges to single line thicknss
     bpoints = bwmorph(imsk,'branchpoints',1);%searching for points at intercies of
         lines
     imsk(bpoints)=0;%remove points at intersections - all lines now distinct

     %Pull up area properties, sort
     lineStats = regionprops(imsk, {'Area','PixelList'});
     [Stats,indi]=sortrows(struct2table(lineStats),1,'descend');

     %Label matrix
     CCl = bwconncomp(imsk);
     Ll = labelmatrix(CCl);
```

```
    %Make new edge matrix from the NL largest lines
    %default(NL)=2;%NL is also saved in TrialThO
    CrdMat=zeros(size(Edges));
    for i=1:NL
        CrdMat(Ll==indi(i))=1;
    end

else
    CC=bwconncomp(Edges);
    Ll = labelmatrix(CC);
    Stats = regionprops(CC,'area','PixelIdxList');
    idx = find([Stats.Area] > 20); %to update with GUI
    %imshow(ismember(labelmatrix(CC), idx));
    CrdMat=zeros(size(Edges));
    IND=[];
    for i=idx
        IND=[IND;Stats(i).PixelIdxList];%linear indices %Produces a cell array
        CrdMat(Ll==i)=1;
    end
    %New Canny edge (no mask) for SBE
    Edges=edge(Grey,'canny',ThreshC);
    imsk = bwmorph(Edges,'thin',Inf);%thinning edges to single line thicknss
    bpoints = bwmorph(imsk,'branchpoints',1);%searching for points at intercies of lines
    imsk(bpoints)=0;%remove points at intersections - all lines now distinct
    CCl = bwconncomp(imsk);
    Ll = labelmatrix(CCl);
end

%FlagOptns={SBD,WhiteMask(WM),BubMask}
if FlagOptns(1) == 1 %(right)(SBE)
    %...Clear components touching the edge
        %...Canny seems to clear the final row of edge pixels. Isolate top and right facing pixels.
        Edge2=imclearborder(imsk(:,1:end-1));
        Edge3=zeros(size(imsk));
        Edge3(:,1:end-1)=Edge2;

    %...Subtract to Search for lines connected to the right-most edge
        Edge4=imsk;
        Edge4(Edge3==1)=0;
        %imshow(Edge4+0.5*Edges-0.5);

    %...Region properties - using major axis for length.
        CC = bwconncomp(Edge4);
        if CC.NumObjects>0 %objects exist
```

```
            L = labelmatrix(CC);
            Stats=regionprops(L,'PixelList','area','majorAxisLength');
            for i=1:length(Stats);
                Stats(i).Label=i;
            end%add column with label
            t=struct2table(Stats,'AsArray', true);
        %...Identify the largest major axis lengths.
            t2=sortrows(t,'MajorAxisLength','descend');
            if size(t2,1)<N
               Labels=t2.Label(:);
            else
               Labels=t2.Label(1:N);
            end
        %...Working on the longest lengths, clear the left-most (inside) 5 x width
            NewMask=zeros(size(Edge3));%
            for l=Labels
                SegCrd=table2array(t.PixelList(l));
                %SegCrd will be sorted by x-value, ascending.
                for i=1:length(SegCrd)
                    if SegCrd(i,1)>SegCrd(1,1)+C,
                        NewMask(SegCrd(i,2),SegCrd(i,1))=1;
                    end
                end%i=1:length(SegCrd)
            end%for l=Labels

        %...Search again.
            CC = bwconncomp(NewMask);
            L = labelmatrix(CC);
            Stats=regionprops(L,'PixelIdxList','PixelList','Area','MajorAxisLength
                ','Orientation');
            for i=1:length(Stats);
                Stats(i).Label=i;
            end%add column with label
            t=struct2table(Stats,'AsArray', true);
            t2=sortrows(t,'MajorAxisLength','descend');

        %...Entry with largest major axis length, with POSITIVE orientation (
            angles LL to TR **RIGHT SIDE)
            i=0;
            while i<length(Stats)-1 && t2.Orientation(i+1)<=0
                i=i+1;
            end%while

            if i~=length(Stats)%segments matching the critera were found.
             %--Remove "longest line" coords overlapping the new x coordinates.
                %--x limit
```

```
                CrdB=table2array(t2.PixelList(i+1,:));
                yr = CrdB(1,2);
                %--Add longest line
%                 CrdMat=zeros(size(NewMask));
%                 CrdMat(Ll==index)=1;
                %--Clear SBD area and add SBD
                CrdMat(1:yr,:)=0;
                CrdMat(L==t2.Label(i+1))=1;%t2.Label(i+1) --> component label
                Stats = regionprops(CrdMat, 'PixelIdxList');
                IND=Stats(1).PixelIdxList;%linear indices
            end%if
        end%(if numobj==1)

    elseif FlagOptns ==2 %(left)
    %...Clear components touching the edge
    %...Canny seems to clear the final row of edge pixels. Isolate top and right facing pixels.
        %EdgeL=edge(Grey,'canny',ThreshC);
        Edge2=imclearborder(imsk(:,2:end));
        Edge3=zeros(size(imsk));
        Edge3(:,2:end)=Edge2;

    %...Subtract to Search for lines connected to the right-most edge
        Edge4=imsk;
        Edge4(Edge3==1)=0;
        %imshow(Edge4+0.5*Edges-0.5);

    %...Region properties - using major axis for length.
        CC = bwconncomp(Edge4);
        if CC.NumObjects>0 %objects exist
            L = labelmatrix(CC);
            Stats=regionprops(L,'PixelList','area','majorAxisLength');
            for i=1:length(Stats);
                Stats(i).Label=i;
            end%add column with label
            t=struct2table(Stats,'AsArray', true);
        %...Identify the largest major axis lengths.
            t2=sortrows(t,'MajorAxisLength','descend');
            if size(t2,1)<N
               Labels=t2.Label(:);
            else
               Labels=t2.Label(1:N);
            end
            %...Working on the longest lengths, clear the left-most (inside) 5 x width
            NewMask=zeros(size(Edge2));%smaller
```

```
            for l=Labels
                SegCrd=flipud(table2array(t.PixelList(l)));%innermost x = x1
                for i=1:length(SegCrd)
                    if SegCrd(i,1)<SegCrd(1,1)-C,
                        NewMask(SegCrd(i,2),SegCrd(i,1))=1;
                    end
                end%i=1:length(SegCrd)
            end%for l=Labels

        %...Search again.
            CC = bwconncomp(NewMask);
            L = labelmatrix(CC);
            Stats=regionprops(L,'PixelList','Area','MajorAxisLength','Orientation'
                );
            for i=1:length(Stats);
                Stats(i).Label=i;
            end%add column with label
            t=struct2table(Stats,'AsArray', true);
            t2=sortrows(t,'MajorAxisLength','descend');

        %...Entry with largest major axis length, with NEGAIVE orientation (angles
             LR to TL **LEFT SIDE)
            i=0;
            while i<length(Stats)-1 && t2.Orientation(i+1)>=0
                i=i+1;
            end%while
            %t2.MajorAxisLength(1);
            if i~=length(Stats)%segments matching the critera were found.
                %--Remove "longest line" coords overlapping the new x coordinates.
                %--x limit
                CrdB=table2array(t2.PixelList(i+1,:));
                yl = CrdB(end,2);
                %--Add longest line
%                 CrdMat=zeros(size(NewMask));
%                 CrdMat(Ll==index)=1;
                %--Clear SBD area (above) and add SBD
                CrdMat(1:yl,:)=0;
                CrdMat(L==t2.Label(i+1))=1;%t2.Label(i+1) --> component label
                Stats = regionprops(CrdMat, 'PixelIdxList');
                IND=Stats(1).PixelIdxList;%linear indices
            end%if
        end%if numobj==1
end%(if SBD=)
%------------------------------------------------------------------------


```

```
function ET_Notes_Callback(hObject, eventdata, handles)
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------

% --- Executes during object creation, after setting all properties.
function ET_Notes_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in CB_Print.
function CB_Print_Callback(hObject, eventdata, handles)
% Hint: get(hObject,'Value') returns toggle state of CB_Print


% --- Executes on selection change in PU_Data.
function PU_Data_Callback(hObject, eventdata, handles)
%-------------------------------------------------------------------------
%-- Enable ET_BestNum if call back "best" is used.
%-------------------------------------------------------------------------


% --- Executes during object creation, after setting all properties.
function PU_Data_CreateFcn(hObject, eventdata, handles)
%-------------------------------------------------------------------------
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%-------------------------------------------------------------------------


% --- Executes on selection change in PU_Side.
function PU_Side_Callback(hObject, eventdata, handles)
% hObject    handle to PU_Side (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns PU_Side contents as cell array
%        contents{get(hObject,'Value')} returns selected item from PU_Side


% --- Executes during object creation, after setting all properties.
function PU_Side_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to PU_Side (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in PB_LoadResults.
function PB_LoadResults_Callback(hObject, eventdata, handles)
%----------------------------------------------------------------
%-- ALLOW USER TO CALL A SPECIFIC RESULTS FILE
    [ResultFile,Path]=uigetfile('.mat','multiselect','off')
    FullFileName=fullfile(Path,ResultFile);
    load(FullFileName);

%-- Plot
    axes(handles.axes2), hold on
    plot(Results(:,1),Results(:,21),'+r','markersize',3)%plot gamma (aVE)
    xlabel('time (min)'); ylabel('interfacial tension (L,R BEST), mN/mm, Temp(C)')
        ;
%----------------------------------------------------------------



function ET_BestNum_Callback(hObject, eventdata, handles)
%----------------------------------------------------------------

%----------------------------------------------------------------


% --- Executes during object creation, after setting all properties.
function ET_BestNum_CreateFcn(hObject, eventdata, handles)
%----------------------------------------------------------------
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%----------------------------------------------------------------


% --- Executes on button press in CB_MWtog.
```

```
function CB_MWtog_Callback(hObject, eventdata, handles)
%--------------------------------------------------------------
load(fullfile(handles.Path,'MData.mat')); %microwave data
%-- PLOT MICROWAVE RUNNING TIME
MS=MicrStart/60;%frame when microwave is turned on
MO=MicrStart/60+irTime;%frame when microwave is turned off
plot([MS,MS],[0,100],'y',[MO,MO],[0,100],'y');
%--------------------------------------------------------------


% --- Executes on button press in CB_Temp.
function CB_Temp_Callback(hObject, eventdata, handles)
%--------------------------------------------------------------
Tnum=get(handles.DD_Temp,'Value');
load(fullfile(handles.Path,sprintf('TC%i.mat',Tnum)));
TC2=[TC,transpose(1:length(TC))];
TC2(:,2)=TC2(:,2)/60;

axes(handles.axes2)
plot(TC2(:,2),TC2(:,1),'g');
%--------------------------------------------------------------


% --- Executes on button press in PB_SaveFig.
function PB_SaveFig_Callback(hObject, eventdata, handles)
%--------------------------------------------------------------
%-- SAVE GUI FIGURE
%--------------------------------------------------------------
folder=getappdata(0,'folder');
saveas(gcf,fullfile(folder,'ResultsMainGUI'),'fig');
fprintf('figure saved');
%--------------------------------------------------------------



% --- Executes on button press in PB_Cnr.
function PB_Cnr_Callback(hObject, eventdata, handles)
%--------------------------------------------------------------
    if isfield(handles,'ImName')==1
        DropMovie = VideoReader(handles.ImName);
        Grey = read(DropMovie, handles.ST);
        if get(handles.CB_Flip,'Value')==1
            Grey=flipud(Grey);
            set(handles.CB_Flip,'value',1);
        end
        hold off, imshow(Grey);
```

```
        button=questdlg('Crop images?','Crop?','Yes','No','No');
        if strcmp(button,'Yes')==1
            fprintf(1,'Select area to crop\n')
            rec=round(getrect());
            Crop=[rec(1),rec(2),rec(1)+rec(3),rec(2)+rec(4)];
            if Crop(3)>size(Grey,2);Crop(3)=size(Grey,2);end
            if Crop(4)>size(Grey,1);Crop(4)=size(Grey,1);end
        else
            Crop=handles.Crop;
        end
        Cnr=Crop;
        Grey=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));
        %update
        handles.Crop=Crop;
        handles.Grey=Grey;
    else
%           ImName=sprintf('%s%04i.%s',handles.Prefix,handles.ST,handles.Ext);
%           Grey = imread(ImName);
        Grey=handles.Grey; %already cropped)
    end

%--- REDEFINE FITTING AREAS
    axes(handles.axes1); hold off
    imshow(Grey); hold on

    fprintf(1,'Selecting an area outside of image boundaries will return an error\
        n')
    fprintf(1,'Please select the area to fit for the holm (right)\n(click and drag
         box):\n')
    rec=round(getrect());
    HlmCnrR=[rec(1),rec(2),rec(1)+rec(3),rec(2)+rec(4)];
    if HlmCnrR(3)>size(Grey,2);HlmCnrR(3)=size(Grey,2);end
    if HlmCnrR(4)>size(Grey,1);HlmCnrR(4)=size(Grey,1);end

    fprintf(1,'Please select the area to fit for the holm (left)\n(click and drag
        box):\n')
    rec=round(getrect());
    HlmCnrL=[rec(1),rec(2),rec(1)+rec(3),rec(2)+rec(4)];
    if HlmCnrL(3)>size(Grey,2);HlmCnrL(3)=size(Grey,2);end
    if HlmCnrL(4)>size(Grey,1);HlmCnrL(4)=size(Grey,1);end

    HlmCnr=[HlmCnrR;HlmCnrL];

    fprintf(1,'Please select the area to fit for the sphere (right) \n(click and
        drag box):\n')
```

```
    rec=round(getrect());
    SphCnr(1,:)=round([rec(1),rec(2),rec(1)+rec(3),rec(2)+rec(4)]);

    fprintf(1,'Please select the area to fit for the sphere (left) \n(click and
        drag box):\n')
    rec=round(getrect());
    SphCnr(2,:)=[rec(1),rec(2),rec(1)+rec(3),rec(2)+rec(4)];

    %Display boxes
    plot([HlmCnrR(1),HlmCnrR(3),HlmCnrR(3),HlmCnrR(1),HlmCnrR(1)],[HlmCnrR(2),
        HlmCnrR(2),HlmCnrR(4),HlmCnrR(4),HlmCnrR(2)],'r');
    plot([HlmCnrL(1),HlmCnrL(3),HlmCnrL(3),HlmCnrL(1),HlmCnrL(1)],[HlmCnrL(2),
        HlmCnrL(2),HlmCnrL(4),HlmCnrL(4),HlmCnrL(2)],'r');
    plot([SphCnr(1,1),SphCnr(1,3),SphCnr(1,3),SphCnr(1,1),SphCnr(1,1)],[SphCnr
        (1,2),SphCnr(1,2),SphCnr(1,4),SphCnr(1,4),SphCnr(1,2)],'r');
    plot([SphCnr(2,1),SphCnr(2,3),SphCnr(2,3),SphCnr(2,1),SphCnr(2,1)],[SphCnr
        (2,2),SphCnr(2,2),SphCnr(2,4),SphCnr(2,4),SphCnr(2,2)],'r');

    %Save
    handles.HlmCnr=HlmCnr;
    handles.SphCnr=SphCnr;
    guidata(hObject, handles);
%-----------------------------------------------------------------------

% --- Executes during object creation, after setting all properties.
function axes2_CreateFcn(hObject, eventdata, handles)
%-----------------------------------------------------------------------
cla(hObject);
xlabel('time (min)'); ylabel('interfacial tension (L,R BEST), mN/mm, Temp(C)');
%-----------------------------------------------------------------------


% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
%-----------------------------------------------------------------------
%cla(hObject,'reset'); %Clear axis
%-----------------------------------------------------------------------


% --- Executes on button press in CB_Cnt.
function CB_Cnt_Callback(hObject, eventdata, handles)
% hObject    handle to CB_Cnt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of CB_Cnt
```

```
function ET_Prefix_Callback(hObject, eventdata, handles)
% hObject    handle to ET_Prefix (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_Prefix as text
%        str2double(get(hObject,'String')) returns contents of ET_Prefix as a
    double


% --- Executes during object creation, after setting all properties.
function ET_Prefix_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_Prefix (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_Ext_Callback(hObject, eventdata, handles)
% hObject    handle to ET_Ext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_Ext as text
%        str2double(get(hObject,'String')) returns contents of ET_Ext as a double


% --- Executes during object creation, after setting all properties.
function ET_Ext_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_Ext (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in CB_Check.
function CB_Check_Callback(hObject, eventdata, handles)
% hObject    handle to CB_Check (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of CB_Check



function ET_Ref_Callback(hObject, eventdata, handles)
handles.Ref=str2double(get(hObject,'String'));

% --- Executes during object creation, after setting all properties.
function ET_Ref_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_Ref (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in radiobutton4.
function radiobutton4_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton4


% --- Executes on button press in radiobutton1.
function radiobutton1_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton1


% --- Executes on button press in PB_Modify.
function PB_Modify_Callback(hObject, eventdata, handles)
% hObject    handle to PB_Modify (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in CB_Flip.
function CB_Flip_Callback(hObject, eventdata, handles)
% hObject    handle to CB_Flip (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of CB_Flip


% --- Executes on button press in PB_ExportData.
function PB_ExportData_Callback(hObject, eventdata, handles)
%------------------------------------------------------------------
% EXPORT CURRENT DATA TO SPREADSHEET
%------------------------------------------------------------------
% Export requested
disp('Opening GUI')
% Open dialigue box, confirm settings - incl sheet name
    % Suggest values for existing files
    FileInfo=SaveOutputGUI;
    %Template = 'C:\Users\14291160\Dropbox\PhD-ChemEng\Updated code - Image
        Analysis (holm)\NewGUI\ResultsTemp6.xlsm';
    Template = 'C:\Users\AHyde\Dropbox\PhD-ChemEng\Updated code - Image Analysis (
        holm)\NewGUI\ResultsTemp6.xlsm';

% Comfirm whether file exists:
if exist(FileInfo.XLfileName, 'file') == 2
   disp('File exists')
    %EXISTS: Check if file is open
else
    %DOES NOT EXIST: Copy template to new file name
    copyfile(Template, FileInfo.XLfileName);
    disp('File created')
end
disp('Matlab is writing to the specified file')
```

```
% Search for saved results file (.mat)
    %Headfile
    load(fullfile(FileInfo.FolderPath,'Head.mat'));

% Export data:
%     HEADER
    A = {'Run identifier:','','',folder,'','','Movie name:','',FileName,'','','','
        ','','Adj:',Thta;};
    xlswrite(FileInfo.XLfileName,A,FileInfo.WorkSheet,'A3')
%     DATA
    load(FileInfo.MATfileName);%data file

    %--variable is an original file, with results matrix variable name 'Results'
    if exist('Results')==1
        xlswrite(FileInfo.XLfileName,Results,FileInfo.WorkSheet,'A11')
        xlswrite(FileInfo.XLfileName,FPS,FileInfo.WorkSheet,'AK8')%fps
        xlswrite(FileInfo.XLfileName,Data.Size,FileInfo.WorkSheet,'AJ10')%Sphere

    %--variable is a 'scaled' file, with results matrix variable name 'ResultsNew'
    elseif exist('ResultsNew')==1
        xlswrite(FileInfo.XLfileName,ResultsNew,FileInfo.WorkSheet,'A11')
        xlswrite(FileInfo.XLfileName,FPS,FileInfo.WorkSheet,'AK8')%fps
        xlswrite(FileInfo.XLfileName,Data.Size,FileInfo.WorkSheet,'AJ10')%Sphere

    else
        h=msgbox({'There was an error writing the file.' 'It looks like the
            variable does not exist.' 'This action will terminate.'})
        uiwait(h)
    end
    %xlswrite(FileInfo.XLfileName,Results,FileInfo.WorkSheet,'AO8')%MW on
    %xlswrite(FileInfo.XLfileName,Results,FileInfo.WorkSheet,'AP8')%MW off

    %ex=winopen(FileInfo.XLfileName); %open excel file

disp('Done')
%----------------------------------------------------------------


% --- Executes on button press in PB_Rescale.
function PB_Rescale_Callback(hObject, eventdata, handles)
%----------------------------------------------------------------
%--RESCALE SHAPE FACTOR
%----------------------------------------------------------------
% Call GUI
disp('Opening GUI...')
ReScaleGamma
```

```
disp('Rescale complete')
%-----------------------------------------------------------------
```

B.2 Holm_Main (back-end analysis)

Listing B.2: Holm_MAIN_3.m

```
%------------------------------------------------------------------------------%
%DETERMINATION OF FLUID INTERFACE PROPERTIES THROUGH IMAGE ANALYSIS
%------------------------------------------------------------------------------%
%% Version-up info
%  New edge detecton module to handle broken edges and reflection.

%% INFORMATION
%Matlab code to calculate fluid-fluid surface tension Submerged holm
%Can be used repetitively with movies (use MovieHolmX to read frames in).

%CALLS EXTERNAL FUNCTIONS:
%
%------------------------------------------------------------------------------%
%Written for PhD(ChemEng), 2014-2018
%Anita Hyde, 14291160
%Supervisor: Chi Phan
%Written in MATLAB R2012.  Requires Image Analysis Toolbox and Curve Fitting
    Toolbox.
%Migrated to Matlab 2015b in 2016 (major change to graphics system introduced in
    2014b).

%% START
function [Gamma,aBF,fval,OptStore]=Holm_MAIN_3(Side,ImCase,ImName,PrintYN,x0,y0,R,
    HolmCoord,Grey,folder,fig,FDisp,Data)
    %% DECLARATIONS
    %ImCase= num %IDENTIFY DATABASE ENTRY
    %askBox = use getrect function to unput holm and sphere location
    %Xu, Yu: points to estimate edge location
    %Grey - Greyscale image.  For movies.
    %SL=5; %number of points to search
    %N=5; %number of reruns at each starting point (different set of random
    %coordinates)

    %------------------------------------------------------------------------------
    %READ IMAGE DATA
    %------------------------------------------------------------------------------
    SL=Data.SL;
    N=Data.N;
    nran=Data.nRan;
    Tol=Data.Tol;
    g=9.81;
    GamEst=Data.GamEst;
```

```
    Size=Data.Size;
    dRho=Data.dRho;
    %ImName=Data.ImName;
    SphCnr=Data.SphCnr;
    HlmCnr=Data.HlmCnr;

    %------------------------------------------------------------------
    %SAVE DATA - FILES - PrintYN=Yes
    %------------------------------------------------------------------
    if PrintYN==true
        %files
        Result_file=sprintf('%s-O-%s-%i.txt',folder,Side,ImCase);
        %fullFileNameA = fullfile(folder, Result_file);
        fileID=fopen(Result_file,'a');%use a+ for reading & writing, append
        %--abbrv file header
        fprintf(fileID,'%s: results for file, %s\r\n',datestr(now),ImName);
        %===============CLEANUP================
        finishup = onCleanup(@() myCleanupFun(fileID));
        %======================================
        %fprintf(fileID,'[    a    |  hlim   | theta(rad)| fvalh | Coord num]\r\n', datestr(now),ImName);
    else
        fileID=1;%print to screen
    end
    %------------------------------------------------------------------
    %ANALYSIS
    %------------------------------------------------------------------


%FITTING (2 pass)
axes(fig),
%--Estimate shape factor
a0=sqrt(dRho*g/(GamEst/1000));%gamma in mN/m -> a (m).  See {Huh, Chun and Scriven, L.E. 1969}
Scale=Size/2/R;%image scale mm/p. 'Size'=sphere diameter in mm
a0=a0/1000*Scale;%Scale = mm/pixeRl
fprintf(1,'starting "a": %f\r\n',a0);
%--fit polynomial to initial segment to calculte tangent
[poly2,MonSrt,PolyCoord]=FitPoly(x0,y0,R,Tol,int32(70),SL,HolmCoord);%(x0,y0,R,Tol,PL,HolmCoord). TOl is sum(diff)

if PrintYN==1
    %print edge information
    fprintf(fileID,'Search area: sphere x1,y1,x4,y4; coordinates x/y \r\n');
    fprintf(fileID,'% 5i',SphCnr);
    fprintf(fileID,'\r\n Sphere profile: x0: %f, y0: %f, R: %f \r\n',x0,y0,R);
```

```
    fprintf(fileID,'\r\n \r\n');
    fprintf(fileID,'Search area: holm x1,y1,x4,y4; coordinates x,y \r\n');
    fprintf(fileID,'\r\n');
    fprintf(fileID,'% 5i',HolmCoord(1,:));
    fprintf(fileID,'\r\n');
    fprintf(fileID,'% 5i',HolmCoord(2,:));
    fprintf(fileID,'\r\n \r\n Polynomial (Poly2): % 10f, % 10f, % 10f \r\n',poly2);
    fprintf(fileID,'\r\n \r\n Additional information: Holm starting coordinate:% 3i; SL:% 3i; N:% 3i; number of points (udist): %i\r\n \r\n',MonSrt,SL,N,nran);

    %labels
    fprintf(fileID,'%s\r\n','First past - searching for best initial point');
    fprintf(fileID,'%12s','a0','a','hlim','theta','fvalh','coord','fitting index');
end

%--Initialise matrices for storage
HOsize=1:SL;%Specify indices (in HolmCoord) of starting points to consider.

