TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechanical Engineering

Analysis Tool for the Evaluation of Measurements of the Single Bubble and Bubbles Structures Dynamic

Master thesis

Study programme:	N2301 – Mechanical Engineering
Study branch:	2302T010 – Machines and Equipment Design
Author:	**Runer Shiloh Salonga**
Supervisor:	Ing. Miloš Müller, Ph.D.

Liberec 2019

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechanical Engineering

Master Thesis Assignment Form

Analysis Tool for the Evaluation of Measurements of the Single Bubble and Bubbles Structures Dynamic

Name and Surname:	**Runer Shiloh Salonga**
Identification Number:	S17000390
Study Programme:	N2301 Mechanical Engineering
Specialisation:	Machines and Equipment Design
Assigning Department:	Department of Power Engineering Equipment
Academic Year:	**2018/2019**

Rules for Elaboration:

1. State of art in the investigation of the behavior of the single cavitation bubble and bubble structures close to a solid wall.
2. Definition and preparation of the experimental setup for the measurement of the single bubble and bubbles structures dynamic using acoustical and optical methods.
3. Preparing the tool for the evaluation of the bubble dynamic experiments.
4. Realization of experiments for the comparison of the single bubble and bubbles structures dynamic.
5. Evaluation of the acoustical and optical measurement using the designed tool.
6. Detailed analysis of the single bubble and bubbles structures dynamic based on the results obtained from the acoustical and optical methods.

Scope of Graphic Work:	10 stran
Scope of Report:	50 stran
Thesis Form:	printed
Thesis Language:	English

List of Specialised Literature:

[1] BRENNEN, Christopher E. *Cavitation and bubble dynamics.* New York: Oxford University Press, 1995. ISBN 0195094093.

[2] FRANC, Jean-Pierre a Jean-Marie MICHEL. *Fundamentals of cavitation.* Boston: Kluwer Academic Publishers, c2004. ISBN 1402022328.

[3] KIM, Ki-Han, Georges CHAHINE, Jean-Pierre FRANC a Ayat KARIMI. *Advanced experimental and numerical techniques for cavitation erosion prediction.* Dordrecht: Springer, [2014]. Fluid mechanics and its applications, v. 106.

Thesis Supervisor:	Ing. Miloš Müller, Ph.D. Department of Power Engineering Equipment
Date of Thesis Assignment:	1 November 2018
Date of Thesis Submission:	30 April 2020

Technická univerzita v Liberci
-8-

[illegible]

prof. Dr. Ing. Petr Lenfeld
Dean

[illegible]

doc. Ing. Václav Dvořák, Ph.D.
Head of Department

Liberec 1 February 2019

Declaration

I hereby certify I have been informed that my master thesis is fully governed by Act No. 121/2000 Coll., the Copyright Act, in particular Article 60 – School Work.

I acknowledge that the Technical University of Liberec (TUL) dœs not infringe my copyrights by using my master thesis for the TUL's internal purposes.

I am aware of my obligation to inform the TUL on having used or granted license to use the results of my master thesis; in such a case the TUL may require reimbursement of the costs incurred for creating the result up to their actual amount.

I have written my master thesis myself using the literature listed below and consulting it with my thesis supervisor and my tutor.

At the same time, I honestly declare that the texts of the printed version of my master thesis and of the electronic version uploaded into the IS STAG are identical.

[illegible]

2. 5. 2019 Runer Shiloh Salonga

Analysis Tool for the Evaluation of Measurements of the Single Bubble and Bubbles Structures Dynamic

Abstract

Cavitation is a phenomenon that is commonly observed in turbines, water pumps, and other similar mechanisms which involve the high-speed and high-pressure movement of fluids, as it is the change in state of a material due to a fast pressure change, in particular from the liquid to the solid state. As a result, bubbles of water vapor form within the fluid, which interact with solid surfaces in a short span of time, particularly the creation of jets during the collapse of these bubbles. These jets inflict force on these solid ”walls”, and this interaction between bubble and solid wall may cause positive or negative effects, which may be harnessed when better understood.

This project aims to develop an analysis tool, developed in C++, that will aid in the understanding of the dynamics of the cavitation bubble in the context of a dynamic flow system with water as the medium. Investigations are made with the tool on an experimental setup consisting of a spark generator for creating the cavitation bubbles, monitored by a high-speed camera and measured by a polyvinylidene fluoride (PVDF) piezœlectric sensor. The camera is used to observe the development of the bubble and determine its size through image processing of the resulting images that are taken, while the PVDF sensor is a low cost option for measuring pressure on a flat surface, as it can convert pressure into voltage.

In this study, the differences between the single- and double-bubble spark generated setups are observed and analyzed through the optical and acoustical measurement methods. From these comparisons, the project aims to set a baseline for continued study of further specialized measurement and analysis tools of cavitation behavior.

Keywords: Cavitation, High-speed camera, PVDF sensor, Measurement tool, Bubble dynamics, Image recognition, C++

Acknowledgements

I thank God for this life and giving me this opportunity to explore masteral studies in Europe. I dedicate this work to my family in the Philippines; my parents Ruben and Nerissa with my sister Benessa, who gave their tireless support in my pursuit of studies abroad. I also dedicate this work to my partner, Krizzia Navarro, whose unwavering love and patience continue to push me to succeed, even from a very long distance away.

I extend my greatest thanks to the Technical University of Liberec, especially to my thesis adviser, Ing. Miloš Müller, Ph.D., for his patience, insight, and assistance for my completion of this master's thesis. I also thank Ing. Jan Hujer for his assistance in troubleshooting the experimental setup. I thank my fellow students Jiří Lampa, Gowtham Ananthasayanan, and Myka Mæ Duran for their assistance in hours of work in the Anemometry Laboratory. My heartfelt thanks also to doc. Ing. Václav Dvořák, Ph.D. of the Department of Power Engineering Equipment of TUL for overseeing our progress through the masters program. Special credit to doc. RNDr. Pavel Satrapa, Ph.D. of NTI - TUL for the LaTeX template used for this text, and Michal Hoftich for the `biblatex-iso690` library for the ČSN ISO 690 bibliography citations.

This publication was written at the Technical University of Liberec as part of the project *"Experimental, theoretical and numerical research in fluid mechanics and thermomechanics, no. 21291"* with the support of the Specific University Research Grant, as provided by the Ministry of Education, Youth and Sports of the Czech Republic in the year 2019. My greatest gratitude to the Ministry for this grant and also for this huge opportunity, the Strategy for Granting Government Scholarships for Students from Developing Countries.

I thank the Czech Embassy in Manila, especially former Deputy Ambassador Jan Vytopil and former Ambassador Jaroslav Olša, Jr., as well as former TUL scholar Anjelynn Guanlao, for their support to the Filipino government scholars in this regime. My deepest gratitude also to the Gaylon family and the Filipino Catholic Community in Prague for being my family far away from home.

Thank you very much to all of you and to many more I have not mentioned for your support in making this project possible.

Runer Shiloh Salonga

Author

+

Ad Majórem Dei Glóriam

Contents

List of abbreviations

BNC	Bayonet Neill–Concelman (connector)
CCD	Charge-Coupled Device
CSV	Comma Separated Values
DAQ	Data Acquisition
DC	Direct Current
DLL	Dynamic Linked Library
FPS	Frames per Second
IDE	Integrated Development Environment
LED	Light-Emitting Diode
MOSFET	Metal-Oxide-Semiconductor Field Effect Transistor
NI	National Instruments
OpenCV	Open Source Computer Vision Library
PVDF	Polyvinylidene Fluoride
RC	Resistive-Capacitive (circuit)
ROI	Region of Interest
SDK	Software Development Kit
TIFF	Tagged Image File Format
BPx	Bubble Period (1 or 2)
CAx	Collapse Area (1 or 2)
CFx	Collapse Force (1 or 2)
CJx	Collapse Impulse (1 or 2)
CVx	Collapse Voltage (1 or 2)
CWx	Collapse Width (1 or 2)
ETx	Estimated Collapse Time (0, 0.5, 1, 2)
Tx	Collapse Time (1 or 2)

List of Figures

1 Introduction

Cavitation is defined as the breakdown of a liquid under the condition of very low pressure [1]. It is a mechanical phenomenon observed in fluids which is a vital consideration in different fields of engineering, most notably in ærospace and energetics engineering due to its effects on water pumps, as well as in the fields of medicine and biœngineering. Hydrodynamic cavitation is when this liquid medium is in motion, happening commonly in tubes, propellers, valves, nozzles, and similar structures. As such, the study of hydrodynamic cavitation has been explored in numerous studies throughout the years [2].

Understanding cavitation is important due to its effects on the systems where fluid flow is an important component, particularly in the earlier mentioned fields of study. The aim and rationale of the study is to create a tool to aid in the analysis of cavitation regimes and bubble behavior through the comparison of experimental sensor data to further the understanding of what might happen in this process.

1.1 Theory of Cavitation Behavior

Cavitation behavior can be best explained by the P-T (pressure-temperature) phase diagram for a fluid as shown in Figure 1.1.

Cavitation occurs as cavities of vapor inside a homogeneous liquid medium under certain situations, both in cases where the liquid is static or in motion [1]. As it is comparable to the phenomenon of boiling since they are both phase changes, they are differentiated in this figure. Once a liquid reaches a threshold temperature at a specific pressure as the temperature is increased, it will undergo vaporization (boiling) and change phase into gas. However, it is also possible, as shown in Figure 1.1, to reach a threshold pressure at a specific temperature as the pressure is decreased. At this moment, the liquid will also undergo phase change into gas, specifically as cavitation.