FitCoordX=zeros(length(HOsize),length(HolmCoord)+1);
FitCoordY=zeros(length(HOsize),length(HolmCoord)+1);
ODEX=zeros(length(HOsize),length(0:0.001:3*pi));
ODEY=zeros(length(HOsize),length(0:0.001:3*pi));
OptStore=zeros(length(HOsize),7);%Stores the optimium values found at each iteration.

hold all;
%OPTIMISATION
fprintf('Evaluating...');
HOsize=zeros(2,SL*N);
SL=double(SL);N=double(N);MonSrt=double(MonSrt);
for i=0:SL
    j=N*i+1;
    HOsize(1,j:j+N-1)=(i+1)*ones(1,N);
end
for i=1:3:SL*N
    HOsize(2,i:i+2)=[a0/2,a0,a0*2];
end

% fprintf(fileID,'\r\n Best starting points found: %i, %i \r\n', S)
% fprintf(fileID,'Fitting %i times using %i random points \r\n',N*p,nran)

for k=1:length(HOsize)
    CoordNum=HOsize(1,k);
```

```
    a=HOsize(2,k);
    fprintf('%d...',CoordNum)

    %--Select points for fitting (nran + starting point)
    pts=random('unid',length(HolmCoord)-MonSrt-SL,1,nran);%does not pick points from initial SL
    pts=sort(pts,2)+MonSrt+SL;%50 random points > starting point
%   pts=[CoordNum,pts];%starting point + 50 random points as index
    Coord=[PolyCoord(:,CoordNum:end),HolmCoord(:,pts)];
    %plot(Coord(1,:),Coord(2,:),'o','markersize',4);
    fvalS=1;%%TEMP
    %--FIT HOLM CURVE (3 VAR OPT)
    [HolmOpt,fvalH,Xxi,Yyi,X,Y]=holm(x0,y0,R,a,Coord,poly2);
    FitCoordX(k,1:length(Xxi)+1)=[length(Xxi),Xxi];
    FitCoordY(k,1:length(Yyi)+1)=[length(Yyi),Yyi];
    ODEX(k,1:length(X)+1)=[length(X);X];
    ODEY(k,1:length(Y)+1)=[length(Y);Y];
    OptStore(k,:)=[a,HolmOpt,fvalH,fvalS,CoordNum];%[a0,(a,hlim,theta(rad),fvalh),Coord num]

    %--print output to file
    if PrintYN==1
        fprintf(fileID,'\r\n');
        fprintf(fileID,'%4i, % 12.4f, % 12.4f, % 12.4f, % 12.4f, % 12.4f % 12i',k,HOsize(2,k),OptStore(k,:));
        fprintf(fileID,'% 5i','',pts);
        fprintf(fileID,'% 12.4f','',X,'',Y);
    end
    %plot(X,Y)
end%REPEAT at sucessive coordinates from start
    %Identify the minimum and mean error
    MinError=min(OptStore(:,5));
    %Keep entries less than X times the minimum error.
    S=OptStore(:,5)<2*MinError;%logical matrix of rows which 'pass'
    Store=zeros(sum(S),8);
    Store(:,1:7)=OptStore(S==1,:);%Rows which pass
    Store(:,9)=dRho*g*1000./(Store(:,2)./Scale*1000).^2;
    ODEX=ODEX(S==1,:); ODEY=ODEY(S==1,:);%limit ODEX/Y to match Store.

    [~,Best]=min(Store(:,5));
    if strcmp(Side,'Left')==1
         for i=1:sum(S)
              if PrintYN==1
                  fprintf(fileID,'% 5i, % 12.6f, % 12.6f, % 12.4f, % 12.4f, % 12.4f, % 12i, % 12.4f, % 12.2f\r\n',i,Store(i,:));
              end
```

```
                    st=ODEX(i,1);
                    Xrev=size(Grey,2)-ODEX(i,2:st);
                    plot(Xrev,ODEY(i,2:st),'-','color',[0.5,0.5,0.5]);
                    %figure(2), plot(ODEX(i,2:st),ODEY(i,2:st),'-y')%,'color
                        ',[0.5,0.5,0.5]);
           end
     else %right
          for i=1:sum(S)
                    if PrintYN==1
                         fprintf(fileID,'% 5i, % 12.4f, % 12.4f, % 12.4f, % 12.4f, %
                              12.4f, % 12i, % 12.4f, % 12.2f\r\n',i,Store(i,:));
                    end
                    st=ODEX(i,1);
                    plot(ODEX(i,2:st),ODEY(i,2:st),'-','color',[0.5,0.5,0.5]);
           end
     end

%SCALING AND CALCULATIONS
fHave=mean(Store(:,5));%filtered, average fitting error
aBest=Store(Best,2);
fprintf('Best "a": %f',aBest);
GammaAveF=mean(Store(:,9));
Gamma=Store(Best,9);
fvalH=OptStore(Best,5);fvalS=OptStore(Best,6);
fval=[fvalH,fvalS,fHave];
aBF=[aBest,mean(Store(:,2))];
Gamma=[Gamma,GammaAveF];
stDev=std(Store(:,9));

%Show best result
    st=ODEX(Best,1);
    axes(fig), hold on
    if strcmp(Side,'Left')==1
         Xrev=size(Grey,2)-ODEX(Best,2:st);
         plot(Xrev,ODEY(Best,2:st),'-','color',[0.8,0.3,0.1],'linewidth',1.5);
         figure(FDisp), plot(Xrev,ODEY(Best,2:st),'-','color',[0.8,0.3,0.1],'
              linewidth',1.5);
         figname=sprintf('Edge-%d',ImCase);
         fullFileName = fullfile(folder, figname);
         saveas(FDisp, [fullFileName '.png']) %2nd side
         plot([1,1200],[Store(Best,3),Store(Best,3)],'y');
    else
         plot(ODEX(Best,2:st),ODEY(Best,2:st),'-','color',[0.8,0.3,0.1],'linewidth'
              ,1.5);
         figure(FDisp), plot(ODEX(Best,2:st),ODEY(Best,2:st),'-','color'
              ,[0.8,0.3,0.1],'linewidth',1.5);
```

```
        plot([1,1200],[Store(Best,3),Store(Best,3)],':y');
    end

    %====================================
    %Results
    %====================================
    Res=[Gamma(1);GammaAveF;MinError;stDev;sum(S)];
    Res=array2table(Res,'VariableNames',{Side},'RowNames',{'IFT-Best','IFT-Ave','MinError','stDev','n'});
    disp(Res)

end %end main function


%% FIT POLY
function [poly2,MonSrt,PolyCoord]=FitPoly(x0,y0,R,Tol,PL,SL,Coord)
%------------------------------------------------------------
%FN FITPOLY: fits polynomial over short range at start (bubble end) of
%curve.  Returns coefficients as matrix (polyfit format) to allow for
%tangent calculation.  ALso return MonSrt - 1st coordinate.
%------------------------------------------------------------
 %STARTING COORDINATES - based on deviation from sphere
%..Max sphere height (pixel position):
    Ym=y0-R;
    i=length(Coord);
    while Coord(2,i)<Ym
        i=i-1;
    end
    Ym=[Ym,i];
%..Half Sphere height
    j=i;
    while Coord(2,j)<y0 && j>2
        j=j-1;
    end

%     %re-order lower segment by x, not y.
%     CrdLow=sortrows(transpose(Coord(:,j:i)),2);
%     Coord(:,j:i)=transpose(flipud(CrdLow));

%..Search backwards
    Diff=10;
    while Diff>1&&i>2
        i=i-1;
        X=Coord(1,i);
        Y=Coord(2,i);
        x=sqrt(abs(R^2-(Y-y0)^2))+x0;%(x-X)^2+(y-Y)^2=R^2
```

```
            Diff=X−x;%not abs − do not want to start inside the circle
        end
        %plot(X,Y,'yo')
%..Search forwards
        Diff=0;
        while Diff<Tol
            i=i+1;
            X=Coord(1,i);
            Y=Coord(2,i);
            x=sqrt(abs(R^2−(Y−y0)^2))+x0;%(x−X)^2+(y−Y)^2=R^2
            diff=X−x;%not abs − do not want to start inside the circle
            if diff>1%Will not pick up negative values.
                Diff=Diff+diff;
            end
            %plot([X,x],[Y,Y],'y')
            PolStart=int32(i);
        end

        %plot(X,Y,'mo')
        MonSrt=PolStart;
        if PolStart<=2
            display(sprintf('Unable to determine starting point for polynomial.
                Default is first coordinate.  Please check interface between sphere
                and holm'));
        elseif PolStart<6
            PolStart=1;
        else
            PolStart=PolStart−5;
        end

        %temp for sharp angles
        PolStart=MonSrt;
        MonSrt=MonSrt+2;

    %FIT ploy/power curve to edge section.  Use X=fn(Y, quadratic)
        poly2=polyfit(Coord(2,PolStart:MonSrt+PL),Coord(1,PolStart:MonSrt+PL),2);
        PolyCoord=[polyval(poly2,Coord(2,MonSrt:MonSrt+SL));Coord(2,MonSrt:MonSrt+SL)
            ];
    %PLOT (plots on right side)
        plot(polyval(poly2,Coord(2,PolStart:MonSrt+PL)),Coord(2,PolStart:MonSrt+PL),'b
            ')
        plot(Coord(1,MonSrt),Coord(2,MonSrt),'go',Coord(1,MonSrt+PL),Coord(2,MonSrt+PL
            ),'go')
        %plot(Coord(1,:),Coord(2,:),'r')

end
```

```
%-----------------------------------------------------------------%

%% HOLM
function [HolmOpt,fvalH,Xxi,Yyi,X,Y]=holm(x0,y0,R,a,Coord,poly2)
%-----------------------------------------------------------------%
%HOLM - outer function, calls solver
%-----------------------------------------------------------------%
 %COORD has intital coordinate + 50 random points for fitting
 %--estimate x-axis position
     Xxi=[];Yyi=[];X=[];Y=[];
     HLim=Coord(2,end);%--y=0 (lim(holm edge))-->average final 10 points
 %--Scaling
     ScaleR=[1,10,1];

 %--Initial conditions
     [Initial]=initialCoord(Coord,poly2);%
     Guess=[a/ScaleR(1),0.2*HLim/ScaleR(2),Initial(3)/ScaleR(3)];%a, x axis position. Scaled.

 %--Boundaries for fmincon
     LowerBound=[0.01*a/ScaleR(1),-100*HLim/ScaleR(2),0.9*Initial(3)/ScaleR(3)];
     UpperBound=[5*a/ScaleR(1),5*HLim/ScaleR(2),1.1*Initial(3)/ScaleR(3)];%NOTE: Y=0 at top of image

 %OPTIMISATION
     Opts = optimset('Display','none','Algorithm','active-set','LargeScale','on','MaxFunEval',1000);
     [HolmOpt,fvalH,~,output]= fmincon(@(Guess)HolmObj(Guess,x0,Coord,ScaleR,Initial),Guess,[],[],[],[],LowerBound,UpperBound,[],Opts);
        %% OBJECTIVE FUNCTION (HOLMOBJ - nested)
        function [eMin]=HolmObj(Guess,x0,Coord,Scale,Initial)
        %-----------------------------------------------------------------%
        %Objective function called by fmincon
        %-----------------------------------------------------------------%

         %--OPTIMISATION PARAMETERS
             a_=Guess(1)*Scale(1);%scaling/reducing factor
             HLim_=Guess(2)*Scale(2);%estimate of y(inf) value (asymptote)
             theta0=Guess(3)*Scale(3);
         %--imported variables
             X0=Initial(1);Y0=Initial(2);

         %----Convert to calculation space
             %Origin a x0,Hlim. Regular axis direction (image, y axis reversed)
             %--y=0 at asymptote -> HLim
             Y0=-(Y0-HLim_);%image coordinates from top right corner
```

```
            %--x=0 (sphere centre)-->circle x0
            X0=X0-x0;%same direction. Always +ve.

         %--NUMERICAL INTEGRATION
            v0=[theta0,X0*a_,Y0*a_];
            Span=[0:0.01:3*pi];%why was is 20 Pi?
            [S,v] = ode45(@(S,v) holmODE(S,v),Span,v0);%integrate

        %--isolate area of interest from theoretical curve
            i=1;mono=length(v);
            while v(i,1)>1*pi/180&&i<length(v) %for theta > 10
                mono=i;
                i=i+1;
            end%determine theoretical curve before asymptote

        %--THEORETICAL CURVE (with asymptote)
            X = v(1:mono,2)/a_+x0;
            Y = -v(1:mono,3)/a_+HLim_;

            if isempty(X)==1||length(X)<=2
                eMin=2*10^10;%if solver returns only 1 point, large error
                Xxi=[]; Yyi=[];
                disp('<<holm opt>>length(X)==0')
            else %determine error for
                [eMin,Xxi,Yyi]=errorX(X,Y,Coord);
            end
        end  %x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)
    HolmOpt=[HolmOpt(1)*ScaleR(1),HolmOpt(2)*ScaleR(2),HolmOpt(3)*ScaleR(3)];%
        rescale for export
end

%% HOLM OPT INITIAL COORDINATES
function [Initial]=initialCoord(Coord,poly2)
%------------------------------------------------------------------
%Curve starting from first point (CN already adjusted)
%Estimate theta from polynomial at known X
%------------------------------------------------------------------
    %Starting coordinates
     X=Coord(1,1);
     Y=Coord(2,1);
    %calculate tangent&theta
     tangent=polyval(polyder(poly2),Y);%X=fn(Y^2)>>tangent=dX/dY
     theta=atan(-1/tangent);%(y coord reversed)
     if theta<0; theta=theta+pi;end %accounts for reversed holm
     Initial=[X,Y,theta];
     %y=m(x-x0)+y0
```

```
      %tang=tan(theta+pi);
      %X2=0.9*X; Y2=Y-tang*(X2-X);
      %plot([X,X2],[Y,Y2],'y','linewidth',2);
      %plot([X,X+100],[Y,Y],'y','linewidth',2);
end

%% ODE (REDUCED FORM - HOLM)
function [dvdS]=holmODE(S,v0)
%------------------------------------------------------------------
%Young-Laplace differential equations for ODE solver
%------------------------------------------------------------------
    %Import variables
    theta = v0(1);
    X = v0(2);
    Y = v0(3);

    % Differential equations
    if X~=0
        dthetadS = Y-sin(theta)/X;
    else
        dthetadS = Y/2;
    end
    dXdS = cos(theta);
    dYdS = sin(theta);

    % Pack derivatives for return
    dvdS = [dthetadS; dXdS; dYdS];
end

%% X-ERROR CALCULATION - HOLM
function [error,Xxi,Yyi]=errorX(X,Y,Coord)%X,Y theoretical, x,y detected
%------------------------------------------------------------------
%Error calculation between theoretical and detected profiles.
%------------------------------------------------------------------

%Make theoretical curve monotonic & determine inflection points for holm
    k=0;Mid=0;Theo=zeros(2,length(Y));
    for i=1:length(Y)-1
        if Y(i)~=Y(i+1)%&& X(i)<=X(1,end)%monotonic Y for range X = HolmCoord(1,:)
            k=k+1;
            Theo(1,k)=X(i);
            Theo(2,k)=Y(i);
        end
    end%Theoretical curve - coordinates at unique Y values only
    Theo=Theo(:,1:k);%remove empty elements
    if isempty(Theo)==1; disp('<<error1D>>Theo empty'); end
```

```
    endflag=false;Mid=0;
    for i=1:length(Theo)-1
        if Theo(1,i)>=Theo(1,i+1)&&endflag==false
            Mid=i;
        else endflag=true;
        end
    end

    if isempty(Coord)==1; disp('<<error1D>>Coord empty'); end
    MidH=0;CE=length(Coord);
    if Mid==0; %disp('<<error1D>>Mid==0');
        for i=1:length(Coord)
            if Theo(2,end)<Coord(2,i); CE=i;end
        end
    elseif Mid~=0
        endflag=false;
        for i=1:length(Coord)
            if Coord(2,i)>=Theo(2,Mid)&&endflag==false
                MidH=i;
            else endflag=true;
            end
            if Theo(2,end)>Coord(2,i); CE=i;end
        end
        %if MidH==0; disp('<<error1D>>MidH==0 (Coord) but Mid~=0 (theo) - lower section empty'); end
    else %disp('<<error1D>>Mid<0');
    end

%INTERPOLATE
   %split X,Y, HolmCoord into monotonic sections
    YyiU=Coord(2,MidH+1:CE);%Y limit at highest point of theo curve
    XxiL=[];XxiU=[];flagempty=true;
    if Mid==0;Mid=1;%not reversed holm - remove lower portion
    else%reversed holm - interpolate lower section
        YyiL=Coord(2,1:MidH);flagempty=false;
        if length(Theo(1,1:Mid))>1%interp requires two points
            XxiL=interp1(Theo(2,1:Mid),Theo(1,1:Mid),YyiL,'linear');%lower region
        else %disp('<<error1D>>no lower coordinates');
        end
    end
    if length(Theo(1,Mid+1:end))>1
        XxiU=interp1(Theo(2,Mid+1:end),Theo(1,Mid+1:end),YyiU,'linear');%upper region
    else disp('<<error1D>>no upper coordinates');
    end

```

```
    if flagempty==true; YyiCat=YyiU; XxiCat=XxiU;
    else XxiCat=[XxiL,XxiU];YyiCat=[YyiL,YyiU];%catonate
    end

%REMOVE NaN >> matlab will return NaN if asked to EXTRAPOLATE using interp1, linear
%& ADD ARTIFICIAL ASYMPTOTE (bias unfeasible solutions)
    Xxi=zeros(1,length(Coord));Yyi=zeros(1,length(Coord));
    Ymax=Coord(2,1);
    for i=1:length(Coord)
        if i>length(XxiCat)%asymptote
            Xxi(i)=Coord(1,i);
            Yyi(i)=Ymax;
        elseif isnan(XxiCat(i))==0%not NaN
            Xxi(i)=XxiCat(i);
            Yyi(i)=YyiCat(i); Ymax=YyiCat(i);
        else %is NaN >> out of range. Replace with Xcoord,max(Yyi) within range
            Xxi(i)=Coord(1,i);
            Yyi(i)=Ymax;
        end
    end

%DETERMINE ERROR
    if length(Xxi)<=1; error=10^7*length(Theo)^2*length(Coord)^2;disp('<<error1D>> length(Xxi)<=1');
    else
        Error=zeros(1,length(Coord));
        for i=1:length(Error)
            Error(i)=(Xxi(i)-Coord(1,i))^2+(Yyi(i)-Coord(2,i))^2;
            %plot([Xxi(i),Coord(1,i)],[Yyi(i),Coord(2,i)],'y')
        end%error based on x&y difference >> y differences if 'artificial' point
        error=sqrt(sum(Error)/length(Error));%standard deviation
    end
    %plot(Xxi,Yyi);
end
%-----------------------------------------------------------------
function myCleanupFun(fileID)
    fclose(fileID);
end
```

B.3 ThreshGUI (interactive thresholding and edge identification)

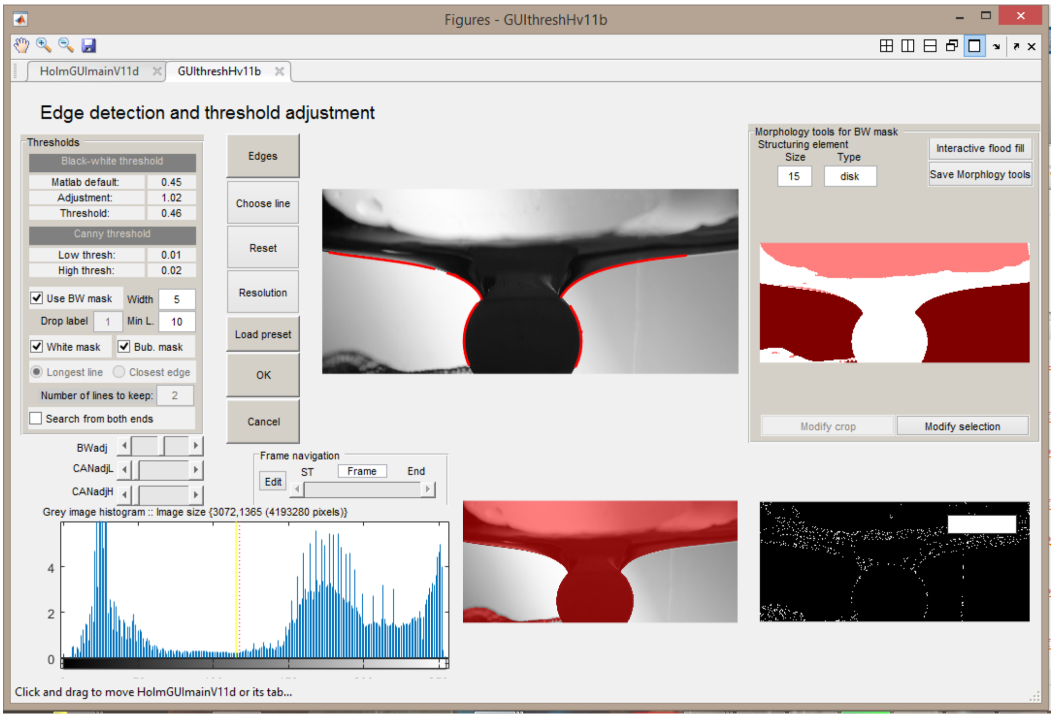

Figure B.2: Interactive thresholding GUI.

Listing B.3: GUIthreshHv11c.m

```
function varargout = GUIthreshHv11c(varargin)
% GUITHRESHHV11C MATLAB code for GUIthreshHv11c.fig
%      GUITHRESHHV11C, by itself, creates a new GUITHRESHHV11C or raises the existing
%      singleton*.
%
%      H = GUITHRESHHV11C returns the handle to a new GUITHRESHHV11C or the handle to
%      the existing singleton*.
%
%      GUITHRESHHV11C('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUITHRESHHV11C.M with the given input arguments.
%
%      GUITHRESHHV11C('Property','Value',...) creates a new GUITHRESHHV11C or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before GUIthreshHv11c_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to GUIthreshHv11c_OpeningFcn via varargin.
%
```

```
%       *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%       instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help GUIthreshHv11c

% Last Modified by GUIDE v2.5 29-Nov-2017 10:05:54

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @GUIthreshHv11c_OpeningFcn, ...
                   'gui_OutputFcn',  @GUIthreshHv11c_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before GUIthreshHv11c is made visible.
function GUIthreshHv11c_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to GUIthreshHv11c (see VARARGIN)

% Choose default command line output for GUIthreshHv11c
handles.output = hObject;
%-----------------------------------------------------------------
%.. Set Handles data in desktop
    GUI_Thresh = gcf;
    setappdata(0,'GUI_Thresh',GUI_Thresh);
    GUI_Hmain = getappdata(0,'GUI_Hmain');
%.. Get function handles for TRI and IMMASK
    handles.hImMask = getappdata(GUI_Hmain,'hImMask');
```

```
    handles.hWIM = getappdata(GUI_Hmain,'hWIM');
    handles.hTri = getappdata(GUI_Hmain,'hTri');
    handles.hOrderCoord = getappdata(GUI_Hmain,'hOrderCoord');

%.. Load data
    Tag = getappdata(GUI_Hmain, 'Tag');
    handles.Path=getappdata(GUI_Hmain,'Path');
    load(fullfile(handles.Path,sprintf('%sThI',Tag)));
 if exist(fullfile(handles.Path,sprintf('%sThO.mat',Tag)),'file')==2
    load(fullfile(handles.Path,sprintf('%sThO',Tag)));
    sz=size(size(Grey));
    if sz(2)==3
        Grey=rgb2gray(Grey);
    end
    handles.Grey=Grey;
    handles.SphCnr=SphCnr;
    handles.HlmCnr=HlmCnr;
    handles.ST=ST;
    handles.End=End;
    handles.Intv=Intv;
    handles.Prefix=Prefix;
    handles.Extn=Extn;
    handles.Crop=Crop;
    if(exist('NL','var')==1)%allowing for old files
        handles.NL=NL;
    else
        handles.NL=2; %default
    end

%.. Update thresholds
    handles.BWadj=BWadj;
        set(handles.T_Adj,'string',sprintf('%0.2d',BWadj));
        set(handles.SL_BW,'value',BWadj);
    handles.CANadjL=CANadjL;
        set(handles.T_CL,'string',sprintf('%0.2d',CANadjL));
        set(handles.SL_CanL,'value',BWadj);
    handles.CANadjH=CANadjH;
        set(handles.T_CH,'string',sprintf('%0.2d',CANadjH));
        set(handles.SL_CanH,'value',BWadj);
    handles.wd = wd;
        set(handles.ET_wd,'string',sprintf('%0.2d',wd));
    handles.SBD = SBD; %default
        if SBD ==1
            set(handles.CB_SBE,'value',1)
        end
%.. Default settings for mask
```

```
    set(handles.ET_SEsize,'string',sprintf('%0.1d',MorphProps.Size));
    set(handles.ET_SEtype,'string',sprintf('%s',MorphProps.Type));
    handles.MorphProps=MorphProps;
else
%-------------------------------------------------------------------
% RUN EDGE DETECTION USING MATLAB'S VALUES AND FILL ALL FIGURES
%-------------------------------------------------------------------
    sz=size(size(Grey));
    if sz(2)==3
        Grey=rgb2gray(Grey);
    end
    handles.Grey=Grey;
    handles.SphCnr=SphCnr;
    handles.HlmCnr=HlmCnr;
    handles.ST=ST;
    handles.End=End;
    handles.Intv=Intv;
    handles.Prefix=Prefix;
    handles.Extn=Extn;
    handles.Crop=Crop;
    handles.NL=2;

%.. Set new thresholds
    if isfield(handles,'BWadj')==0
        handles.BWadj=1;
    end
    if isfield(handles,'CANadjL')==0
        handles.CANadjL=[];
    end
    if isfield(handles,'CANadjH')==0
        handles.CANadjH=[];
    end
    if isfield(handles,'wd')==0
        handles.wd = 5;
    end

%.. Load width from GUI
    handles.wd = str2double(get(handles.ET_wd,'String'));
    handles.SBD = 0; %default
%.. Default settings for mask
    MorphProps.Size=15;
    MorphProps.Type='disk';
    MorphProps.FilLoc=[];
    handles.MorphProps=MorphProps;

end
```

```
%.. Update function
    [BWadj,CANadjL,CANadjH]=UpdatePlotsH(SphCnr,HlmCnr,Grey,handles.BWadj,handles.
        CANadjL, handles.CANadjH,handles);
    handles.CANadjL=CANadjL;
    handles.CANadjH=CANadjH;

%.. Set slider starting positions
    set(handles.SL_CanL,'Value',CANadjL);
    set(handles.SL_CanH,'Value',CANadjH);

%.. Show image histogram
    axes(handles.Ax_Hist); hold on
%     nn = hist( Grey(:), 0:255 ); % histogram for 0..255 bins
%     bar( 0:255, nn*numel(Grey)/100 );
    set(handles.ST_HistLab,'String',sprintf('Grey image histogram :: Image size {%
        d,%d (%d pixels)}',size(Grey,2),size(Grey,1),numel(Grey)));
    imhist(Grey),hold on
    xlim([-2 257]);
%.. Update handles structure
    guidata(hObject, handles);

%.. Give instructions
    h=msgbox('Use the sliders to adjust the thresholding values for BW and Canny.
        \n Close this box to continue.');
    uiwait(h);
%--------------------------------------------------------------------------

% --- Executes during object creation, after setting all properties.
function figure1_CreateFcn(hObject, eventdata, handles)%Main figure createfnc.
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% --- Outputs from this function are returned to the command line.

function varargout = GUIthreshHv11c_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on slider movement.
function SL_CanL_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to SL_CanL (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%--------------------------------------------------------------------
%.. Get slider value
    CANadjL = get(hObject,'Value');
%.. Save slider value
    handles.CANadjL =CANadjL;
    CANadjH=handles.CANadjH;
    set(handles.T_CL,'string',sprintf('%0.3f',CANadjL));
%.. Edge image
    if CANadjL>=CANadjH
       ThreshC=[0.95*CANadjH,CANadjH];
    else
       ThreshC=[CANadjL,CANadjH];
    end
    axes(handles.Ax_Edge);imshow(edge(handles.Grey,'canny',ThreshC));
%.. Update images & handles
    guidata(hObject, handles);
%--------------------------------------------------------------------

% --- Executes during object creation, after setting all properties.
function SL_CanL_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SL_CanL (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function SL_BW_Callback(hObject, eventdata, handles)
% hObject    handle to SL_BW (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%--------------------------------------------------------------------
%.. Get slider value
    BWadj = get(hObject,'Value');
%.. Save slider value
    handles.BWadj=BWadj;
%.. BW image
    %--Create mask
```

```
    [WIM, ThreshC, Thr]=handles.hWIM(handles.Grey, BWadj, handles.CANadjL, handles.
        CANadjH, handles.MorphProps, 1, handles.Path);
    axes(handles.Ax_BW); cla
    imshow(WIM); hold on
    %--Plain BW
    Thr=graythresh(handles.Grey);
    BW=im2bw(handles.Grey, BWadj*Thr);
    %--Display a solid green "image" on top of the original image.
    green = cat(3, ones(size(WIM)), zeros(size(WIM)), zeros(size(WIM)));
    transMask=imshow(green);
    hold off
    %-- Use the influence map pixels to control the transparency of each pixel of
        the green image.
    set(transMask, 'AlphaData', 0.5*BW)
    %axes(handles.Ax_BW); imshow(BW2-0.5*BW1);

    axes(handles.Ax_Hist);
    hold off, imhist(handles.Grey), hold on
    xlim([-2 257]);
    plot([Thr*255,Thr*255],[0,10e4],'y');%Plot threshold
    plot([BWadj*Thr*255,BWadj*Thr*255],[0,10e5],':r');%Plot threshold

    axes(handles.Ax_Edge);
    Edge=edge(handles.Grey,[handles.CANadjL, handles.CANadjH],'Canny');

    imshow(Edge);
%.. Update images & handles
    set(handles.T_de,'String',sprintf('%1.2f',Thr));
    set(handles.T_Thr,'String',sprintf('%1.2f',Thr*BWadj));
    set(handles.T_Adj,'String',sprintf('%1.2f',BWadj));
    legend('Pixel information','Default threshold','User threshold');
    guidata(hObject, handles);
    %UpdatePlotsH(handles.SphCnr, handles.HlmCnr, handles.Grey, handles.BWadj, handles
        .CANadjL, handles.CANadjH, handles);
%--------------------------------------------------------------------------


% --- Executes on slider movement.
function SL_CanH_Callback(hObject, eventdata, handles)
% hObject    handle to SL_CanH (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%--------------------------------------------------------------------------
% handles    structure with handles and user data (see GUIDATA)
%--------------------------------------------------------------------------
%.. Get slider value
```

```
    CANadjH = get(hObject,'Value');
%.. Save slider value
    handles.CANadjH=CANadjH;
    CANadjL=handles.CANadjL;
    set(handles.T_CH,'string',sprintf('%0.3f',CANadjH));
%.. Edge image
    if CANadjL>=CANadjH
        ThreshC=[0.95*CANadjH,CANadjH];
    else
        ThreshC=[CANadjL,CANadjH];
    end
    axes(handles.Ax_Edge);imshow(edge(handles.Grey,'canny',ThreshC));
%.. Update images & handles
    guidata(hObject, handles);
%------------------------------------------------------------




%.. Update images & handles
    guidata(hObject, handles);
%------------------------------------------------------------


% --- Executes during object creation, after setting all properties.
function SL_CanH_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SL_CanH (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'
    ))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

% --- Executes during object creation, after setting all properties.
function SL_BW_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SL_BW (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'
    ))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
```

```
end


% --- Executes on button press in PB_OK.
function PB_OK_Callback(hObject, eventdata, handles)
% hObject    handle to PB_OK (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%--------------------------------------------------------------------
% SAVE DATA FOR THE MAIN GUI
%--------------------------------------------------------------------
%.. Load tag and save thresholds
    GUI_Hmain = getappdata(0, 'GUI_Hmain');
    Tag = getappdata(GUI_Hmain, 'Tag');
    BWadj=handles.BWadj;
    CANadjL=handles.CANadjL;
    CANadjH=handles.CANadjH;
    wd=handles.wd;
    minL=str2double(get(handles.ET_MinL,'String'));
    SBD=get(handles.CB_SBE,'Value');
    MorphProps=handles.MorphProps;
    SphCnr=handles.SphCnr;
    HlmCnr=handles.HlmCnr;
    NL=handles.NL;
%--------------------------------------------------------------------
%.. Save and Close:
    save(fullfile(handles.Path,sprintf('%sThO',Tag)),'BWadj','CANadjL','CANadjH','
        wd','minL','SBD','MorphProps','SphCnr','HlmCnr','NL');
    fprintf(1,'Threshold data saved. Returning to main GUI.\n');
%.. Call handles
    GUI_Thresh = getappdata(0,'GUI_Thresh');
%.. Close GUI
    close(GUI_Thresh);
%--------------------------------------------------------------------


% --- Executes on button press in PB_Cancel.
function PB_Cancel_Callback(hObject, eventdata, handles)
% hObject    handle to PB_Cancel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%--------------------------------------------------------------------
%.. Call handles
    GUI_Thresh = getappdata(0,'GUI_Thresh');
%.. Close GUI
    fprintf(1,'Closing Thresh GUI. File not saved.\n');
```

```
    close(GUI_Thresh);
%------------------------------------------------------------------------

% --- Executes on button press in PB_Reset.
function PB_Reset_Callback(hObject, eventdata, handles)
% hObject    handle to PB_Reset (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%------------------------------------------------------------------------
%.. Reset values
    handles.BWadj=1;
    handles.CANadjL=[];
    handles.CANadjH=[];
    guidata(hObject, handles);
%.. Update
    [~,CANadjL,CANadjH]=UpdatePlotsH(handles.SphCnr,handles.HlmCnr,handles.Grey,
        handles.BWadj,handles.CANadjL, handles.CANadjH,handles);
    handles.CANadjL=CANadjL;
    handles.CANadjH=CANadjH;
    fprintf(1,'Reset to default values.\n');
%------------------------------------------------------------------------

%% UpdatePlot
function [BWadj,CANadjL,CANadjH]=UpdatePlotsH(SphCnr,HlmCnr,Grey,BWadj,CANadjL,
    CANadjH,handles)
%------------------------------------------------------------------------
%--Update the GUI interface as the user interacts.
%------------------------------------------------------------------------
try %matlab may be unable to detect the edge with the default coordinates. Return
    blank but load GUI.
%--EDGE DETECTION
%......................................................................
%------------------------------------------------------------------------
% GENERATE WHOLE IMAGE MASK ON BW IMAGE
%------------------------------------------------------------------------
    [WIM,ThreshC,Thr]=handles.hWIM(Grey,BWadj,CANadjL,CANadjH,handles.MorphProps
        ,1,handles.Path);

%.. Sphere Left & Right sides
    [SphCrdL,SphCrdR]=handles.hImMask(SphCnr(2,:),SphCnr(1,:),Grey,ThreshC,Thr,1,
        handles.wd,0,WIM,handles.Path,handles.NL);

%.. Holm Left and Right sides
    [HolmCrdL,HolmCrdR]=handles.hImMask(HlmCnr(2,:),HlmCnr(1,:),Grey,ThreshC,Thr
        ,1,handles.wd,handles.SBD,WIM,handles.Path,handles.NL);

```

```
%.. Load display files
    load(fullfile(handles.Path,'ThreshTemp1'));
    load(fullfile(handles.Path,'ThreshTemp2'));

%.. Display
    %-- Display mask on "MOrph tools" axes
        CC=bwconncomp(WIM);
        L=labelmatrix(CC);%Create label matrix
        axes(handles.Ax_BW); cla;
        imshow(label2rgb(L));
    %-- Clear AX_BWmask
        cla(handles.Ax_BWmask);
        axes(handles.Ax_BWmask);
    %-- Dsplay the original GREY image.
        imshow(Grey);
        hold on
    %-- Display a solid green "image" on top of the original image.
        green = cat(3, ones(size(WIM)),zeros(size(WIM)), zeros(size(WIM)));
        transMask=imshow(green);
        hold off
    %-- Use the influence map pixels to control the transparency of each pixel of
        the green image.
        set(transMask, 'AlphaData', 0.5*WIM)

    %.. Main
        axes(handles.Ax_Main), imshow(Grey),hold on
        plot(SphCrdL(:,1),SphCrdL(:,2),'.r','linewidth',2);
        plot(SphCrdR(:,1),SphCrdR(:,2),'.r','linewidth',2);
        plot(HolmCrdL(:,1),HolmCrdL(:,2),'.r','linewidth',2);
        plot(HolmCrdR(:,1),HolmCrdR(:,2),'.r','linewidth',2);

    %.. Update tags
    set(handles.T_de,'String',sprintf('%0.2f',Thr));
    set(handles.T_Adj,'String',sprintf('%0.2f',BWadj));
    set(handles.T_Thr,'String',sprintf('%0.2f',BWadj*Thr));
    set(handles.T_CL,'String',sprintf('%0.2f',CANadjL));
    set(handles.T_CH,'String',sprintf('%0.2f',CANadjH));
catch
   disp('Matlab failed to generate an edge using the default coordinates. Failed
       in "update plots".')
end

    GUI_Hmain=getappdata(0,'GUI_Hmain');
    setappdata(GUI_Hmain,'ImType','Default');

```

```
    fprintf(1,'BW adj: %1.2f; Canny Low: %1.2f, Canny High: %1.2f', BWadj, CANadjL
        , CANadjH);
%--------------------------------------------------------------------

% --- Executes during object creation, after setting all properties.
function Tog_Mask_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Tog_Mask (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
%--------------------------------------------------------------------
% Default value is "use mask" - adjustment not required.
% Thresholds will update if toggled.
%--------------------------------------------------------------------
% --- Executes on button press in Tog_Mask.

function Tog_Mask_Callback(hObject, eventdata, handles)
% hObject    handle to Tog_Mask (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%--------------------------------------------------------------------
% FLAGS THAT NO MASK SHOULD BE USED TO DETECT THE EDGE - CANNY ONLY
%--------------------------------------------------------------------
%.. Read toggle state
    MaskFlag = get(hObject,'Value');
%.. If inactive, the mask should not exclude any edges areas:
    if MaskFlag == 0
        %Disable mask (thresh = 1)
        BWadj = 0; %--> set as zero, then reverse max()=0 in ImMask
        %Disable BW slider
        set(handles.SL_BW, 'enable','off');
        set(handles.ET_wd, 'enable','off');
        set(handles.RB_Close, 'enable','on');
        set(handles.RB_Longest, 'enable','on');
        set(handles.ET_NL, 'enable','on');%allows the user to specify the number
            of lines
        %set(handles.CB_SBE,'enable','on');
    else
%.. If active, the mask behaves normally. Re-enable the slider and call old adj.
        %Enable slider
        set(handles.SL_BW, 'enable','on');
        set(handles.ET_wd, 'enable','on');
        %Reset old value
        BWadj = get(handles.SL_BW,'Value');
        %..Disable
        set(handles.RB_Close, 'enable','off');
        set(handles.RB_Longest, 'enable','off');
```

```
        set(handles.ET_NL, 'enable','off');
        %set(handles.CB_SBE,'enable','off');
    end
%.. Update plots
        [~,CANadjL,CANadjH]=UpdatePlotsH(handles.SphCnr,handles.HlmCnr,handles.
            Grey,BWadj,handles.CANadjL, handles.CANadjH,handles);
%.. Update handles
        handles.BWadj=BWadj;
        guidata(hObject, handles);
%--------------------------------------------------------------------



function ET_wd_Callback(hObject, eventdata, handles)
% hObject    handle to ET_wd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%--------------------------------------------------------------------
% DESCRIBES THE WIDTH OF THE MASK ON EITHER SIDE OF THE EDGE
%--------------------------------------------------------------------
%.. Get value
    handles.wd = str2double(get(hObject,'String'));
    guidata(hObject, handles);

%.. Update GUI
    [~,CANadjL,CANadjH]=UpdatePlotsH(handles.SphCnr,handles.HlmCnr,handles.Grey,
        handles.BWadj,handles.CANadjL,handles.CANadjH,handles);

%--------------------------------------------------------------------


% --- Executes during object creation, after setting all properties.
function ET_wd_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_wd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on slider movement.
```

```matlab
function SL_Frame_Callback(hObject, eventdata, handles)
% hObject    handle to SL_Frame (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function SL_Frame_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SL_Frame (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider4_Callback(hObject, eventdata, handles)
% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function slider4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on button press in PB_ChooseLine.
```

```
function PB_ChooseLine_Callback(hObject, eventdata, handles)
%-----------------------------------------------------------------------
%--USE-SPECIFIED EDGE DETECTION FOR STILLS & SEQUENCES
%-----------------------------------------------------------------------
ST=handles.ST;
FTA=handles.End;
Intv=handles.Intv;
Prefix=handles.Prefix;
Extn=handles.Extn;
Crop=handles.Crop;

CANadjH=handles.CANadjH;
CANadjL=handles.CANadjL;
%.. Thresholds
    if isempty(CANadjL)==1||isempty(CANadjH)==1 %if canny parameters aren't set
        ThreshC=[];
    else
        if CANadjL>=CANadjH
            ThreshC=[0.95*CANadjH,CANadjH];
        else
            ThreshC=[CANadjL,CANadjH];
        end
    end

if exist(fullfile(handles.Path,'UserEdges'))==0
    save(fullfile(handles.Path,'UserEdges'),'ST','FTA','Intv');
end

choice = questdlg('Run sequence, or current image?');
% Handle response
switch choice
    case 'Current'
        Grey=handles.Grey;

            %
                ....................................................................

    %-----------------------------------------------------------------------
    % GENERATE WHOLE IMAGE MASK ON BW IMAGE
    %-----------------------------------------------------------------------
    WIM=WholeImageMask(Grey,BWadj,CANadjL,CANadjH,MorphProps);

    %.. Sphere Left & Right sides (auto)
        SphCnr=handles.SphCnr;
        [SphCrdL,SphCrdR]=handles.hImMask(SphCnr(2,:),SphCnr(1,:),Grey,handles.
            BWadj,handles.CANadjL,handles.CANadjH,1,handles.wd,0,WIM);
```

```
%.. Holm Left side
    Cnr=handles.HlmCnr(2,:);
    GreyL=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));

    EdgeL=edge(GreyL,'canny',ThreshC);
    cntflg=false;
    [Stats,RegionsOfInterest]=RegionColour(EdgeL,20,2);
    R=inputdlg('Which lines should be taken?');

    while cntflg==false
        R=str2num(R{:});
         %--Combine all as linear indices
        IND=[];
        for k=R
            IND=[IND;Stats(k).PixelIdxList];%linear indices
        end %K = R plot

     %--Convert Linear indices, order
        s=size(EdgeL);%size of 'grey' image for converting linear indices
        [y,x]=ind2sub(s,IND);
        plot(x,y,'g.');

        R=inputdlg('Edit the lines or press cancel to continue.');
        if isempty(R)==1
            cntflg=true;
        else
            [Stats,RegionsOfInterest]=RegionColour(EdgeL,20,2);
        end
    end
    CrdL=sortrows([x,y],2);%sort rows
    HolmCrdL(:,1)=CrdL(:,1)+handles.HlmCnr(2,1);
    HolmCrdL(:,2)=CrdL(:,2)+handles.HlmCnr(2,2);

%.. Holm Right side
    Cnr=handles.HlmCnr(1,:);
    GreyR=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));
    EdgeR=edge(GreyR,'canny');
    [Stats,RegionsOfInterest]=RegionColour(EdgeR,20,2);
    R=inputdlg('Which lines should be taken?');
    cntflg=false;
    while cntflg==false
        R=str2num(R{:});
         %--Combine all as linear indices
        IND=[];
        for k=R
```

```
                IND=[IND; Stats(k).PixelIdxList];%linear indices
            end %K = R plot

        %--Convert Linear indices, order
            s=size(EdgeR);%size of 'grey' image for converting linear indices
            [y,x]=ind2sub(s,IND);
            plot(x,y,'g.');

            R=inputdlg('Edit the lines or press cancel to continue.');
            if isempty(R)==1
                cntflg=true;
            else
                [Stats,RegionsOfInterest]=RegionColour(EdgeR,20,2);
            end
        end
        CrdR=sortrows([x,y],2);%sort rows
        HolmCrdR(:,1)=CrdR(:,1)+handles.HlmCnr(1,1);
        HolmCrdR(:,2)=CrdR(:,2)+handles.HlmCnr(1,2);

    %.. Display

        axes(handles.Ax_Main), imshow(Grey),hold on
        plot(SphCrdL(:,1),SphCrdL(:,2),'.r');
        plot(SphCrdR(:,1),SphCrdR(:,2),'.r');
        plot(HolmCrdL(:,1),HolmCrdL(:,2),'.r');
        plot(HolmCrdR(:,1),HolmCrdR(:,2),'.r');

        UsrEdge.k=ki;
        UsrEdge.HolmCrdL=HolmCrdL;
        UsrEdge.HolmCrdR=HolmCrdR;
        UsrEdge.SphCrdL=SphCrdL;
        UsrEdge.SphCrdR=SphCrdR;

        FrameIdenti=sprintf('UsrEdge%04i',ki);
        eval([FrameIdenti '=UsrEdge'])
        save(fullfile(handles.Path,'UserEdges'),FrameIdenti,'-append')



    case 'Sequence'


    for k = ST : Intv : FTA %subsequent images
            fprintf('--- processing frame %i (from %d to %d)',k, ST, FTA);
            %---Clear old data
            clear HolmCrdL HolmCrdR SphereCrdL SphereCrdR
```

```
                %------------------------------------------------------------------------
                %---Find a frame which can be read----------------------------------------
                %------------------------------------------------------------------------
                 CntFlag=false;ki=k-1;%find a frame which can be read
        while CntFlag==false && ki<=k+Intv,%if flag is false, have not found an image to read
                ki=ki+1;  %(on first entry, ki=k-1+1=k
          try
             ImFile=fullfile(handles.Path,sprintf('%s%04i.%s',Prefix,ki,Extn));
             disp(ImFile);
             Grey = imread(ImFile);
             Cnr=Crop;
             Grey=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));
             CntFlag=true;

    %--EDGE DETECTION
    %.......................................................................
    %----------------------------------------------------------------------
    % GENERATE WHOLE IMAGE MASK ON BW IMAGE
    %----------------------------------------------------------------------
    WIM=WholeImageMask(Grey,BWadj,CANadjL,CANadjH,MorphProps);

    %.. Sphere Left & Right sides (auto)
        SphCnr=handles.SphCnr;
        [SphCrdL,SphCrdR]=handles.hImMask(SphCnr(2,:),SphCnr(1,:),Grey,handles.BWadj,handles.CANadjL,handles.CANadjH,1,handles.wd,0,WIM);

    %.. Holm Left side
        Cnr=handles.HlmCnr(2,:);
        GreyL=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));

        EdgeL=edge(GreyL,'canny',ThreshC);
        cntflg=false;
        [Stats,RegionsOfInterest]=RegionColour(EdgeL,20,2);
        R=inputdlg('Which lines should be taken?');

        while cntflg==false
            R=str2num(R{:});
             %--Combine all as linear indices
            IND=[];
            for k=R
```

```
                IND=[IND; Stats(k).PixelIdxList];%linear indices
            end %K = R plot

         %--Convert Linear indices, order
            s=size(EdgeL);%size of 'grey' image for converting linear indices
            [y,x]=ind2sub(s,IND);
            plot(x,y,'g.');

            R=inputdlg('Edit the lines or press cancel to continue.');
            if isempty(R)==1
                cntflg=true;
            else
                [Stats,RegionsOfInterest]=RegionColour(EdgeL,20,2);
            end
        end
        CrdL=sortrows([x,y],2);%sort rows
        HolmCrdL(:,1)=CrdL(:,1)+handles.HlmCnr(2,1);
        HolmCrdL(:,2)=CrdL(:,2)+handles.HlmCnr(2,2);

    %.. Holm Right side
        Cnr=handles.HlmCnr(1,:);
        GreyR=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));
        EdgeR=edge(GreyR,'canny');
        [Stats,RegionsOfInterest]=RegionColour(EdgeR,20,2);
        R=inputdlg('Which lines should be taken?');
        cntflg=false;
        while cntflg==false
            R=str2num(R{:});
             %--Combine all as linear indices
            IND=[];
            for k=R
                IND=[IND; Stats(k).PixelIdxList];%linear indices
            end %K = R plot

         %--Convert Linear indices, order
            s=size(EdgeR);%size of 'grey' image for converting linear indices
            [y,x]=ind2sub(s,IND);
            plot(x,y,'g.');

            R=inputdlg('Edit the lines or press cancel to continue.');
            if isempty(R)==1
                cntflg=true;
            else
                [Stats,RegionsOfInterest]=RegionColour(EdgeR,20,2);
            end
        end
```

```
        CrdR=sortrows([x,y],2);%sort rows
        HolmCrdR(:,1)=CrdR(:,1)+handles.HlmCnr(1,1);
        HolmCrdR(:,2)=CrdR(:,2)+handles.HlmCnr(1,2);

    %.. Display

        axes(handles.Ax_Main), imshow(Grey),hold on
        plot(SphCrdL(:,1),SphCrdL(:,2),'.r');
        plot(SphCrdR(:,1),SphCrdR(:,2),'.r');
        plot(HolmCrdL(:,1),HolmCrdL(:,2),'.r');
        plot(HolmCrdR(:,1),HolmCrdR(:,2),'.r');

        UsrEdge.k=ki;
        UsrEdge.HolmCrdL=HolmCrdL;
        UsrEdge.HolmCrdR=HolmCrdR;
        UsrEdge.SphCrdL=SphCrdL;
        UsrEdge.SphCrdR=SphCrdR;

        FrameIdenti=sprintf('UsrEdge%04i',ki);
        eval([FrameIdenti '=UsrEdge'])
        save(fullfile(handles.Path,'UserEdges'),FrameIdenti,'-append')

        catch errR
             fprintf(1,'----------Unable to read frame %04i.  \r\n', ki);
        end%End for try
      end%end for inner loop
    end%for loop through frames
end%case

GUI_Hmain=getappdata(0,'GUI_Hmain');
setappdata(GUI_Hmain,'ImType','User');
%--------------------------------------------------------------------------


%% RegionColour
function [Stats,RegionsOfInterest]=RegionColour(Edges,n,figE)%n=number of lines to show
%--------------------------------------------------------------------------
%This program identifies the regions of BW edge image and returns the edge
%matrix with the n largest regions coloured and identified.
%n is int or 'All'
%--------------------------------------------------------------------------
    figure(figE);
    hold off
    imshow(Edges);
    hold on
```

```

    [L,num]=bwlabel(Edges);%repeat?
    if strcmp(n,'All')==1
        n=num;
    elseif n>num
        n=num;
    end
    Stats=regionprops(L,'Area','PixelIdxList');
    Area=zeros(num,2);
    for R=1:num %region
        Area(R,:)=[R,Stats(R).Area];
    end %R
    Area=flipud(sortrows(Area,2));%sort based on area.  flipup(ascending)=
        descending.

    %Identify n regions of interest to plot
    RegionsOfInterest=Area(1:n,1);

    %plot regions of interest
    s=size(Edges);%size of 'grey' image for converting linear indices

    for k=1:n
        R=RegionsOfInterest(k,1);
        if Stats(R).Area>9
            IND=Stats(R).PixelIdxList;%linear indices
            [y,x]=ind2sub(s,IND);
            plot(x,y,'.');
            h = text(x(1)+1, y(1)-1, num2str(R));%plot region name
            set(h,'Color','y',...
            'FontSize',10,'FontWeight','bold');
        end%plot edges with >9 pixles
    end %K = R plot


% --- Executes on button press in PB_Edge.
function PB_Edge_Callback(hObject, eventdata, handles)
%------------------------------------------------------------------------
%Edge detection
UpdatePlotsH(handles.SphCnr,handles.HlmCnr,handles.Grey,handles.BWadj,handles.
    CANadjL, handles.CANadjH,handles);
fprintf('\n Done.')
%------------------------------------------------------------------------


% --- Executes during object creation, after setting all properties.
function Ax_BW_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to Ax_BW (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate Ax_BW


% --- Executes during object creation, after setting all properties.
function Ax_Edge_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Ax_Edge (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: place code in OpeningFcn to populate Ax_Edge


% --- Executes on button press in RB_Longest.
function RB_Longest_Callback(hObject, eventdata, handles)
%-------------------------------------------------------------
%--Toggle to search for the longest line (no mask)
%..If Toggle is on, RB_Close should be off.
if get(hObject, 'Value')==1
    set(handles.RB_Close,'Value',0);
    set(handles.ET_NL, 'enable','on')
end
%-------------------------------------------------------------


% --- Executes on button press in RB_Close.
function RB_Close_Callback(hObject, eventdata, handles)
%-------------------------------------------------------------
%--Toggle to use image masking and return the closest edge points
%..If Toggle is on, RB_Longest should be off.
if get(hObject, 'Value')==1
    set(handles.RB_Longest,'Value',0);
    set(handles.ET_NL, 'enable','off')
    %..LEFT SIDE
    axes(handles.Ax_BW); hold off
    Cnr=handles.HlmCnr(2,:);
    GreyL=handles.Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));

    [Edge2]=SubMask(GreyL,handles.BWadj);
    imshow(Edge2);hold on
    [Edge3]=CoordSearch(Edge2);
    [Coord]=ExtractCoord(Edge3);

```

```
    axes(handles.Ax_Main); imshow(handles.Grey), hold on
    Coord(:,1)=Coord(:,1)+handles.HlmCnr(2,1);
    Coord(:,2)=Coord(:,2)+handles.HlmCnr(2,2);

    CoordL=sortrows(Coord,2);%sort rows
    axes(handles.Ax_Main);
    plot(CoordL(:,1),CoordL(:,2),'r')
    %edge
    axes(handles.Ax_BWmask);
    plot(CoordL(:,1),CoordL(:,2),'r')

        %..RIGHT SIDE
    axes(handles.Ax_Edge); hold off
    Cnr=handles.HlmCnr(1,:);
    GreyR=handles.Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));

    [Edge2]=SubMask(fliplr(GreyR),handles.BWadj);
    imshow(Edge2);hold on
    [Edge3]=CoordSearch(Edge2);

    [Coord]=ExtractCoord(fliplr(Edge3));
    Coord(:,1)=Coord(:,1)+handles.HlmCnr(1,1);
    Coord(:,2)=Coord(:,2)+handles.HlmCnr(1,2);

    CoordR=sortrows(Coord,2);%sort rows
    axes(handles.Ax_Main);
    plot(CoordR(:,1),CoordR(:,2),'r')
    axes(handles.Ax_BWmask);
    plot(CoordR(:,1),CoordR(:,2),'r')

    disp('Done')
end
%-----------------------------------------------------------------------------------


function [Edge2]=SubMask(Grey,BWadj)
%--BW Mask
Thr=graythresh(Grey);
BW=im2bw(Grey,0.7*Thr);
CCb=bwconncomp(~BW);
Lb=labelmatrix(CCb);
imshow(label2rgb(Lb))%display

 Stats = regionprops(CCb, 'Area');
Midx = find([Stats.Area] > 90);
Mask = ismember(labelmatrix(CCb), Midx);
```

```
Mask(Lb==1)=0;%Assuming blank component 1
Mask=~Mask;
imshow(Mask);%Black forground (=0) - background = 1

%--Subtraction mask
Sidx = find([Stats.Area] < 90);
SubMask = ismember(labelmatrix(CCb), Sidx);
SubMask(Lb==1)=1;%Assuming blank component 1
imshow(SubMask)

%--Canny edge detection
Edge=edge(Grey,'Canny'); %default parameters
%--Generate label matrix on Canny edge
Edge2=Edge;
Edge2(SubMask==1)=0;
CC=bwconncomp(Edge2);
L=labelmatrix(CC);
imshow(label2rgb(L))
%--Label
Stats = regionprops(CC, 'Area');
idx = find([Stats.Area] <= 50);
SM2 = ismember(labelmatrix(CC), idx);
SubMask(SM2==1)=1;

Edge2(SubMask==1)=0;

function [Edge3]=CoordSearch(Edge2)
    CC=bwconncomp(Edge2);
    L=labelmatrix(CC);

    StatsA = regionprops(CC, 'PixelList','Extrema');
    %Create list of all coords for searching
    AllCoords=[];BR=[];LT=[];
    for i=1:CC.NumObjects
        AllCoords=[AllCoords;StatsA(i).PixelList];
    end
    %Create list of all coords for searching
    %Generate search point: mid way, left side
    x=0;
    y=round(size(Edge2,1)/2);
    plot(x,y,'ro')
    plot([0,size(Edge2,2)],[y,y],'r')
    %Find the closest coordinate to the search point
    k=dsearchn(AllCoords,[x,y]);
    plot(AllCoords(k,1),AllCoords(k,2),'.r')

```

```
    %Run across the centre line, taking the first point above, until hitting an
    %edge on the search line. Then search down, finding each point to the
    %right.
    Edge3=zeros(size(Edge2));
    %Search right
    c = AllCoords(k,1);
    while Edge2(y,c)==0&&c<size(Edge2,2)
        r=y;
        while Edge2(r,c)==0 && r>1
            r=r-1;
        end
        Edge3(r,c)=1;
        c=c+1;
    end
    %Search down
    x=round(size(Edge2,2)/2);
     plot([x,x],[y,2*y],'r')
    while Edge2(r,x)==0 && r<size(Edge2,1)
       c=x;
       while Edge2(r,c)==0 && c<size(Edge2,2)
           c=c+1;
       end
       Edge3(r,c)=1;
       r=r+1;
    end

function [Coord]=ExtractCoord(Edge3)
    CC=bwconncomp(Edge3);
    L=labelmatrix(CC);
    %imshow(label2rgb(L))
    %--Label
    Stats = regionprops(CC, 'Area','PixelList');
    idx = find([Stats.Area] > 2);

    Coord=[];
    for i=idx
        Coord=[Coord;Stats(i).PixelList];
    end



function ET_MinL_Callback(hObject, eventdata, handles)
%---------------------------------------------------------------------
%---------------------------------------------------------------------

% --- Executes during object creation, after setting all properties.
```

```
function ET_MinL_CreateFcn(hObject, eventdata, handles)
%--------------------------------------------------------------------
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%--------------------------------------------------------------------



function ET_DropLabel_Callback(hObject, eventdata, handles)
% hObject    handle to ET_DropLabel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_DropLabel as text
%        str2double(get(hObject,'String')) returns contents of ET_DropLabel as a double


% --- Executes during object creation, after setting all properties.
function ET_DropLabel_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_DropLabel (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in PB_Res.
function PB_Res_Callback(hObject, eventdata, handles)
%--------------------------------------------------------------------
%-- Show the edge resolution
%--------------------------------------------------------------------
%--------------------------------------------------------------------


% --- Executes on button press in CB_SBE.
function CB_SBE_Callback(hObject, eventdata, handles)
%--------------------------------------------------------------------
%-- Show the edge resolution
```

```
%———————————————————————————————————————————————————
handles.SBD=get(hObject,'Value');
disp('toggle: searching from both directions');

guidata(hObject, handles);

fprintf('\n Done.')
%———————————————————————————————————————————————————



function ET_Fr_Callback(hObject, eventdata, handles)
%———————————————————————————————————————————————————
if isfield(handles,'ImName')==0
    GUI_Hmain = getappdata(0,'GUI_Hmain');
    Tag = getappdata(GUI_Hmain, 'Tag');
    load(fullfile(handles.Path,sprintf('%sThI',Tag)));

end

disp('\n Done.')
%———————————————————————————————————————————————————


% --- Executes during object creation, after setting all properties.
function ET_Fr_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_Fr (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in PB_Ed.
function PB_Ed_Callback(hObject, eventdata, handles)
% hObject    handle to PB_Ed (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in CB_WMask.
```

```
function CB_WMask_Callback(hObject, eventdata, handles)
% hObject    handle to CB_WMask (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of CB_WMask


% --- Executes on button press in CB_BMask.
function CB_BMask_Callback(hObject, eventdata, handles)
% hObject    handle to CB_BMask (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of CB_BMask



function ET_SEsize_Callback(hObject, eventdata, handles)
MorphProps=handles.MorphProps;
MorphProps.Size=str2double(get(hObject,'String'));
handles.MorphProps=MorphProps;
guidata(hObject, handles);


% --- Executes during object creation, after setting all properties.
function ET_SEsize_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_SEsize (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function ET_SEtype_Callback(hObject, eventdata, handles)
MorphProps=handles.MorphProps;
MorphProps.Type=str2double(get(hObject,'String'));
handles.MorphProps=MorphProps;
guidata(hObject, handles);guidata(hObject, handles);

% --- Executes during object creation, after setting all properties.
function ET_SEtype_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to ET_SEtype (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in PB_Flood.
function PB_Flood_Callback(hObject, eventdata, handles)
%--------------------------------------------------------------------------
%-- User chooses points (mouse click) to start the flood fill (imfill)
%--------------------------------------------------------------------------
load(fullfile(handles.Path,'ThreshTemp1'));

axes(handles.Ax_BW), imshow(BW3);
[BW5,FilLoc] = imfill(BW3);
imshow(BW5);
handles.FilLoc=FilLoc;
guidata(hObject, handles);
%--------------------------------------------------------------------------


% --- Executes on button press in PB_FloodOK.
function PB_FloodOK_Callback(hObject, eventdata, handles)
%--------------------------------------------------------------------------
MorphProps=handles.MorphProps;
MorphProps.FilLoc=handles.FilLoc;
handles.MorphProps=MorphProps;
guidata(hObject, handles);
%Thresholds.Thr=handles.Thr;
Thresholds.BWadj=handles.BWadj;
Thresholds.CanL=handles.CANadjL;
Thresholds.CanH=handles.CANadjH;
save('MorphProps','MorphProps','Thresholds');
disp('Done.')
%--------------------------------------------------------------------------


% --- Executes on button press in PB_CropChange.
function PB_CropChange_Callback(hObject, eventdata, handles)
% hObject    handle to PB_CropChange (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in PB_SelChange.
function PB_SelChange_Callback(hObject, eventdata, handles)
%-----------------------------------------------------------------------
%--- Show fitting areas on main axis for editing
axes(handles.Ax_Main);
    Cnr=handles.HlmCnr(1,:);
    Cnr2=[Cnr(1),Cnr(2),Cnr(3)-Cnr(1), Cnr(4)-Cnr(2)];
hR = imrect(gca, Cnr2);
    Cnr=handles.HlmCnr(2,:);
    Cnr2=[Cnr(1),Cnr(2),Cnr(3)-Cnr(1), Cnr(4)-Cnr(2)];
hL = imrect(gca, Cnr2);
    Cnr=handles.SphCnr(1,:);
    Cnr2=[Cnr(1),Cnr(2),Cnr(3)-Cnr(1), Cnr(4)-Cnr(2)];
sR = imrect(gca, Cnr2);
    Cnr=handles.SphCnr(2,:);
    Cnr2=[Cnr(1),Cnr(2),Cnr(3)-Cnr(1), Cnr(4)-Cnr(2)];
sL = imrect(gca, Cnr2);

waitforme=msgbox('Adjust fitting areas by dragging the boxes, then close this dialogue box to continue.');
uiwait(waitforme);
pos = getPosition(hR);
    Cnr=[pos(1),pos(2),pos(1)+pos(3),pos(2)+pos(4)];
    handles.HlmCnr(1,:)=Cnr;
    delete(hR);
pos = getPosition(hL);
    Cnr=[pos(1),pos(2),pos(1)+pos(3),pos(2)+pos(4)];
    handles.HlmCnr(2,:)=Cnr;
    delete(hL);
pos = getPosition(sR);
    Cnr=[pos(1),pos(2),pos(1)+pos(3),pos(2)+pos(4)];
    handles.SphCnr(1,:)=Cnr;
    delete(sR);
pos = getPosition(sL);
    Cnr=[pos(1),pos(2),pos(1)+pos(3),pos(2)+pos(4)];
    handles.SphCnr(2,:)=Cnr;
    delete(sL);

guidata(hObject, handles);
disp('Done.')
%-----------------------------------------------------------------------
```

```

% --- Executes on button press in PB_Load.
function PB_Load_Callback(hObject, eventdata, handles)
%------------------------------------------------------------
%--- Load a settings file.
    GUI_Hmain = getappdata(0,'GUI_Hmain');
    Tag = getappdata(GUI_Hmain, 'Tag');
    load(fullfile(handles.Path,sprintf('%sThI',Tag)));

    [ResultFile,Path]=uigetfile('.mat','multiselect','off')
    FullFileName=fullfile(Path,ResultFile);
    load(FullFileName);

    sz=size(size(Grey));
    if sz(2)==3
        Grey=rgb2gray(Grey);
    end
    handles.Grey=Grey;
    handles.SphCnr=SphCnr;
    handles.HlmCnr=HlmCnr;
    handles.ST=ST;
    handles.End=End;
    handles.Intv=Intv;
    handles.Prefix=Prefix;
    handles.Extn=Extn;
    handles.Crop=Crop;

%.. Update thresholds
    handles.BWadj=BWadj;
        set(handles.T_Adj,'string',sprintf('%0.2d',BWadj));
        set(handles.SL_BW,'value',BWadj);
    handles.CANadjL=CANadjL;
        set(handles.T_CL,'string',sprintf('%0.2d',CANadjL));
        set(handles.SL_CanL,'value',BWadj);
    handles.CANadjH=CANadjH;
        set(handles.T_CH,'string',sprintf('%0.2d',CANadjH));
        set(handles.SL_CanH,'value',BWadj);
    handles.wd = wd;
        set(handles.ET_wd,'string',sprintf('%0.2d',wd));
    handles.SBD = SBD; %default
        if SBD ==1
            set(handles.CB_SBE,'value',1)
        end
%.. Default settings for mask
    set(handles.ET_SEsize,'string',sprintf('%0.1d',MorphProps.Size));
    set(handles.ET_SEtype,'string',sprintf('%s',MorphProps.Type));
```

```
    handles.MorphProps=MorphProps;

guidata(hObject, handles);
disp('Done.')
%--------------------------------------------------------------------



function ET_NL_Callback(hObject, eventdata, handles)
%--------------------------------------------------------------------
handles.NL=str2double(get(hObject,'String'));
guidata(hObject, handles);
disp('Done.')
%--------------------------------------------------------------------

% --- Executes during object creation, after setting all properties.
function ET_NL_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_NL (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

B.4 AngleGUI (automated angle adjustment)

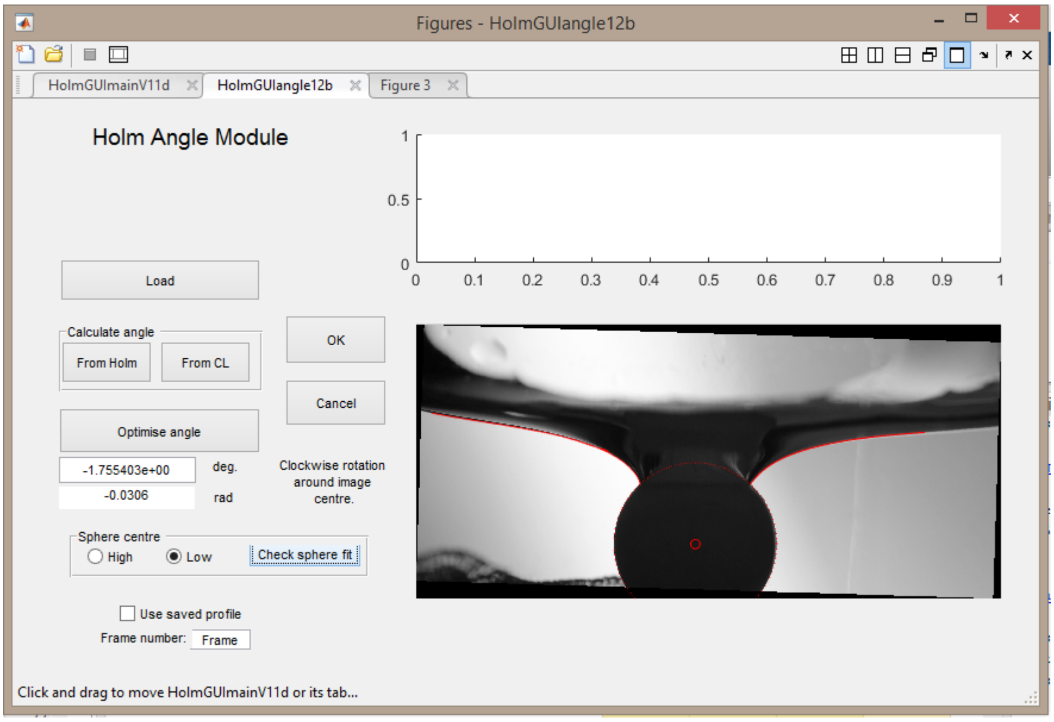

Figure B.3: Angle adjustment GUI.

Listing B.4: HolmGUIangle12b.m

```
function varargout = HolmGUIangle12b(varargin)
% HOLMGUIANGLE11 MATLAB code for HolmGUIangle11.fig
%      HOLMGUIANGLE11, by itself, creates a new HOLMGUIANGLE11 or raises the existing
%      singleton*.
%
%      H = HOLMGUIANGLE11 returns the handle to a new HOLMGUIANGLE11 or the handle to
%      the existing singleton*.
%
%      HOLMGUIANGLE11('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in HOLMGUIANGLE11.M with the given input arguments.
%
%      HOLMGUIANGLE11('Property','Value',...) creates a new HOLMGUIANGLE11 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before HolmGUIangle11_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to HolmGUIangle11_OpeningFcn via varargin.
%
```

```
%       *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%       instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help HolmGUIangle11

% Last Modified by GUIDE v2.5 04-Oct-2016 09:52:46

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @HolmGUIangle11_OpeningFcn, ...
                   'gui_OutputFcn',  @HolmGUIangle11_OutputFcn, ...
                   'gui_LayoutFcn',  []  , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before HolmGUIangle11 is made visible.
function HolmGUIangle11_OpeningFcn(hObject, eventdata, handles, varargin)
%-------------------------------------------------------------------
    GUI_Angle = gcf;
    setappdata(0,'GUI_Angle',GUI_Angle);
    GUI_Hmain=getappdata(0,'GUI_Hmain');
    handles.Path=getappdata(GUI_Hmain,'Path');
% Choose default command line output for HolmGUIangle11
    handles.output = hObject;

%-- Check for existing information stored with the main GUI
    if isappdata(GUI_Hmain,'Alpha')==1
        handles.Theta=getappdata(GUI_Hmain,'Theta');
        handles.Alpha=getappdata(GUI_Hmain,'Alpha');
        set(handles.ET_Theta,'String',sprintf('%0.4f',handles.Theta));
        set(handles.ST_Alpha,'String',sprintf('%0.4f',handles.Alpha));
        fprintf('Angle data exists in main GUI. Loading to angle GUI.\n');
```

```
    else
        fprintf('No data exists yet.\n');
        handles.Theta=0;
        handles.Alpha=0;
    end

    % Update handles structure
    guidata(hObject, handles);
%--------------------------------------------------------------------


% --- Outputs from this function are returned to the command line.
function varargout = HolmGUIangle11_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in PB_OK.
function PB_OK_Callback(hObject, eventdata, handles)
%--------------------------------------------------------------------
%-- SAVE DATA AND RETURN TO MAIN GUI
    GUI_Hmain = getappdata(0,'GUI_Hmain');

    Tag=getappdata(GUI_Hmain,'Tag');
    file=fullfile(handles.Path,sprintf('%s-Data.mat',Tag));
    load(file)

    Alpha=handles.Alpha
    Theta=handles.Theta

    Data.Alpha=Alpha;
    Data.Theta=Theta;
    Data.HL=get(get(handles.UI_Sph,'SelectedObject'),'String');
    setappdata(GUI_Hmain,'Alpha',Alpha);
    setappdata(GUI_Hmain,'Theta',Theta);
    setappdata(GUI_Hmain,'HL',Data.HL);

    save(file,'Data');

    fprintf(1,'Angle data saved. Returning to main GUI.\n');
%.. Call handles
```

```
    GUI_Angle = getappdata(0,'GUI_Angle');
%.. Close GUI
    close(GUI_Angle);
%-------------------------------------------------------------------------

% --- Executes on button press in PB_Cancel.
function PB_Cancel_Callback(hObject, eventdata, handles)
%-------------------------------------------------------------------------
%-- CLOSE GUI AND RETURN TO MAIN GUI
    fprintf(1,'Returning to main GUI. Parameters not saved.');
%.. Call handles
    GUI_Angle = getappdata(0,'GUI_Angle');
%.. Close GUI
    close(GUI_Angle);
%-------------------------------------------------------------------------


% --- Executes on button press in PB_Optimise.
function PB_Optimise_Callback(hObject, eventdata, handles)
%-------------------------------------------------------------------------
%-- RUN HOLM ANGLE MODULE TO CALCULATE THE ANGLE ADJUSTMENT
disp('Attempting to calculate image adjustment angle')
%-------------------------------------------------------------------------
%-- GEOMETRIC CALCULATION TO ADJUST FOR IMAGE (HORIZONTAL) TILT
%.. If the holm is symmetrical, the inflection point of the two sides (the
%narrowest point) should be on the same horizontal line.
%-------------------------------------------------------------------------
Tol=15;
%-- Handles
    GUI_Hmain = getappdata(0,'GUI_Hmain');
    handles.hImMask = getappdata(GUI_Hmain,'hImMask');
    handles.hTri = getappdata(GUI_Hmain,'hTri');
    handles.hTriSphere = getappdata(GUI_Hmain,'hTriSphere');
    handles.hRotateCoords = getappdata(GUI_Hmain,'hRotateCoords');
%-- Load parameters
    Tag=getappdata(GUI_Hmain,'Tag');
    handles.Path=getappdata(GUI_Hmain,'Path');
    file=fullfile(handles.Path,sprintf('%s-Data.mat',Tag));
    load(file)
    load(fullfile(handles.Path,'ThreshTemp2'))
    load(fullfile(handles.Path,'ThreshTemp1'))
    if strcmp('Movie',Data.ImType)==1
        DropMovie = VideoReader(Data.ImName);
        Grey = read(DropMovie, Data.ST);
    else
        Grey=imread(Data.ImName);
```

```
    end

    SphCnr=Data.SphCnr;
    HlmCnr=Data.HlmCnr;
    Cnr=Data.Crop;
    %---------------------------------------------------------------
    % GREY IMAGE
    %---------------------------------------------------------------
    sz=size(size(Grey));
    if sz(2)>2,
        Grey=rgb2gray(Grey);
    end

    Grey=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));
    axes(handles.axes1), hold off, imshow(Grey), hold on

    %---------------------------------------------------------------
    % SPHERE EDGE DETECTION
    %---------------------------------------------------------------
    %--Edge detection, Left & Right sides
    [SphereCoordL,SphereCoordR]=handles.hImMask(SphCnr(2,:),SphCnr(1,:),Grey,
        ThreshC,Thr,0,wd,0,WIM,[],NL,BMask);%'0' for flag - needed in ThreshGUI
        only.
    %---------------------------------------------------------------
    % HOLM EDGE DETECTION
    %---------------------------------------------------------------
    [HolmCrdL,HolmCrdR]=handles.hImMask(HlmCnr(2,:),HlmCnr(1,:),Grey,ThreshC,Thr
        ,0,wd,SBD,WIM,[],NL,BMask);
    HolmCoordL=sortrows(HolmCrdL,2);
    HolmCoordR=sortrows(HolmCrdR,2);


    %---------------------------------------------------------------
    % SPHERE FIT
    %---------------------------------------------------------------
    SphereCoord=[transpose(SphereCoordR),transpose(SphereCoordL)];%One layer - fit
         together

    %--Fit circle
    [SphereOpt,fvalS]=handles.hTriSphere(SphereCoordR(1,1)-SphereCoordL(1,1),
        SphereCoord);%first argument is a VERY rough guess of the width
    %--Optimised sphere coordinates
    x0=SphereOpt(1);
    y0=SphereOpt(2);%adjusting x,y to main image
    plot(x0,y0,'+r')
    R=SphereOpt(3);
```

```
    if R==0; disp('Error fitting sphere profile (R==0)');end

    %--Theoretical points
    SphIde=zeros(3,int32(R/5));
    k=0;i=int16(0);
    for i=x0+R:-2:x0-R
        k=k+1;
        SphIde(1,k)=i;
        SphIde(2,k)=sqrt(R^2-(i-x0)^2)+y0;
        SphIde(3,k)=-sqrt(R^2-(i-x0)^2)+y0;
    end

            %-----------------------------------------------------------------
            % ROTATE COORDINATES
            %-----------------------------------------------------------------
                if Alpha==0
                    x0a=x0;
                    y0a=y0;
                    GreyA=Grey;
                else
                    [HolmCoordL,HolmCoordR,x0a,y0a]=RotateCoords(Alpha,x0,y0,R,
                        HolmCoordR,HolmCoordL,Grey,HL);
                    GreyA=imrotate(Grey,Thta);%for plotting only
                end

                HolmCoordLr=HolmCoordL;
                HolmCoordLr(1,:)=size(GreyA,2)-HolmCoordL(1,:);

                xL=size(GreyA,2)-x0a;

                figure(FDisp), imshow(GreyA), hold on
                axes(handles.axes1); hold off, imshow(GreyA), hold on
                plot(HolmCoordL(1,:),HolmCoordL(2,:),'.-r');
                plot(HolmCoordR(1,:),HolmCoordR(2,:),'.-r');
                plot(x0a,y0a,'r+');

                %--Theoretical points
                SphIde=zeros(3,int32(R/5));
                m=0;
                for n=x0a+R:-2:x0a-R
                    m=m+1;
                    SphIde(1,m)=n;
                    SphIde(2,m)=sqrt(R^2-(n-x0a)^2)+y0a;
                    SphIde(3,m)=-sqrt(R^2-(n-x0a)^2)+y0a;
                end
                %--plot ideal circle for comparison
```

```
                    plot(SphIde(1,:),SphIde(2,:),':r',SphIde(1,:),SphIde(3,:),':r')
figure(1),
                    plot(HolmCoordL(1,:),HolmCoordL(2,:),'.-r');
                    plot(HolmCoordR(1,:),HolmCoordR(2,:),'.-r');
                    plot(x0a,y0a,'r+');
                    plot(SphIde(1,:),SphIde(2,:),':r',SphIde(1,:),SphIde(3,:),':r')
drawnow;
               %---------------------------------------------------------------
               % Analyse RIGHT
               %---------------------------------------------------------------
                    [GammaR,Shape,fval,OptStore]=Holm_MAIN_3('Right',ki,ImName,PrintYN
                         ,x0a,y0a,R,HolmCoordR,GreyA,fullfile(handles.Path,folder),fig,
                         FDisp,Data);
                         %Output::Gamma=[Gamma,GammaAveF];shape=aBF=[aBest,aAveF];fval
                              =[fvalH,fvalS,fHave];
                    fprintf(1,'Frame %d, Right - complete\n',ki);
               %---------------------------------------------------------------
               % Analyse LEFT
               %---------------------------------------------------------------
                    %Left -->code will flip image (fliplr)
                    figure(2); imshow(fliplr(GreyA)); hold on
                    plot(xL,y0a,'+r');%circle centre, flipped
                    plot(HolmCoordLr(1,:),HolmCoordLr(2,:),'r');

                    [GammaL,Shape,fval,OptStore]=Holm_MAIN_3('Left',ki,ImName,PrintYN,
                         xL,y0a,R,HolmCoordLr,fliplr(GreyA),fullfile(handles.Path,
                         folder),fig,FDisp,Data);
                    fprintf(1,'Frame %d, Left - complete\n',ki);
                    %abbreviated results file
                    fprintf(fileIDa,'% 3.6f,',Results(im,:));
                    fprintf(fileIDa,'\r\n');

                    axes(handles.axes2), hold on
                    plot(ki/FPS/60,GammaL(1),'+b', ki/FPS/60,GammaR(1),'+m','
                         markersize',3)%min
               %---------------------------------------------------------------
               % Analyse ENDS (if no error)






     Alpha = mean(Angles(:,1));
     Theta=Alpha*180/pi;

```

```
    save(fullfile(handles.Path,'AngleMat2'),'Alpha','Angles','Theta','Data')

    set(handles.ET_Theta,'String',sprintf('%0.3f',Theta));
    set(handles.ST_Alpha,'String',sprintf('%0.3f',Alpha));

%-- Optimise angle
    handles.Alpha=Alpha;
    handles.Theta=Theta;

    %.. Update handles structure
    guidata(hObject, handles);
%---------------------------------------------------------------


% --- Executes on button press in PB_Load.
function PB_Load_Callback(hObject, eventdata, handles)
%---------------------------------------------------------------
if exist(fullfile(handles.Path,'AngleMat2.mat'),'file')>1
    load(fullfile(handles.Path,'AngleMat2.mat'));
    fprintf('Loading file.\n')
    set(handles.ET_Theta,'String',sprintf('%0.3f',Theta));
    set(handles.ST_Alpha,'String',sprintf('%0.3f',Alpha));
    handles.Theta=Theta;
    handles.Alpha=Alpha;
else
    fprintf('No file to load.\n');
end
    guidata(hObject, handles);
%---------------------------------------------------------------

function ET_Theta_Callback(hObject, eventdata, handles)
%---------------------------------------------------------------
    handles.Theta=str2double(get(hObject,'String'));
    handles.Alpha=handles.Theta*pi/180;
    set(handles.ST_Alpha,'String',sprintf('%0.4f',handles.Alpha));
    guidata(hObject, handles);
%---------------------------------------------------------------

% --- Executes during object creation, after setting all properties.
function ET_Theta_CreateFcn(hObject, eventdata, handles)
%---------------------------------------------------------------
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%---------------------------------------------------------------
```

```
%--------------------------------------------------------------------------


% --- Executes on button press in PB_Calc.
function PB_Calc_Callback(hObject, eventdata, handles)
disp('Attempting to calculate image adjustment angle')
%--------------------------------------------------------------------------
%-- GEOMETRIC CALCULATION TO ADJUST FOR IMAGE (HORIZONTAL) TILT
%.. If the holm is symmetrical, the inflection point of the two sides (the
%narrowest point) should be on the same horizontal line.
%--------------------------------------------------------------------------
Tol=15;
%-- Handles
    GUI_Hmain = getappdata(0,'GUI_Hmain');
    handles.hImMask = getappdata(GUI_Hmain,'hImMask');
    handles.hTri = getappdata(GUI_Hmain,'hTri');
    handles.hTriSphere = getappdata(GUI_Hmain,'hTriSphere');
    handles.hRotateCoords = getappdata(GUI_Hmain,'hRotateCoords');
%-- Load parameters
    Tag=getappdata(GUI_Hmain,'Tag');
    handles.Path=getappdata(GUI_Hmain,'Path');
    file=fullfile(handles.Path,sprintf('%s-Data.mat',Tag));
    load(file)
    load(fullfile(handles.Path,'ThreshTemp2'))
    load(fullfile(handles.Path,'ThreshTemp1'))
    if strcmp('Movie',Data.ImType)==1
        DropMovie = VideoReader(Data.ImName);
        Grey = read(DropMovie, Data.ST);
    else
        Grey=imread(Data.ImName);
    end

    SphCnr=Data.SphCnr;
    HlmCnr=Data.HlmCnr;
    Cnr=Data.Crop;

    if get(handles.UE,'Value')==1
            %--------------------------------------------------------------------------
            % LOAD SAVED EDGES
            %--------------------------------------------------------------------------
                ki=str2double(get(handles.Frame,'String'));
                load(fullfile(handles.Path,'UserEdges',sprintf('UsrEdge%04i',ki)));
                mn=eval(genvarname(sprintf('UsrEdge%04i',ki)));

            %--------------------------------------------------------------------------
            % GREY IMAGE
```

```
            %------------------------------------------------------------
                ImFile=sprintf('%s%04i.%s',Data.Prefix,ki,Data.Ext);
                Grey = imread(ImFile);
                Cnr=Data.Crop;
                Grey=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));

            %------------------------------------------------------------
            % SPHERE EDGE
            %------------------------------------------------------------
                SphereCoordR=mn.SphCrdR;
                SphereCoordL=mn.SphCrdL;
            %------------------------------------------------------------
            % HOLM EDGE
            %------------------------------------------------------------
                HolmCoordL=sortrows(transpose(mn.HolmCrdL),2);
                HolmCoordR=sortrows(transpose(mn.HolmCrdR),2);

        else
            %------------------------------------------------------------
            % GREY IMAGE
            %------------------------------------------------------------
            sz=size(size(Grey));
            if sz(2)>2,
                Grey=rgb2gray(Grey);
            end

            Grey=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));
            axes(handles.axes1), hold off, imshow(Grey), hold on

            %------------------------------------------------------------
            % SPHERE EDGE DETECTION
            %------------------------------------------------------------
            %--Edge detection, Left & Right sides
            [SphereCoordL,SphereCoordR]=handles.hImMask(SphCnr(2,:),SphCnr(1,:),Grey,ThreshC,Thr,0,wd,0,WIM,[],NL,BMask);%'0' for flag - needed in ThreshGUI only.
            %------------------------------------------------------------
            % HOLM EDGE DETECTION
            %------------------------------------------------------------
            [HolmCrdL,HolmCrdR]=handles.hImMask(HlmCnr(2,:),HlmCnr(1,:),Grey,ThreshC,Thr,0,wd,SBD,WIM,[],NL,BMask);
             HolmCoordL=sortrows(HolmCrdL,2);
             HolmCoordR=sortrows(HolmCrdR,2);

    end

```

```
%-------------------------------------------------------------------
% SPHERE FIT
%-------------------------------------------------------------------
SphereCoord=[transpose(SphereCoordR),transpose(SphereCoordL)];%One layer - fit
    together

%--Fit circle
[SphereOpt,fvalS]=handles.hTriSphere(SphereCoordR(1,1)-SphereCoordL(1,1),
    SphereCoord);%first argument is a VERY rough guess of the width
%--Optimised sphere coordinates
x0=SphereOpt(1);
y0=SphereOpt(2);%adjusting x,y to main image
plot(x0,y0,'+r')
R=SphereOpt(3);
if R==0; disp('Error fitting sphere profile (R==0)');end

%--Theoretical points
SphIde=zeros(3,int32(R/5));
k=0;i=int16(0);
for i=x0+R:-2:x0-R
    k=k+1;
    SphIde(1,k)=i;
    SphIde(2,k)=sqrt(R^2-(i-x0)^2)+y0;
    SphIde(3,k)=-sqrt(R^2-(i-x0)^2)+y0;
end

%-------------------------------------------------------------------
% TRIM HOLM
%-------------------------------------------------------------------
%Ignore "holm" below sphere max.
i=0;
while i<length(HolmCoordL)&&HolmCoordL(i+1,2)<y0
    i=i+1;
end
HolmCrdL=HolmCoordL(1:i,:);
i=0;
while i<length(HolmCoordR)&&HolmCoordR(i+1,2)<y0
    i=i+1;
end
HolmCrdR=HolmCoordR(1:i,:);

%-------------------------------------------------------------------
%-- MAP L-R
%-------------------------------------------------------------------
figure(3); imshow(Grey), hold on
%-- Define query points
```

```
L=min(length(HolmCrdL),length(HolmCrdR));
X=HolmCrdL;
q=HolmCrdR;
plot(X(:,1),X(:,2),'ro'), plot(q(:,1),q(:,2),'bo')

%-- Delaunay triangle
dt = DelaunayTri(X);

%-- Find the nearest neighbours
[xi,D] = nearestNeighbor(dt, q);
xnn = X(xi,:);
plot([xnn(:,1) q(:,1)]',[xnn(:,2) q(:,2)]','-b');

%-- Find the shortest eclidian distance
[Dmin,ind]=min(D);
plot([xnn(ind,1) q(ind,1)]',[xnn(ind,2) q(ind,2)]','g'); %shortest distance
    between discrete points.

%-- Fit polynomial +- ni
ni=100;
if ind<ni
    Lb=1;
else
    Lb=ind-ni;
end
if ind>length(q)-ni
    Ub=length(q);
else
    Ub=ind+ni;
end
polyR = polyfit(q(Lb:Ub,2),q(Lb:Ub,1),3);
curveR=polyval(polyR,q(Lb,2):q(Ub,2));
plot(curveR,q(Lb,2):q(Ub,2),'g');%polynomial fit to right side

%-- New q
q=transpose([curveR;q(Lb,2):q(Ub,2)]);
Cd=[Lb,Ub];

%--Left
if xi(ind)<ni
    Lb=1;
else
    Lb=xi(ind)-ni;
end
if xi(ind)>length(X)-ni
    Ub=length(X);
```

```
else
    Ub=xi(ind)+ni;
end
polyL = polyfit(X(Lb:Ub,2),X(Lb:Ub,1),3);
curveL=polyval(polyL,X(Lb,2):X(Ub,2));
plot(curveL,X(Lb,2):X(Ub,2),'g');%Poly nomial fit to left side.

%--New X
X=transpose([curveL;X(Lb,2):X(Ub,2)]);

%-- Pick new best point using polynomials
plot(X(:,1),X(:,2),'.m'), plot(q(:,1),q(:,2),'.c')

%-- Delaunay triangle
dt = DelaunayTri(X);

%-- Find the nearest neighbours
[xi,D] = nearestNeighbor(dt, q);
xnn = X(xi,:);

%-- Find the shortest eclidian distance
[Dmin,ind]=min(D);
plot([xnn(ind,1) q(ind,1)]',[xnn(ind,2) q(ind,2)]','y','linewidth',2); %shorted
    distance between poly


%-- Determine the end points for the shortest distance
L=xnn(ind,:);
R=q(ind,:);


%-- Adjust image
dx=diff([L(1),R(1)]);
dy=diff([L(2),R(2)]);
Alpha=atan2(dy,dx);

im=size(Grey)/2; plot(im(2),im(1),'go');
% if R(2)>im(1);
%     Thta=Alpha*180/pi;
% else
%     Thta=Alpha*180/pi;
%     Alpha=-Alpha;
% end
Thta=Alpha*180/pi;
Alpha=-Alpha;
HL=get(get(handles.UI_Sph,'SelectedObject'),'String');
```

```
axes(handles.axes1); hold on
GreyA=imrotate(Grey,Thta);%for plotting only
imshow(GreyA);
disp(Thta);

%-- Rotate coordinates from original image
[HolmCoordL,HolmCoordR,x0a,y0a]=handles.hRotateCoords(Alpha,x0,y0,R,transpose(
    HolmCrdR),transpose(HolmCrdL),Grey,HL);

plot(HolmCoordL(1,:),HolmCoordL(2,:),'r')%rotates
plot(HolmCrdL(:,1),HolmCrdL(:,2),'g')%Original

%-- Check again for best point -> should be horizontal now
%L=X=transpose(HolmCoordL(:,xi(ind)));
%R=q=transpose(HolmCoordR(:,ind));
%plot(X(:,1),X(:,2),'ro'), plot(q(:,1),q(:,2),'bo')
% X=transpose(HolmCoordL);
% q=transpose(HolmCoordR);%after rotation
%
% %-- Fit polynomial +- 20
% ni=100;
% if ind<ni
%     Lb=1;
% else
%     Lb=ind-ni;
% end
% if ind>length(q)-ni
%     Ub=min(length(q),length(X));
% else
%     Ub=ind+ni;
% end
% polyR = polyfit(q(Lb:Ub,2),q(Lb:Ub,1),3);
% curveR=polyval(polyR,q(Lb,2):q(Ub,2));
% plot(curveR,q(Lb,2):q(Lb,2),'g');
%
% %-- New q
% q=transpose([curveR;q(Lb,2):q(Ub,2)]);
%
% %--Left
% if xi(ind)<ni
%     Lb=1;
% else
%     Lb=xi(ind)-ni;
% end
% if xi(ind)>length(X)-ni
%     Ub=length(X);
```

```
% else
%      Ub=xi(ind)+ni;
% end
% polyL = polyfit(X(Lb:Ub,2),X(Lb:Ub,1),3);
% curveL=polyval(polyL,X(Lb,2):X(Ub,2));
% plot(curveL,X(Lb,2):X(Ub,2),'g');
%
% %—New X
% X=transpose([curveL;X(Lb,2):X(Ub,2)]);
%
% %— Pick new best point using polynomials
% plot(X(:,1),X(:,2),'m'), plot(q(:,1),q(:,2),'c')
%
% %— Delaunay triangle
% dt = DelaunayTri(X);
%
% %— Find the nearest neighbours
% [xi,D] = nearestNeighbor(dt, q);
% xnn = X(xi,:);
%
% %— Find the shortest eclidian distance
% [Dmin,ind]=min(D);
% plot([xnn(ind,1) q(ind,1)]',[xnn(ind,2) q(ind,2)]','y','linewidth',2);
%
% %— Determine the end points for the shortest distance
% L=xnn(ind,:);
% R=q(ind,:);

%-- Update Gui and handles
set(handles.ET_Theta,'String',sprintf('%s',Thta));
set(handles.ST_Alpha,'String',sprintf('%s',Alpha));

handles.Theta=Thta;
handles.Alpha=Alpha;
guidata(hObject, handles);
%--------------------------------------------------------------------------%


% --- Executes on button press in SphereFit.
function SphereFit_Callback(hObject, eventdata, handles)
%--------------------------------------------------------------------------%
%-- Show the edge detection and sphere fit for the new angle. Check whether
%the angle adjustment is affected the proper placement of the centre point.
%--------------------------------------------------------------------------%
%-- Handles
    GUI_Hmain = getappdata(0,'GUI_Hmain');
```

```
    handles.hImMask = getappdata(GUI_Hmain,'hImMask');
    handles.hTri = getappdata(GUI_Hmain,'hTri');
    handles.hTriSphere = getappdata(GUI_Hmain,'hTriSphere');
    handles.hRotateCoords = getappdata(GUI_Hmain,'hRotateCoords');

%.. Load properties file
    Tag=getappdata(GUI_Hmain,'Tag');
    load(fullfile(handles.Path,(sprintf('%s-Data.mat',Tag))));

%.. Load edge detection file
if exist(fullfile(handles.Path,'ThreshTemp2.mat'),'file')==0
    msgbox('Please confirm the edge detection. Routing to main GUI.')
    return
else
    load(fullfile(handles.Path,'ThreshTemp2'))
    %Loads wd, BWadj, CANadjH, CANadjL
end
fig=handles.axes1;
FDisp=1;
Tol=15;

Thta=handles.Theta;
Alpha=-handles.Alpha;

if strcmp('Movie',Data.ImType)==1
    DropMovie = VideoReader(Data.ImName);
    Grey = read(DropMovie, Data.ST);
else
    Grey=imread(Data.ImName);
end

SphCnr=Data.SphCnr;
HlmCnr=Data.HlmCnr;
Cnr=Data.Crop;
%----------------------------------------------------------------
% GREY IMAGE
%----------------------------------------------------------------
sz=size(size(Grey));
if sz(2)>2,
    Grey=rgb2gray(Grey);
end

Grey=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));
axes(handles.axes1), hold off, imshow(Grey), hold on
%----------------------------------------------------------------
% SPHERE EDGE DETECTION AND FIT
```

```
%----------------------------------------------------------------
%--Edge detection, Left & Right sides
[SphereCoordL,SphereCoordR]=handles.hImMask(SphCnr(2,:),SphCnr(1,:),Grey,ThreshC,
    Thr,0,wd,0,WIM,[],NL,BMask);%'0' for flag - needed in ThreshGUI only.
SphereCoord=[transpose(SphereCoordR),transpose(SphereCoordL)];%One layer - fit
    together

%--Fit circle
[SphereOpt,fvalS]=handles.hTriSphere((SphereCoordR(1,1)-SphereCoordL(1,1))/2,
    SphereCoord);%first argument is a VERY rough guess of the width
%--Optimised sphere coordinates
x0=SphereOpt(1);
y0=SphereOpt(2);%adjusting x,y to main image
plot(x0,y0,'+r')
R=SphereOpt(3);
if R==0; disp('Error fitting sphere profile (R==0)');end

%--Theoretical points
SphIde=zeros(3,int32(R/5));
k=0;i=int16(0);
for i=x0+R:-2:x0-R
    k=k+1;
    SphIde(1,k)=i;
    SphIde(2,k)=sqrt(R^2-(i-x0)^2)+y0;
    SphIde(3,k)=-sqrt(R^2-(i-x0)^2)+y0;
end

%----------------------------------------------------------------
% HOLM EDGE DETECTION
%----------------------------------------------------------------
[HolmCrdL,HolmCrdR]=handles.hImMask(HlmCnr(2,:),HlmCnr(1,:),Grey,ThreshC,Thr,0,wd,
    SBD,WIM,[],NL,BMask);
HolmCoordL=fliplr(transpose(HolmCrdL)); HolmCoordR=fliplr(transpose(HolmCrdR));

%----------------------------------------------------------------
% ROTATE COORDINATES
%----------------------------------------------------------------
if Alpha==0
    x0a=x0;
    y0a=y0;
    GreyA=Grey;
else
    HL=get(get(handles.UI_Sph,'SelectedObject'),'String');
    [HolmCoordL,HolmCoordR,x0a,y0a]=handles.hRotateCoords(Alpha,x0,y0,R,HolmCoordR
        ,HolmCoordL,Grey,HL);
    GreyA=imrotate(Grey,Thta);%for plotting only
```

```
end

axes(handles.axes1); hold off, imshow(GreyA), hold on
plot(HolmCoordL(1,:),HolmCoordL(2,:),'r');
plot(HolmCoordR(1,:),HolmCoordR(2,:),'r');
plot(x0a,y0a,'ro');

%--Theoretical points
SphIde=zeros(3,int32(R/5));
m=0;
for n=x0a+R:-2:x0a-R
    m=m+1;
    SphIde(1,m)=n;
    SphIde(2,m)=sqrt(R^2-(n-x0a)^2)+y0a;
    SphIde(3,m)=-sqrt(R^2-(n-x0a)^2)+y0a;
end
%--plot ideal circle for comparison
plot(x0a,y0a,'ro',SphIde(1,:),SphIde(2,:),':r',SphIde(1,:),SphIde(3,:),':r')
disp('Done. ');
%------------------------------------------------------------------------%


% --- Executes on button press in PB_CLcalc.
function PB_CLcalc_Callback(hObject, eventdata, handles)
%------------------------------------------------------------------------%
%-- Calculate the adjustment angle based on the contact line.
%------------------------------------------------------------------------%
disp('Attempting to calculate the rotation angle...')
%-- Handles
    GUI_Hmain = getappdata(0,'GUI_Hmain');
    handles.hImMask = getappdata(GUI_Hmain,'hImMask');
    handles.hTri = getappdata(GUI_Hmain,'hTri');
    handles.hTriSphere = getappdata(GUI_Hmain,'hTriSphere');
    handles.hRotateCoords = getappdata(GUI_Hmain,'hRotateCoords');

%.. Load properties file
    Tag=getappdata(GUI_Hmain,'Tag');
    load(fullfile(handles.Path,(sprintf('%s-Data.mat',Tag))));

%.. Load edge detection file
if exist(fullfile(handles.Path,'ThreshTemp2.mat'),'file')==0
    msgbox('No edge detection file exists. Routing to main GUI.')
    return
else
    load(fullfile(handles.Path,'ThreshTemp1'))
    load(fullfile(handles.Path,'ThreshTemp2'))
```

```
    %Loads wd, BWadj, CANadjH, CANadjL
end
fig=handles.axes1;
FDisp=1;
Tol=15;

SphCnr=Data.SphCnr;
HlmCnr=Data.HlmCnr;
Cnr=Data.Crop;

%-----------------------------------------------------------------
% GREY IMAGE
%-----------------------------------------------------------------
if strcmp('Movie',Data.ImType)==1
    DropMovie = VideoReader(Data.ImName);
    Grey = read(DropMovie, Data.ST);
else
    Grey=imread(Data.ImName);
end
sz=size(size(Grey));
if sz(2)>2,
    Grey=rgb2gray(Grey);
end

Grey=Grey(Cnr(2):Cnr(4),Cnr(1):Cnr(3));

%-----------------------------------------------------------------
% SPHERE EDGE DETECTION AND FIT
%-----------------------------------------------------------------
%--Edge detection, Left & Right sides
[SphereCoordL,SphereCoordR]=handles.hImMask(SphCnr(2,:),SphCnr(1,:),Grey,ThreshC,
    Thr,SvFlag,wd,SBD,WIM,Path,NL,BMask);%'0' for flag - needed in ThreshGUI only.
%CnrL,CnrR,Grey,ThreshC,Thr,SvFlag,wd,SBD,WIM,Path
SphereCoord=[transpose(SphereCoordR),transpose(SphereCoordL)];%One layer - fit
    together

%--Fit circle
[SphereOpt,fvalS]=handles.hTriSphere(SphereCoordR(1,1)-SphereCoordL(1,1),
    SphereCoord);%first argument is a VERY rough guess of the width
%--Optimised sphere coordinates
x0=SphereOpt(1);
y0=SphereOpt(2);%adjusting x,y to main image
plot(x0,y0,'+r')
R=SphereOpt(3);
if R==0; disp('Error fitting sphere profile (R==0)');end

```

```
%--Theoretical points
SphIde=zeros(3,int32(R/5));
k=0;i=int16(0);
for i=x0+R:-2:x0-R
    k=k+1;
    SphIde(1,k)=i;
    SphIde(2,k)=sqrt(R^2-(i-x0)^2)+y0;
    SphIde(3,k)=-sqrt(R^2-(i-x0)^2)+y0;
end
%-----------------------------------------------------------------------
% HOLM EDGE DETECTION
%-----------------------------------------------------------------------
[HolmCrdL,HolmCrdR]=handles.hImMask(HlmCnr(2,:),HlmCnr(1,:),Grey,ThreshC,Thr,
    SvFlag,wd,SBD,WIM,Path,NL,BMask);
HolmCoordL=fliplr(transpose(HolmCrdL)); HolmCoordR=fliplr(transpose(HolmCrdR));

%-----------------------------------------------------------------------
% UPDATE GUI FIGURE
%-----------------------------------------------------------------------
axes(handles.axes1), hold off, imshow(Grey), hold on
plot(x0,y0,'ro',SphIde(1,:),SphIde(2,:),':r',SphIde(1,:),SphIde(3,:),':r')
plot(HolmCoordL(1,:),HolmCoordL(2,:),'r',HolmCoordR(1,:),HolmCoordR(2,:),'r');

%-----------------------------------------------------------------------
% FIND CONTACT LINE
%-----------------------------------------------------------------------
%..Right
i=0;Diff=0;
while Diff<Tol
    i=i+1;
    X=HolmCoordR(1,i);
    Y=HolmCoordR(2,i);
    x=sqrt(abs(R^2-(Y-y0)^2))+x0;%(x-X)^2+(y-Y)^2=R^2
    dDiff=X-x;%not abs - do not want to start inside the circle
    if dDiff>3%Will not pick up negative values.
        Diff=Diff+dDiff;
    end
    plot([X,x],[Y,Y],'y')
end
CLr=[X,Y,i];
%..Left
i=0;Diff=0;
while Diff<Tol
    i=i+1;
    X=HolmCoordL(1,i);
    Y=HolmCoordL(2,i);
```

```
        x=x0-sqrt(abs(R^2-(Y-y0)^2));%(x-X)^2+(y-Y)^2=R^2
        dDiff=x-X;%not abs - do not want to start inside the circle
        if dDiff>3%Will not pick up negative values.
            Diff=Diff+dDiff;
        end
        plot([X,x],[Y,Y],'y')
end
CLl=[X,Y,i];
plot([CLl(1),CLr(1)],[CLl(2),CLr(2)],'g');
%-----------------------------------------------------------------------
% CALCULATE ROTATION ANGLE (CL should be horizontal)
%-----------------------------------------------------------------------
%-- Adjust image
dx=diff([CLl(1),CLr(1)]);
dy=diff([CLl(2),CLr(2)]);
Alpha=atan2(dy,dx);

im=size(Grey)/2; plot(im(2),im(1),'go');
Thta=Alpha*180/pi;
Alpha=-Alpha;
HL=get(get(handles.UI_Sph,'SelectedObject'),'String');
axes(handles.axes1); hold on
GreyA=imrotate(Grey,Thta);%for plotting only
imshow(GreyA);
disp(Thta);

%-- Rotate coordinates from original image
[HolmCoordL,HolmCoordR,x0a,y0a]=handles.hRotateCoords(Alpha,x0,y0,R,HolmCoordR,HolmCoordL,Grey,HL);

plot(HolmCoordL(1,:),HolmCoordL(2,:),'r')%rotates
plot(HolmCrdL(:,1),HolmCrdL(:,2),'g')%Original
plot([HolmCoordL(1,CLl(3)),HolmCoordR(1,CLr(3))],[HolmCoordL(2,CLl(3)),HolmCoordR(2,CLr(3))],'y');
%-----------------------------------------------------------------------
% SAVE DATA AND UPDATE FIGURES
%-----------------------------------------------------------------------
%-- Update Gui and handles
set(handles.ET_Theta,'String',sprintf('%0.3f',Thta));
set(handles.ST_Alpha,'String',sprintf('%0.3f',Alpha));

handles.Theta=Thta;
handles.Alpha=Alpha;
guidata(hObject, handles);
%-----------------------------------------------------------------------

```

```

% --- Executes on button press in UE.
function UE_Callback(hObject, eventdata, handles)
% hObject    handle to UE (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of UE



function Frame_Callback(hObject, eventdata, handles)
% hObject    handle to Frame (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of Frame as text
%        str2double(get(hObject,'String')) returns contents of Frame as a double


% --- Executes during object creation, after setting all properties.
function Frame_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Frame (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

B.5 Timeline (formatting of temperature files)

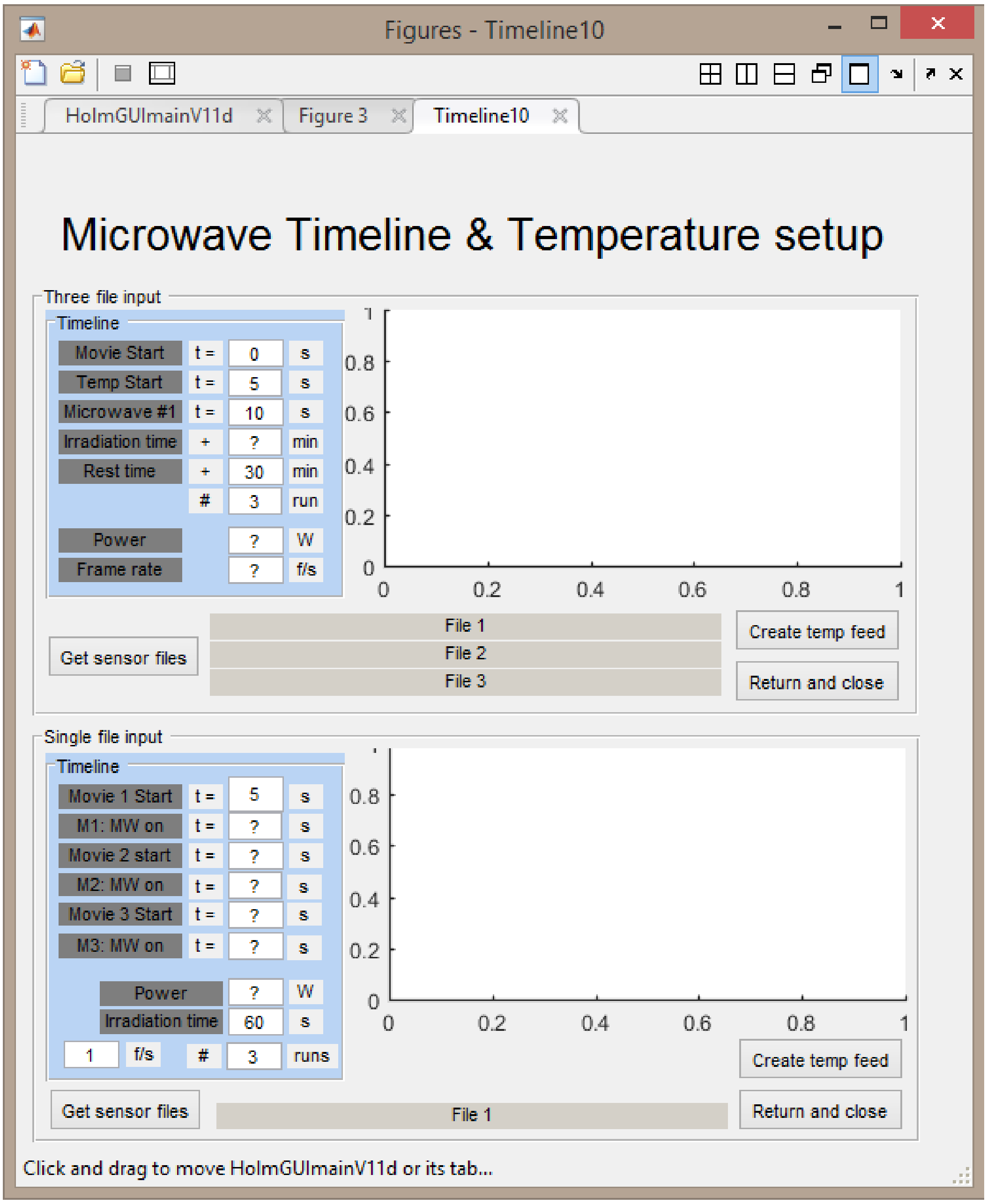

Figure B.4: Formatting for temperature files.

Listing B.5: Timeline10.m

```
function varargout = Timeline10(varargin)
% TIMELINE10 MATLAB code for Timeline10.fig
%      TIMELINE10, by itself, creates a new TIMELINE10 or raises the existing
%      singleton*.
```

```
%
%      H = TIMELINE10 returns the handle to a new TIMELINE10 or the handle to
%      the existing singleton*.
%
%      TIMELINE10('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in TIMELINE10.M with the given input arguments.
%
%      TIMELINE10('Property','Value',...) creates a new TIMELINE10 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Timeline10_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Timeline10_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Timeline10

% Last Modified by GUIDE v2.5 29-Jun-2016 19:16:02

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Timeline10_OpeningFcn, ...
                   'gui_OutputFcn',  @Timeline10_OutputFcn, ...
                   'gui_LayoutFcn',  []  , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Timeline10 is made visible.
function Timeline10_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
%----------------------------------------------------------------
```

```
% Choose default command line output for Timeline10
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
%------------------------------------------------------------
%..Set Handles data in desktop
setappdata(0,'GUI_hTimeline',gcf)
try
    GUI_Thresh = gcf;
    setappdata(0,'GUI_Thresh',GUI_Thresh);
    GUI_Hmain = getappdata(0,'GUI_hmain');
catch
    fprintf(1,'Not called from Main GUI\r\n');
end
%------------------------------------------------------------


% --- Outputs from this function are returned to the command line.
function varargout = Timeline10_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function ET_Mstart_Callback(hObject, eventdata, handles)
% hObject    handle to ET_Mstart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_Mstart as text
%        str2double(get(hObject,'String')) returns contents of ET_Mstart as a
    double


% --- Executes during object creation, after setting all properties.
function ET_Mstart_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_Mstart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_Tstart_Callback(hObject, eventdata, handles)
% hObject    handle to ET_Tstart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_Tstart as text
%        str2double(get(hObject,'String')) returns contents of ET_Tstart as a double


% --- Executes during object creation, after setting all properties.
function ET_Tstart_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_Tstart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_Micro1_Callback(hObject, eventdata, handles)
% hObject    handle to ET_Micro1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_Micro1 as text
%        str2double(get(hObject,'String')) returns contents of ET_Micro1 as a double


% --- Executes during object creation, after setting all properties.
function ET_Micro1_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to ET_Micro1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_irTime_Callback(hObject, eventdata, handles)
% hObject    handle to ET_irTime (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_irTime as text
%        str2double(get(hObject,'String')) returns contents of ET_irTime as a
    double


% --- Executes during object creation, after setting all properties.
function ET_irTime_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_irTime (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_Rest_Callback(hObject, eventdata, handles)
% hObject    handle to ET_Rest (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_Rest as text
%        str2double(get(hObject,'String')) returns contents of ET_Rest as a double

```

```
% --- Executes during object creation, after setting all properties.
function ET_Rest_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_Rest (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_runs_Callback(hObject, eventdata, handles)
% hObject    handle to ET_runs (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_runs as text
%        str2double(get(hObject,'String')) returns contents of ET_runs as a double


% --- Executes during object creation, after setting all properties.
function ET_runs_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_runs (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in PB_getFile.
function PB_getFile_Callback(hObject, eventdata, handles)
% hObject    handle to PB_getFile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%-----------------------------------------------------------------
%.. Ask user to select files
```

```
    [Files,Path]=uigetfile('.csv','multiselect','on')
    Files=cellstr(Files);%converts singleton into cell string
%.. Update filename tags
for n = 1:size(Files,2)
    switch n
        case 1
            set(handles.T_F1,'String',Files(n));
        case 2
            set(handles.T_F2,'String',Files(2));
        case 3
            set(handles.T_F3,'String',Files(3));
    end
end
%.. Update handles structure
    handles.Files=Files;
    handles.Path=Path;
    guidata(hObject, handles);

%.. Send update
    fprintf('Files loaded successfully.\r\n')
%------------------------------------------------------------------


function PB_create_Callback(hObject, eventdata, handles)
% hObject    handle to PB_create (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%------------------------------------------------------------------
%.. Load data (ignore header on 1st line), Modify matrix & save
MovStart = str2double(get(handles.ET_Mstart,'String'));
Tstart = str2double(get(handles.ET_Tstart,'String'));
adjR = Tstart - MovStart - 1; %missing time at the front of the temp file.
%.. Plot temp profiles
axes(handles.axes1);hold on

for n = 1:length(handles.Files)%the temperature probe provides information every
    second.
%------------------------------------------------------------------
% Temperature probe starts "adjR" seconds after the movie starts recording
% --> add "adjR" seconds worth of temperature to the start of the temp
% record.
%------------------------------------------------------------------
    switch n
        case 1
            TC = dlmread(fullfile(char(handles.Path),char(handles.Files(n))),','
                ,1,0);
```

```
                adj = [TC(1,1)*ones(adjR,1),TC(1,2)*ones(adjR,1)];
                TC = [adj; TC];
                plot(TC(:,1),TC(:,2),'b');
                TC=TC(:,2);
                save(fullfile(char(handles.Path),'TC1.mat'),'TC');
            case 2
                TC = dlmread(fullfile(char(handles.Path),char(handles.Files(n))),',',1,0);
                adj = [TC(1,1)*ones(adjR,1),TC(1,2)*ones(adjR,1)];
                TC = [adj; TC];
                plot(TC(:,1),TC(:,2),'g');
                TC=TC(:,2);
                save(fullfile(char(handles.Path),'TC2.mat'),'TC');
            case 3
                TC = dlmread(fullfile(char(handles.Path),char(handles.Files(n))),',',1,0);
                adj = [TC(1,1)*ones(adjR,1),TC(1,2)*ones(adjR,1)];
                TC = [adj; TC];
                plot(TC(:,1),TC(:,2),'r');
                TC=TC(:,2);
                save(fullfile(char(handles.Path),'TC3.mat'),'TC');
        end
end
legend('TC1','TC2','TC3');

%..Get GUI data
MicrStart=str2double(get(handles.ET_Micro1,'String'));
irTime=str2double(get(handles.ET_irTime,'String'));
Rest=str2double(get(handles.ET_Rest,'String'));
NumRuns=str2double(get(handles.ET_runs,'String'));
FrameRate=str2double(get(handles.ET_FrameRate,'String'));
Power=str2double(get(handles.ET_Power,'String'));

%..Save GUI data
MData=fullfile(handles.Path,'MData');
save(MData,'Power','MicrStart','Rest','NumRuns','irTime','FrameRate','MovStart','Tstart')
saveas(gcf,fullfile(char(handles.Path),'TimelineOUT'),'fig');

%..Alert
fprintf('Files have been saved.\r\n')
%------------------------------------------------------------------------



function ET_Power_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to ET_Power (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_Power as text
%        str2double(get(hObject,'String')) returns contents of ET_Power as a
    double


% --- Executes during object creation, after setting all properties.
function ET_Power_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_Power (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_FrameRate_Callback(hObject, eventdata, handles)
%--------------------------------------------------------------
     %GUI_Hmain = getappdata(0,'GUI_hmain');
     %setappdata(GUI_Hmain,'Fps',str2double(get(hObject,'String')));
%--------------------------------------------------------------

% --- Executes during object creation, after setting all properties.
function ET_FrameRate_CreateFcn(hObject, eventdata, handles)
%--------------------------------------------------------------
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
%--------------------------------------------------------------
try
    GUI_Hmain = getappdata(0,'GUI_hmain');
    Fps=getappdata(GUI_Hmain,'Fps');
    set(handles.ET_FrameRate,'String',sprintf('%d',Fps))
catch
    fprintf('no saved framerate\r\n');
end
%--------------------------------------------------------------
```

```matlab


% --- Executes on button press in PB_Return.
function PB_Return_Callback(hObject, eventdata, handles)
%-------------------------------------------------------------
%--- CLOSE AND RETURN TO MAIN GUI, OR CLOSE
    GUI_hTimeline=getappdata(0,'GUI_hTimeline');

%.. Check if temperature profiles exist, and warn.
    if exist(fullfile(handles.Path,'TC1.mat'),'file') == 0
        Msg=sprintf('No temperature profile has been exported. Close anyway?\r\n');
        fprintf(Msg)
        button=questdlg(Msg,'No');
        switch button
            case 'Yes'
                fprintf('Closing...\r\n');
                close(GUI_hTimeline)
            case 'No'
                fprintf('Returning...\r\n');
        end
%.. Close
    else
        fprintf('Closing...\r\n');
        close(GUI_hTimeline)
    end
%-------------------------------------------------------------


% --- Executes on button press in PB_Feed2.
function PB_Feed2_Callback(hObject, eventdata, handles)
%-------------------------------------------------------------
%.. Load data (ignore header on 1st line), Modify matrix & save
M1st = str2double(get(handles.ET_M1st,'String'));
M2st = str2double(get(handles.ET_M2st,'String'));
M3st = str2double(get(handles.ET_M3st,'String'));

MWOn1 = str2double(get(handles.ET_M1mwOn,'String'));
MWOn2 = str2double(get(handles.ET_M2mwOn,'String'));
MWOn3 = str2double(get(handles.ET_M3mwOn,'String'));

Ird = str2double(get(handles.ET_IrTime,'String'));
n = str2double(get(handles.ET_runs,'String'));

%.. Plot temp profiles
axes(handles.axes2); hold off
```

```matlab

%------------------------------------------------------------
% No tempt adjustment
%------------------------------------------------------------
TCa = dlmread(fullfile(char(handles.Path),char(handles.Files)),',',1,0);
plot(TCa(:,1),TCa(:,2),'b');

hold all

%Set up temp files & augment plot
for i=1:n
    switch i
        case 1
            if M1st <0
                a=0-M1st;
                TC=[TCa(1,2)*ones(a,1);TCa(1:M2st,2)];
            else
                TC=TCa(M1st+1:M2st,2);
            end
        save(fullfile(char(handles.Path),'TC1.mat'),'TC');
        plot([MWOn1,MWOn1,MWOn1+Ird,MWOn1+Ird],[0,100,100,0]);
        plot([M1st,M1st],[0,100],':');

        case 2
        TC=TCa(M2st+1:M3st,2);
        save(fullfile(char(handles.Path),'TC2.mat'),'TC');
        plot([MWOn2,MWOn2,MWOn2+Ird,MWOn2+Ird],[0,100,100,0]);
        plot([M2st,M2st],[0,100],':');

        case 3
        TC=TCa(M3st+1:end,2);
        save(fullfile(char(handles.Path),'TC3.mat'),'TC');
        plot([MWOn3,MWOn3,MWOn3+Ird,MWOn3+Ird],[0,100,100,0]);
        plot([M3st,M3st],[0,100],':');
    end
end

%legend('Temp','MW1','MW2','MW3');

%..Get GUI data
FrameRate=str2double(get(handles.ET_fps,'String'));
Power=str2double(get(handles.ET_P,'String'));
NumRuns = n;
%..Save GUI data
save('MData','Power','NumRuns','Ird','FrameRate','M1st','M2st','M3st')
saveas(gcf,fullfile(char(handles.Path),'TimelineOUT'),'fig');
```

```

%..Alert
 fprintf('Files have been saved.\r\n')
%------------------------------------------------------------------

% --- Executes on button press in PB_GetFiles2.
function PB_GetFiles2_Callback(hObject, eventdata, handles)
%.. Ask user to select files
    [Files,Path]=uigetfile('.csv','multiselect','off')
    Files=cellstr(Files);%converts singleton into cell string
%.. Update filename tags
    set(handles.ST_Filename,'String',Files);
%.. Update handles structure
    handles.Files=Files;
    handles.Path=Path;
    guidata(hObject, handles);

%.. Send update
    fprintf('Files loaded successfully.\r\n')

% --- Executes on button press in PB_Close2.
function PB_Close2_Callback(hObject, eventdata, handles)
%------------------------------------------------------------------
%--- CLOSE AND RETURN TO MAIN GUI, OR CLOSE
    GUI_hTimeline=getappdata(0,'GUI_hTimeline');

%.. Check if temperature profiles exist, and warn.
    if exist(fullfile(handles.Path,'TC1.mat'),'file') == 0
        Msg=sprintf('No temperature profile has been exported. Close anyway?\r\n')
            ;
        fprintf(Msg)
        button=questdlg(Msg,'No');
        switch button
            case 'Yes'
                fprintf('Closing...\r\n');
                close(GUI_hTimeline)
            case 'No'
                fprintf('Returning...\r\n');
        end
%.. Close
    else
        fprintf('Closing...\r\n');
        close(GUI_hTimeline)
    end
%------------------------------------------------------------------

```

```


function edit20_Callback(hObject, eventdata, handles)
% hObject    handle to edit20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit20 as text
%        str2double(get(hObject,'String')) returns contents of edit20 as a double


% --- Executes during object creation, after setting all properties.
function edit20_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit21_Callback(hObject, eventdata, handles)
% hObject    handle to edit21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit21 as text
%        str2double(get(hObject,'String')) returns contents of edit21 as a double


% --- Executes during object creation, after setting all properties.
function edit21_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end



function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12 as a double


% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_M3mwOn_Callback(hObject, eventdata, handles)
% hObject    handle to ET_M3mwOn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_M3mwOn as text
%        str2double(get(hObject,'String')) returns contents of ET_M3mwOn as a
    double


% --- Executes during object creation, after setting all properties.
function ET_M3mwOn_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_M3mwOn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit14 as text
%        str2double(get(hObject,'String')) returns contents of edit14 as a double


% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_M3st_Callback(hObject, eventdata, handles)
% hObject    handle to ET_M3st (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_M3st as text
%        str2double(get(hObject,'String')) returns contents of ET_M3st as a double


% --- Executes during object creation, after setting all properties.
function ET_M3st_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_M3st (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_M2mwOn_Callback(hObject, eventdata, handles)
% hObject    handle to ET_M2mwOn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_M2mwOn as text
%        str2double(get(hObject,'String')) returns contents of ET_M2mwOn as a
    double


% --- Executes during object creation, after setting all properties.
function ET_M2mwOn_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_M2mwOn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_M2st_Callback(hObject, eventdata, handles)
% hObject    handle to ET_M2st (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_M2st as text
%        str2double(get(hObject,'String')) returns contents of ET_M2st as a double


% --- Executes during object creation, after setting all properties.
function ET_M2st_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_M2st (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_M1mwOn_Callback(hObject, eventdata, handles)
% hObject    handle to ET_M1mwOn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_M1mwOn as text
%        str2double(get(hObject,'String')) returns contents of ET_M1mwOn as a
    double


% --- Executes during object creation, after setting all properties.
function ET_M1mwOn_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_M1mwOn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_M1st_Callback(hObject, eventdata, handles)
% hObject    handle to ET_M1st (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_M1st as text
%        str2double(get(hObject,'String')) returns contents of ET_M1st as a double
```

```
% --- Executes during object creation, after setting all properties.
function ET_M1st_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_M1st (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_fps_Callback(hObject, eventdata, handles)
% hObject    handle to ET_fps (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_fps as text
%        str2double(get(hObject,'String')) returns contents of ET_fps as a double


% --- Executes during object creation, after setting all properties.
function ET_fps_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_fps (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_P_Callback(hObject, eventdata, handles)
% hObject    handle to ET_P (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_P as text
%        str2double(get(hObject,'String')) returns contents of ET_P as a double
```

```
% --- Executes during object creation, after setting all properties.
function ET_P_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_P (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_IrTime_Callback(hObject, eventdata, handles)
% hObject    handle to ET_IrTime (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_IrTime as text
%        str2double(get(hObject,'String')) returns contents of ET_IrTime as a
    double


% --- Executes during object creation, after setting all properties.
function ET_IrTime_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_IrTime (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

B.6 SaveOutputGUI (export output `.txt` file to Excel template)

Listing B.6: SaveOutputGUI.m

```
function varargout = SaveOutputGUI(varargin)

% Last Modified by GUIDE v2.5 20-Oct-2016 12:39:16

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @SaveOutputGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @SaveOutputGUI_OutputFcn, ...
                   'gui_LayoutFcn',  []  , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before SaveOutputGUI is made visible.
function SaveOutputGUI_OpeningFcn(hObject, eventdata, handles, varargin)

% Choose default command line output for SaveOutputGUI
handles.output = hObject;

%try
    %Head=getappdata(0,'Head');
    %load(Head);
    %Create default excel file name (can change in GUI)
    %XLfileName = fullfile(Path,sprintf('%s-Results',Identi));
    %set(handles.ET_ExcelFile,'string',XLfileName);
    %Load latest data (else choose in GUI)
    %MATfileName = fillfile(Path,folder,'Results.mat');
    %set(handles.ET_DataFile,'string',MatfileName);
    %handles.FolderPath=fullfile(Path,folder);
%catch %i.e. if path is not defined
    %%Find files manually
```

```
    disp('Loading GUI... Data file not defined - entered catch')
%end

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes SaveOutputGUI wait for user response (see UIRESUME)
 uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = SaveOutputGUI_OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure -> alter to get desired
    output
FileInfo.XLfileName = get(handles.ET_ExcelFile,'String');
FileInfo.MATfileName = get(handles.ET_DataFile,'String');
FileInfo.WorkSheet = get(handles.ET_Worksheet,'String');
FileInfo.FolderPath = handles.FolderPath;

varargout{1} = FileInfo;
% The figure can be deleted now
delete(handles.figure1);



function ET_ExcelFile_Callback(hObject, eventdata, handles)
% Update modified file name on close


% --- Executes during object creation, after setting all properties.
function ET_ExcelFile_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_ExcelFile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ET_DataFile_Callback(hObject, eventdata, handles)
% hObject    handle to ET_DataFile (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_DataFile as text
%        str2double(get(hObject,'String')) returns contents of ET_DataFile as a double


% --- Executes during object creation, after setting all properties.
function ET_DataFile_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_DataFile (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in PB_SearchXL.
function PB_SearchXL_Callback(hObject, eventdata, handles)
% Ask user to search for the file (Excel file)
[XLfileName,Path]=uigetfile('.xlsm');
handles.XLfileName = fullfile(Path,XLfileName);
set(handles.ET_ExcelFile,'string',fullfile(Path,XLfileName));
guidata(hObject, handles);

% --- Executes on button press in PB_SearchData.
function PB_SearchData_Callback(hObject, eventdata, handles)
% Ask user to search for the file (matlab data file)
[MATfileName,Path]=uigetfile('.mat');
handles.MATfileName = fullfile(Path,MATfileName);
set(handles.ET_DataFile,'string',fullfile(Path,MATfileName));
handles.FolderPath=Path;
guidata(hObject, handles);


function ET_Worksheet_Callback(hObject, eventdata, handles)
% hObject    handle to ET_Worksheet (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ET_Worksheet as text
```

```
%        str2double(get(hObject,'String')) returns contents of ET_Worksheet as a double


% --- Executes during object creation, after setting all properties.
function ET_Worksheet_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ET_Worksheet (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in PB_Save.
function PB_Save_Callback(hObject, eventdata, handles)
disp('Saving data and returning to main GUI.')
close(gcf);

% --- Executes on button press in PB_Cancel.
function PB_Cancel_Callback(hObject, eventdata, handles)
disp('Closing figure without updating information.')
close(gcf);



% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
    if isequal(get(hObject, 'waitstatus'), 'waiting')
        % The GUI is still in UIWAIT, free with UIRESUME
        uiresume(hObject);
    else
        % The GUI is no longer waiting, just close it
        delete(hObject);
    end
```

APPENDIX C

Secondary filtering in Excel

C.1 Excel template

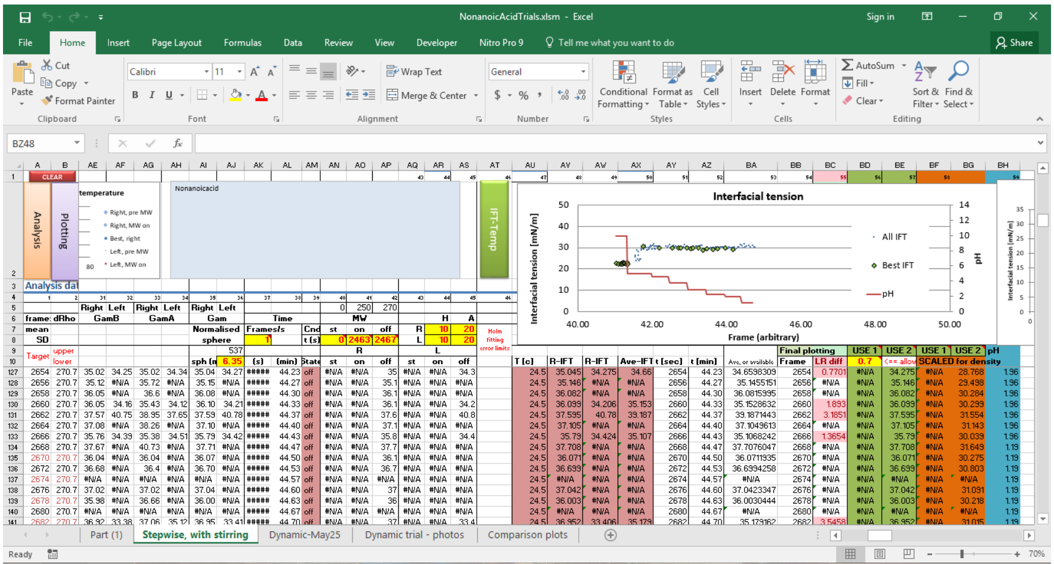

Figure C.1: Template for Excel files.

C.2 First-pass filtering

This code filters based on the fitting errors in the text file from Matlab. The user can specify the error thresholds.

Listing C.1: VBA code for first-pass filtering.

```
Option Explicit
Sub TopBar()
```

```
'
' TopBar Macro
' Fill in analysis for HOLM results generated in Matlab.

' Determine number of entries
Dim numrows As Long
    numrows = Range("F8").CurrentRegion.Rows.Count
Dim i As Integer
Dim j As Integer
Dim st As Integer
    st = 3 'vertical offset for plots

' Calculate mean and StDev for each column
For i = 6 To 19
    Cells(4 + st, i).Formula = "=Average(R[4]C:R[" & numrows - 4 & "]C)"
    Cells(5 + st, i).Formula = "=Stdev(R[3]C:R[" & numrows - 3 & "]C)"
Next i
For i = 20 To 21
    Cells(4 + st, i).Formula = "=TrimMean(R[4]C:R[" & numrows - 4 &
        "]C,0.1)"
    Cells(5 + st, i).Formula = "=Stdev(R[3]C:R[" & numrows - 3 & "]C)"
Next i

'max error values
For i = 10 To 12
    Cells(10, i).Formula = "=max(R[1]C:R[" & numrows - 3 & "]C)"
Next i
For i = 17 To 19
    Cells(10, i).Formula = "=max(R[1]C:R[" & numrows - 3 & "]C)"
Next i

'Left, upper & lower limts (T6 & T7)
Cells(6 + st, 20).Formula = "=R" & 4 + st & "C20 + R8C22*R[-1]C"
Cells(7 + st, 20).Formula = "=R" & 4 + st & "C20 - R8C22*R[-2]C"

'Right, upper & lower limts (U6 & U7)
Cells(6 + st, 21).Formula = "=R" & 4 + st & "C21 + R8C22*R[-1]C"
Cells(7 + st, 21).Formula = "=R" & 4 + st & "C21 - R8C22*R[-2]C"

'Normalised sphere radius
Cells(9, 36).Formula = "=trimmean(R[2]C[-16]:R[" & numrows - 4 &
    "]C[-15],0.2)"
```

```

'Filtering
    'Filter down each row, 6 conditions.
For i = 8 + st To numrows + 1 'row index '' set for st=3
    '%% filter conditions
    'Col 22: Left filter, 1 (R)
     Cells(i, 22).Formula = "=if(RC[-2]<R[" & 9 - i & "]C20, if(RC[-2]>R[" & 10 - i & "]C20,1,0),0)"

    'Col 23: Left filter, 2 (H best)
    Cells(i, 23).Formula = "=if(RC[-13]<R[" & 9 - i & "]C10,1,0)"

    'Col 24: Left filter, 3 (H ave)
    Cells(i, 24).Formula = "=if(RC[-12]<R[" & 9 - i & "]C12,1,0)"

    'Col 25: Right filter, 1 (R)
    Cells(i, 25).Formula = "=if(RC[-4]<R[" & 9 - i & "]C21, if(RC[-4]>R[" & 10 - i & "]C21,1,0),0)"

    'Col 26: Right filter, 2 (H best)
    Cells(i, 26).Formula = "=if(RC[-17]<R[" & 9 - i & "]C17,1,0)"
    'Col 27: Right filter, 3 (H ave)
    Cells(i, 27).Formula = "=if(RC[-8]<R[" & 9 - i & "]C19,1,0)"
    '%% filtered data
    'Col 28: Determine LR average for filtered cells
    Cells(i, 28).Formula = "=if(product(RC[-6]:RC[-1])=1,average(RC[-22],RC[-15]),NA())"

    'Col 29: copy frame
    Cells(i, 29).Formula = "=RC[-28]"
    'Col 30: copy temp
    Cells(i, 30).Formula = "=RC[-25]"

    'Col 31: Filtered interfacial tension (best), left
    Cells(i, 31).Formula = "=if(product(RC[-9]:RC[-7])=1,RC[-25],NA())"

    'Col 32: Filtered interfacial tension (best), right
    Cells(i, 32).Formula = "=if(product(RC[-7]:RC[-5])=1,RC[-19],NA())"

    'Col 33: Filtered interfacial tension (average), left
    Cells(i, 33).Formula = "=if(product(RC[-11]:RC[-9])=1,RC[-26],NA())"

```

```
'Col 34: Filtered interfacial tension (average), right
Cells(i, 34).Formula = "=if(product(RC[-9]:RC[-7])=1,RC[-20],NA())"

'Col 35: Gamma from normalised sphere radius, left
'Gamma=dRho*g*1000/(aBest/Scale*1000)^2;
Cells(i, 35).Formula =
    "=if(product(RC23:RC24)=1,RC2*9.81*1000/(RC8/(R10C36/2/R9C36)*1000)^2,NA())"

'Col 36: Gamma from normalised sphere radius, right
Cells(i, 36).Formula =
    "=if(product(RC26:RC27)=1,RC2*9.81*1000/(RC15/(R10C36/2/R9C36)*1000)^2,NA())"

'Col 37: Time (s)
Cells(i, 37).Formula = "=RC[-36]/R8C37"

'Col 38: Time (m)
Cells(i, 38).Formula = "=RC[-1]/60"

'Col 39: MW status
Cells(i, 39).Formula =
    "=IF(RC[-2]<R8C41,""st"",IF(RC[-2]<R8C42,""on"",""off""))"

'Col 40: Left, st
Cells(i, 40).Formula = "=IF(RC[-3]<R8C39,1,NA())*RC31"

'Col 41: Left, on
Cells(i, 41).Formula =
    "=IF(RC[-4]<R8C41,NA(),IF(RC[-4]<R8C42,1,NA()))*RC31"

'Col 42: Left, off
Cells(i, 42).Formula = "=IF(RC[-5]>R8C42,1,NA())*RC31"

'Col 43: Right, st
Cells(i, 43).Formula = "=IF(RC[-6]<R8C41,1,NA())*RC32"

'Col 44: Right, on
Cells(i, 44).Formula =
    "=IF(RC[-7]<R8C41,NA(),IF(RC[-7]<R8C42,1,NA()))*RC32"

'Col 45: Right, off
Cells(i, 45).Formula = "=IF(RC[-8]>R8C42,1,NA())*RC32"
```

```
Next i

'format
    Columns("A:AH").Select
        Selection.ColumnWidth = 6.71
    Range("F4:U5").Select
        Selection.NumberFormat = "0.000"
    Columns("V:AA").Select
        Selection.ColumnWidth = 2.57
    Rows("2").Select
        Selection.RowHeight = 135
    Rows("4").Select
        Selection.NumberFormat = "0"

    Range(Cells(8 + st, 35), Cells(numrows + 1, 36)).Select
    Selection.NumberFormat = "0.00"


    'set ranges for graph
    Dim Frames As Range
        Set Frames = Range(Cells(8 + st, 29), Cells(numrows, 29))
        ActiveWorkbook.Names.Add Name:="Frames", RefersTo:=Frames

    Dim Time As Range
        Set Time = Range(Cells(8 + st, 38), Cells(numrows, 38))
        ActiveWorkbook.Names.Add Name:="Time", RefersTo:=Time

    Dim Temp As Range
        Set Temp = Range(Cells(8 + st, 30), Cells(numrows, 30))
        ActiveWorkbook.Names.Add Name:="Temp", RefersTo:=Temp

    Dim GamLB As Range
        Set GamLB = Range(Cells(8 + st, 31), Cells(numrows, 31))
        ActiveWorkbook.Names.Add Name:="GamLB", RefersTo:=GamLB

    Dim GamRB As Range
        Set GamRB = Range(Cells(8 + st, 32), Cells(numrows, 32))
        ActiveWorkbook.Names.Add Name:="GamRB", RefersTo:=GamRB

    Dim GamLBn As Range
        Set GamLB = Range(Cells(8 + st, 35), Cells(numrows, 35))
        ActiveWorkbook.Names.Add Name:="GamLB", RefersTo:=GamLB
```

```

    Dim GamRBn As Range
        Set GamRB = Range(Cells(8 + st, 36), Cells(numrows, 36))
        ActiveWorkbook.Names.Add Name:="GamRB", RefersTo:=GamRB

    Dim dRho As Range
        Set dRho = Range(Cells(8 + st, 2), Cells(numrows, 2))
        ActiveWorkbook.Names.Add Name:="dRho", RefersTo:=dRho

    Dim stL As Range
        Set stL = Range(Cells(8 + st, 40), Cells(numrows, 40))
        ActiveWorkbook.Names.Add Name:="stL", RefersTo:=stL

    Dim onL As Range
        Set onL = Range(Cells(8 + st, 41), Cells(numrows, 41))
        ActiveWorkbook.Names.Add Name:="onL", RefersTo:=onL

    Dim offL As Range
        Set offL = Range(Cells(8 + st, 42), Cells(numrows, 42))
        ActiveWorkbook.Names.Add Name:="offL", RefersTo:=offL

    Dim stR As Range
        Set stR = Range(Cells(8 + st, 43), Cells(numrows, 43))
        ActiveWorkbook.Names.Add Name:="stR", RefersTo:=stR

    Dim onR As Range
        Set onR = Range(Cells(8 + st, 44), Cells(numrows, 44))
        ActiveWorkbook.Names.Add Name:="onR", RefersTo:=onR

    Dim offR As Range
        Set offR = Range(Cells(8 + st, 45), Cells(numrows, 45))
        ActiveWorkbook.Names.Add Name:="offR", RefersTo:=offR

End Sub
```

C.3 Second-pass filtering

This code filters based on the agreement between the left and right side fittings. The user can specify the error thresholds.

Listing C.2: VBA code for second-pass filtering.

```
Sub ColourChange()
''==================================================================
' Secondary Plotting Macro for microwave data, individual sheets
' --Plots IFT v Temp info
' --Colour "st, on, off" data for clarity
'==================================================================


'—————————————————————————————————————————————————————
'Explicit
'—————————————————————————————————————————————————————
Dim numrows As Integer
Dim cnt As Integer
Dim curCell As Range
Dim onCt As Integer
Dim offCt As Integer

numrows = Range("A11").End(xlDown).Row

For cnt = 11 To numrows
        Set curCell = ActiveSheet.Cells(cnt, 37)
        If curCell.Value < Range("AO8").Value Then
            onCt = cnt
        End If
        If curCell.Value < Range("AP8").Value Then
            offCt = cnt
        End If
Next cnt

ActiveSheet.Range(Cells(11, 37), Cells(onCt, 37)).Select
    With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .ThemeColor = xlThemeColorAccent6
        .TintAndShade = 0.599993896298105
        .PatternTintAndShade = 0
    End With

ActiveSheet.Range(Cells(onCt + 1, 38), Cells(offCt, 38)).Select
        With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
```

```
            .ThemeColor = xlThemeColorAccent3
            .TintAndShade = 0.399975585192419
            .PatternTintAndShade = 0
        End With

ActiveSheet.Range(Cells(offCt + 1, 39), Cells(numrows, 39)).Select
            With Selection.Interior
            .Pattern = xlSolid
            .PatternColorIndex = xlAutomatic
            .ThemeColor = xlThemeColorAccent2
            .TintAndShade = 0.399975585192419
            .PatternTintAndShade = 0
        End With
'--------------------------------------------------------------
' Pull out filtered IST against Temperature: Sep (lin) and average
'--------------------------------------------------------------
'' Lables
Range("AU10").Value = "T [c]"
Range("AB10").Value = "L-IFT"
Range("AW10").Value = "R-IFT"
Range("AX10").Value = "Ave-IFT"
Range("AY10").Value = "t [min]"
    Range("AU10:AY10").Select
    Selection.Font.Bold = True

'' populate start
For cnt = 11 To onCt
    Cells(cnt, 47).FormulaR1C1 = "=RC5"
    Cells(cnt, 48).FormulaR1C1 = "=RC40"
    Cells(cnt, 49).FormulaR1C1 = "=RC43"
    Cells(cnt, 50).FormulaR1C1 = "=average(RC48:RC49)"
    Cells(cnt, 51).FormulaR1C1 = "=RC38"
Next cnt


'' populate on
For cnt = onCt + 1 To offCt
    Cells(cnt, 47).FormulaR1C1 = "=RC5"
    Cells(cnt, 48).FormulaR1C1 = "=RC41"
    Cells(cnt, 49).FormulaR1C1 = "=RC44"
    Cells(cnt, 50).FormulaR1C1 = "=average(RC48:RC49)"
    Cells(cnt, 51).FormulaR1C1 = "=RC38"
```

```
Next cnt

'' populate off
For cnt = offCt + 1 To numrows
    Cells(cnt, 47).FormulaR1C1 = "=RC5"
    Cells(cnt, 48).FormulaR1C1 = "=RC42"
    Cells(cnt, 49).FormulaR1C1 = "=RC45"
    Cells(cnt, 50).FormulaR1C1 = "=average(RC48:RC49)"
    Cells(cnt, 51).FormulaR1C1 = "=RC38"
Next cnt

'colour
ActiveSheet.Range(Cells(11, 47), Cells(onCt, 50)).Select
    With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .ThemeColor = xlThemeColorAccent6
        .TintAndShade = 0.599993896298105
        .PatternTintAndShade = 0
    End With

ActiveSheet.Range(Cells(onCt + 1, 47), Cells(offCt, 50)).Select
        With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .ThemeColor = xlThemeColorAccent3
        .TintAndShade = 0.399975585192419
        .PatternTintAndShade = 0
    End With

ActiveSheet.Range(Cells(offCt + 1, 47), Cells(numrows, 50)).Select
        With Selection.Interior
        .Pattern = xlSolid
        .PatternColorIndex = xlAutomatic
        .ThemeColor = xlThemeColorAccent2
        .TintAndShade = 0.399975585192419
        .PatternTintAndShade = 0
    End With


ActiveSheet.ChartObjects("IFTT").Activate
ActiveChart.SeriesCollection(4).Select
```

```
    With Selection
                .Values = ActiveSheet.Range(Cells(onCt, 48),
                    Cells(offCt, 48))
                .XValues = ActiveSheet.Range(Cells(onCt, 47),
                    Cells(offCt, 47))
                .Name = "Left side, heating"
            End With

    ''series 2: Left side, cooling
    ActiveChart.SeriesCollection(2).Select
        With Selection
                .Values = ActiveSheet.Range(Cells(offCt + 1, 48),
                    Cells(numrows, 48))
                .XValues = ActiveSheet.Range(Cells(offCt + 1, 47),
                    Cells(numrows, 47))
                .Name = "Left side, cooling"
            End With

    ''series 3: Right side, heating
    ActiveChart.SeriesCollection(3).Select
        With Selection
                .Values = ActiveSheet.Range(Cells(onCt, 49),
                    Cells(offCt, 49))
                .XValues = ActiveSheet.Range(Cells(onCt, 47),
                    Cells(offCt, 47))
                .Name = "Right side, heating"
            End With

    ''series 4: Right side, cooling
    ActiveChart.SeriesCollection(1).Select
        With Selection
                .Values = ActiveSheet.Range(Cells(offCt + 1, 49),
                    Cells(numrows, 49))
                .XValues = ActiveSheet.Range(Cells(offCt + 1, 47),
                    Cells(numrows, 47))
                .Name = "Right side, cooling"
            End With


    With ActiveChart
            .SetElement (msoElementChartTitleAboveChart)
```

```
            Selection.Caption = "Interfacial␣tension,␣heating␣and␣cooling,␣both␣sides"

            .SetElement (msoElementPrimaryCategoryAxisTitleAdjacentToAxis)
            Selection.Caption = "Temperature␣[C]"

            .SetElement (msoElementPrimaryValueAxisTitleRotated)
            Selection.Caption = "Interfacial␣tension␣[mN/m]"

            .Axes(xlCategory).MinimumScale = 20
            .Axes(xlCategory).MaximumScale = 50

    End With 'active chart
End Sub
```

C.4 Additional formatting

This code provides addition formatting and automated plotting for the template.

Listing C.3: VBA code for additional formatting.

```
Option Explicit

Sub plotter()
'
' plotter Macro
'

'================================
'----------------------------------------------------
' Separate heating/cooling plots
'----------------------------------------------------
'' create chart
Dim cht As ChartObject
Dim Rng As Range

ActiveSheet.Shapes.AddChart.Select
Set cht = ActiveChart.Parent
Set Rng = ActiveSheet.Range("BA14:BM35")
    'size chart
    cht.Left = Rng.Left
    cht.Width = Rng.Width
```

```
    cht.Top = Rng.Top
    cht.Height = Rng.Height

''series 1: Left side, heating
With ActiveChart
    .ChartType = xlXYScatter
    ''clear existing series
    Do Until .SeriesCollection.Count = 0
            .SeriesCollection(1).Delete
        Loop 'http://peltiertech.com/
    With .SeriesCollection.NewSeries
                    .Values = ActiveSheet.Range(Cells(onCt, 48),
                        Cells(offCt, 48))
                    .XValues = ActiveSheet.Range(Cells(offCt + 1, 47),
                        Cells(numrows, 47))
                    .Name = "Left side, heating"
                End With

    ''series 2: Left side, cooling
    With .SeriesCollection.NewSeries
                    .Values = ActiveSheet.Range(Cells(offCt + 1, 48),
                        Cells(numrows, 48))
                    .XValues = ActiveSheet.Range(Cells(offCt + 1, 47),
                        Cells(numrows, 47))
                    .Name = "Left side, cooling"
                End With

    ''series 3: Right side, heating
    With .SeriesCollection.NewSeries
                    .Values = ActiveSheet.Range(Cells(onCt, 49),
                        Cells(offCt, 49))
                    .XValues = ActiveSheet.Range(Cells(onCt, 47),
                        Cells(offCt, 47))
                    .Name = "Right side, heating"
                End With

    ''series 4: Right side, cooling
    With .SeriesCollection.NewSeries
                    .Values = ActiveSheet.Range(Cells(offCt + 1, 49),
                        Cells(numrows, 49))
                    .XValues = ActiveSheet.Range(Cells(offCt + 1, 47),
                        Cells(numrows, 47))
```

```
                        .Name = "Right side, cooling"
                    End With



                .SetElement (msoElementChartTitleAboveChart)
                Selection.Caption = "Interfacial tension, heating and cooling, both sides"

                .SetElement (msoElementPrimaryCategoryAxisTitleAdjacentToAxis)
                Selection.Caption = "Temperature [C]"

                .SetElement (msoElementPrimaryValueAxisTitleRotated)
                Selection.Caption = "Interfacial tension [mN/m]"

                .Axes(xlCategory).MinimumScale = 20
                .Axes(xlCategory).MaximumScale = 50

End With 'active chart

''Titles and formatting

'_____________________________________________________
' Combined plot using LRave IFT
'_____________________________________________________

'=================================

'
    With ActiveChart.SeriesCollection(1)
                .MarkerStyle = 8
                .MarkerSize = 5
                .Format.Line.Visible = msoFalse
        End With


    With ActiveChart.SeriesCollection(2)
                .MarkerStyle = 9
                .MarkerSize = 5
                Selection.Format.Fill.Visible = msoFalse
            End With

```

```

     With ActiveChart.SeriesCollection(3)
             .MarkerStyle = 8
             .MarkerSize = 5
         Selection.Format.Line.Visible = msoFalse
         End With



    With ActiveChart.SeriesCollection(4)
             .MarkerStyle = 9
             .MarkerSize = 5
             .Format.Fill.Visible = msoFalse
             End With

    With ActiveChart

             .SetElement (msoElementChartTitleAboveChart)
             Selection.Caption = "Interfacial tension, heating and
                 cooling, both sides"

             .SetElement (msoElementPrimaryCategoryAxisTitleAdjacentToAxis)
             Selection.Caption = "Temperature [C]"

             .SetElement (msoElementPrimaryValueAxisTitleRotated)
             Selection.Caption = "Interfacial tension [mN/m]"

             .Axes(xlCategory).MinimumScale = 20
             .Axes(xlCategory).MaximumScale = 50
         End With

End Sub

Sub ReplotIFTT()
'=================================================================
' Replot IFT-Temp graphs into "combined" sheet
'=================================================================


Dim numrows As Integer

'=======Sheet 1=========
```

```
'numrows = Worksheet("Part 1").Range("A11").End(xlDown).Row
    Dim cnt As Integer
    Dim curCell As Range
    Dim onCt As Integer
    Dim offCt As Integer

    ActiveWorkbook.Sheets("Part 1").Activate

    numrows = Range("A11").End(xlDown).Row

    For cnt = 11 To numrows
            Set curCell = ActiveSheet.Cells(cnt, 37)
            If curCell.Value < Range("AO8").Value Then
                onCt = cnt
            End If
            If curCell.Value < Range("AP8").Value Then
                offCt = cnt
            End If
    Next cnt

Dim P1HT As Range
    Set P1HT = ActiveSheet.Range(Cells(onCt, 47), Cells(offCt, 47))
    ActiveWorkbook.Names.Add Name:="P1HT", RefersTo:=P1HT

Dim P1HL As Range
    Set P1HL = ActiveSheet.Range(Cells(onCt, 48), Cells(offCt, 48))
        ActiveWorkbook.Names.Add Name:="P1HL", RefersTo:=P1HL

Dim P1HR As Range
    Set P1HR = ActiveSheet.Range(Cells(onCt, 49), Cells(offCt, 49))
        ActiveWorkbook.Names.Add Name:="P1HR", RefersTo:=P1HR

Dim P1HC As Range
    Set P1HC = ActiveSheet.Range(Cells(onCt, 50), Cells(offCt, 50))
        ActiveWorkbook.Names.Add Name:="P1HC", RefersTo:=P1HC


Dim P1CT As Range
    Set P1CT = ActiveSheet.Range(Cells(offCt + 1, 47), Cells(numrows, 47))
            ActiveWorkbook.Names.Add Name:="P1CT", RefersTo:=P1CT
Dim P1CL As Range
    Set P1CL = ActiveSheet.Range(Cells(offCt + 1, 48), Cells(numrows, 48))
```

```
                ActiveWorkbook.Names.Add Name:="P1CL", RefersTo:=P1CL
Dim P1CR As Range
    Set P1CR = ActiveSheet.Range(Cells(offCt + 1, 49), Cells(numrows, 49))
                ActiveWorkbook.Names.Add Name:="P1CR", RefersTo:=P1CR
Dim P1CC As Range
    Set P1CC = ActiveSheet.Range(Cells(offCt + 1, 50), Cells(numrows, 50))
                ActiveWorkbook.Names.Add Name:="P1CC", RefersTo:=P1CC

'=========Sheet 2==========
    ActiveWorkbook.Sheets("Part 2").Activate

    numrows = Range("A11").End(xlDown).Row

    For cnt = 11 To numrows
            Set curCell = ActiveSheet.Cells(cnt, 37)
            If curCell.Value < Range("AO8").Value Then
                onCt = cnt
            End If
            If curCell.Value < Range("AP8").Value Then
                offCt = cnt
            End If
    Next cnt

Dim P2HT As Range
    Set P2HT = ActiveSheet.Range(Cells(onCt, 47), Cells(offCt, 47))
                ActiveWorkbook.Names.Add Name:="P2HT", RefersTo:=P2HT
Dim P2HL As Range
    Set P2HL = ActiveSheet.Range(Cells(onCt, 48), Cells(offCt, 48))
                    ActiveWorkbook.Names.Add Name:="P2HL", RefersTo:=P2HL
Dim P2HR As Range
    Set P2HR = ActiveSheet.Range(Cells(onCt, 49), Cells(offCt, 49))
                    ActiveWorkbook.Names.Add Name:="P2HR", RefersTo:=P2HR
Dim P2HC As Range
    Set P2HC = ActiveSheet.Range(Cells(onCt, 50), Cells(offCt, 50))
                ActiveWorkbook.Names.Add Name:="P2HC", RefersTo:=P2HC

Dim P2CT As Range
    Set P2CT = ActiveSheet.Range(Cells(offCt + 1, 47), Cells(numrows, 47))
                    ActiveWorkbook.Names.Add Name:="P2CT", RefersTo:=P2CT
Dim P2CL As Range
    Set P2CL = ActiveSheet.Range(Cells(offCt + 1, 48), Cells(numrows, 48))
                    ActiveWorkbook.Names.Add Name:="P2CL", RefersTo:=P2CL
```

```
Dim P2CR As Range
    Set P2CR = ActiveSheet.Range(Cells(offCt + 1, 49), Cells(numrows, 49))
                ActiveWorkbook.Names.Add Name:="P2CR", RefersTo:=P2CR
Dim P2CC As Range
    Set P2CC = ActiveSheet.Range(Cells(offCt + 1, 50), Cells(numrows, 50))
                ActiveWorkbook.Names.Add Name:="P2CC", RefersTo:=P2CC


'==========Sheet 3=========
    ActiveWorkbook.Sheets("Part␣3").Activate

    numrows = Range("A11").End(xlDown).Row

    For cnt = 11 To numrows
            Set curCell = ActiveSheet.Cells(cnt, 37)
            If curCell.Value < Range("AO8").Value Then
                onCt = cnt
            End If
            If curCell.Value < Range("AP8").Value Then
                offCt = cnt
            End If
    Next cnt

Dim P3HT As Range
    Set P3HT = ActiveSheet.Range(Cells(onCt, 47), Cells(offCt, 47))
                ActiveWorkbook.Names.Add Name:="P3HT", RefersTo:=P3HT
Dim P3HL As Range
    Set P3HL = ActiveSheet.Range(Cells(onCt, 48), Cells(offCt, 48))
                    ActiveWorkbook.Names.Add Name:="P3HL", RefersTo:=P3HL
Dim P3HR As Range
    Set P3HR = ActiveSheet.Range(Cells(onCt, 49), Cells(offCt, 49))
                    ActiveWorkbook.Names.Add Name:="P3HR", RefersTo:=P3HR
Dim P3HC As Range
    Set P3HC = ActiveSheet.Range(Cells(onCt, 50), Cells(offCt, 50))
                    ActiveWorkbook.Names.Add Name:="P3HC", RefersTo:=P3HC


Dim P3CT As Range
    Set P3CT = ActiveSheet.Range(Cells(offCt + 1, 47), Cells(numrows, 47))
                    ActiveWorkbook.Names.Add Name:="P3CT", RefersTo:=P3CT
Dim P3CL As Range
    Set P3CL = ActiveSheet.Range(Cells(offCt + 1, 48), Cells(numrows, 48))
```

```
                    ActiveWorkbook.Names.Add Name:="P3CL", RefersTo:=P3CL
Dim P3CR As Range
    Set P3CR = ActiveSheet.Range(Cells(offCt + 1, 49), Cells(numrows, 49))
                    ActiveWorkbook.Names.Add Name:="P3CR", RefersTo:=P3CR
Dim P3CC As Range
    Set P3CC = ActiveSheet.Range(Cells(offCt + 1, 50), Cells(numrows, 50))
                    ActiveWorkbook.Names.Add Name:="P3CC", RefersTo:=P3CC


'———————————————————————————————————————————————————————————
' Replot existing graphs
'———————————————————————————————————————————————————————————
 ActiveWorkbook.Sheets("Combined").Activate
'============= 1st ==============
    ActiveSheet.ChartObjects("P1").Activate
    ActiveChart.SeriesCollection(4).Select
    With Selection
                    .Values = Range("P1HL")
                    .XValues = Range("P1HT")
                    .Name = "Left␣side,␣heating"
                End With

    ''series 2: Left side, cooling
    ActiveChart.SeriesCollection(2).Select
        With Selection
                    .Values = Range("P1CL")
                    .XValues = Range("P1CT")
                    .Name = "Left␣side,␣cooling"
                End With

    ''series 3: Right side, heating
    ActiveChart.SeriesCollection(3).Select
        With Selection
                    .Values = Range("P1HR")
                    .XValues = Range("P1HT")
                    .Name = "Right␣side,␣heating"
                End With

    ''series 4: Right side, cooling
    ActiveChart.SeriesCollection(1).Select
        With Selection
                    .Values = Range("P1CR")
```

```
                    .XValues = Range("P1CT")
                    .Name = "Right side, cooling"
                End With

'========== 2nd ===========
    ActiveSheet.ChartObjects("P2").Activate
    ActiveChart.SeriesCollection(4).Select
    With Selection
                    .Values = Range("P2HL")
                    .XValues = Range("P2HT")
                    .Name = "Left side, heating"
                End With

    ''series 2: Left side, cooling
    ActiveChart.SeriesCollection(2).Select
        With Selection
                    .Values = Range("P2CL")
                    .XValues = Range("P2CT")
                    .Name = "Left side, cooling"
                End With

    ''series 3: Right side, heating
    ActiveChart.SeriesCollection(3).Select
        With Selection
                    .Values = Range("P2HR")
                    .XValues = Range("P2HT")
                    .Name = "Right side, heating"
                End With

    ''series 4: Right side, cooling
    ActiveChart.SeriesCollection(1).Select
        With Selection
                    .Values = Range("P2CR")
                    .XValues = Range("P2CT")
                    .Name = "Right side, cooling"
                End With

'========= 3rd ==========
    ActiveSheet.ChartObjects("P3").Activate
        ActiveChart.SeriesCollection(4).Select
    With Selection
                    .Values = Range("P3HL")
```

```
                .XValues = Range("P3HT")
                .Name = "Left side, heating"
            End With

    ''series 2: Left side, cooling
    ActiveChart.SeriesCollection(2).Select
        With Selection
                .Values = Range("P3CL")
                .XValues = Range("P3CT")
                .Name = "Left side, cooling"
            End With

    ''series 3: Right side, heating
    ActiveChart.SeriesCollection(3).Select
        With Selection
                .Values = Range("P3HR")
                .XValues = Range("P3HT")
                .Name = "Right side, heating"
            End With

    ''series 4: Right side, cooling
    ActiveChart.SeriesCollection(1).Select
        With Selection
                .Values = Range("P3CR")
                .XValues = Range("P3CT")
                .Name = "Right side, cooling"
            End With

'=========Combined=========
    ActiveSheet.ChartObjects("Comb").Activate
    ActiveChart.SeriesCollection(1).Select
    With Selection
                .Values = Range("P1HC")
                .XValues = Range("P1HT")
                .Name = "Part 1, heating"
            End With

    ''series 2: Part 1, cooling
    ActiveChart.SeriesCollection(2).Select
        With Selection
                .Values = Range("P1CC")
                .XValues = Range("P1CT")
```

```
                .Name = "Part 1, cooling"
            End With

    ''series 3: Part 2, heating
    ActiveChart.SeriesCollection(3).Select
        With Selection
                .Values = Range("P2HC")
                .XValues = Range("P2HT")
                .Name = "Part 2, heating"
            End With

    ''series 4: Part 2, cooling
    ActiveChart.SeriesCollection(4).Select
        With Selection
                .Values = Range("P2CC")
                .XValues = Range("P2CT")
                .Name = "Part 2, cooling"
            End With
    ''series 5: Part 2, heating
    ActiveChart.SeriesCollection(5).Select
        With Selection
                .Values = Range("P3HC")
                .XValues = Range("P3HT")
                .Name = "Part 3, heating"
            End With

    ''series 6: Part 2, cooling
    ActiveChart.SeriesCollection(6).Select
        With Selection
                .Values = Range("P3CC")
                .XValues = Range("P3CT")
                .Name = "Part 3, cooling"
            End With

End Sub

Sub clear()
'==================================================================
' Clear data in sheet
'==================================================================
On Error Resume Next
'Overflow error (#6) will occur if A11 is empty
```

```

'If IsEmpty(Range("A11").Value = 1) Then
    'MsgBox "No data to clear - 'A11' is empty. Exit sub"
   ' Exit Sub
'End If

Dim numrows As Integer
numrows = Range("A11").End(xlDown).Row

Range(Cells(11, 1), Cells(numrows, 51)).Select
Selection.Formula = ""
    With Selection.Interior
        .Pattern = xlNone
        .TintAndShade = 0
        .PatternTintAndShade = 0
    End With

Range("D3", "F3").Formula = ""
Range("J3", "N3").Formula = ""
Range("Q3", "S3").Formula = ""
Range("V22").Formula = ""

End Sub
```