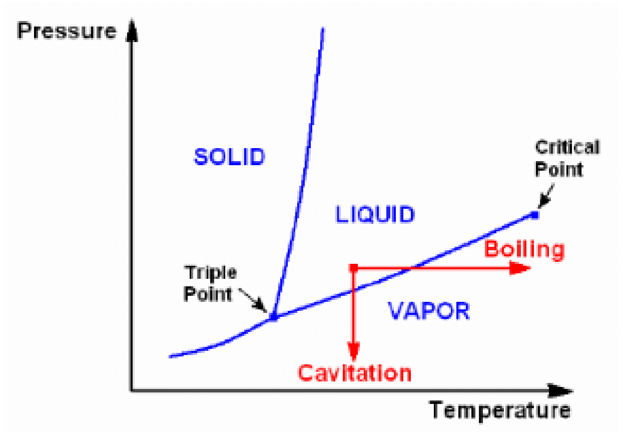


Figure 1.1: Pressure-Temperature Phase Diagram for a Simple Fluid [3]

1.1.1 Bubble Formation

During cavitation, spherical bubbles are formed within the body of the liquid [3]. As the liquid medium undergœs a phase change into gas during the change in pressure, this formed gas usually manifests as spherical air bubbles suspended in the medium in the simplest case [4]. The behavior of these bubbles has then been studied extensively since one of the earliest studies on the phenomenon by Rayleigh in 1917 [5, 6].

Most of the areas where these bubbles usually occur are those sites in the fluid which are most vulnerable are bubbles sized in the micrometers range (microbubbles) or solid particles in the fluid containing these microbubbles. They are called cavitation nuclei [3], and these nuclei comprise most of the nucleation sites of the fluid during the cavitation process. Bubbles formed from the nucleation process are typically around 5 μm to 100 μm.

1.1.2 Bubble Growth

Once a nucleation site undergœs the phase change and the bubble is formed, it continues to grow in size as it gains more vapor and gas from the cavitation process. As the bubble forms in a reducing pressure environment, its growth follows under the conditions of the Rayleigh-Plesset equation [1],

$$\rho\left[R\ddot{R}+\frac{3}{2}\dot{R}^2\right]=p_v-p_\infty(t)+p_{g0}\left(\frac{R_0}{R}\right)^{3\gamma}-\frac{2S}{R}-4\nu_L\frac{\dot{R}}{R} \tag{1.1}$$

which determines the relationship between the bubble radius R and its derivatives (changes in time) $\dot{R}$ and $\ddot{R}$ while at a specific reference pressure r_∞, initial gas pressure p_{g0}, and surface tension S.

The Rayleigh-Plesset equation takes into consideration an incompressible fluid of density ρ and kinematic viscosity ν_L, where the air content of the formed bubble is constant, and that the bubble is filled to saturation with vapor which is at a partial pressure (bubble pressure) p_B at the reference temperature of the liquid T_∞ [1]. The equation can also be further rearranged as [3]:

$$\frac{p_v(t) - p_\infty(t)}{\rho} = \left[R\ddot{R} + \frac{3}{2}\dot{R}^2\right] + \frac{4\nu_L}{R}\dot{R} + \frac{2S}{\rho R} \tag{1.2}$$

considering that bubble vapor pressure p_v and fluid pressure p_∞ both vary in time [7]. Likewise, an equilibrium radius R_0 can be defined as:

$$R_0 = \frac{2S}{p_v - p_\infty} \tag{1.3}$$

This equilibrium radius is used in other calculations such as the circumstances of bubble collapse to be discussed further, or the natural frequency of the oscillations of a single bubble in an infinitely vast fluid domain, defined in the equation as [1, 3, 7]:

$$f_0 = \frac{1}{2\pi R_0}\sqrt{\frac{1}{\rho}\left[3\gamma p_{g0} - \frac{2S}{R_0}\right]} \tag{1.4}$$

From the Rayleigh-Plesset equation, it can be deduced that on the early stages of bubble lifetime, the growth of the bubble then is non-linear and follows a relatively steadier path than after its eventual collapse [3]. The following figure shows an approximate graph of the growth of the bubble, which appears to follow the Rayleigh-Plesset model. In this particular study, the growth proceeds in around 0.2 milliseconds and dissipates thereafter within microseconds [8].

1.1.3 Bubble Collapse

After some time of growth, the bubble will shrink in accordance with the Rayleigh-Plesset equation. At some such time, the radius will attain its minimum value. As this happens, the bubble is said to have undergone collapse. The time it takes for the collapse to occur, known as the Rayleigh time τ can be computed through the manipulation of the Rayleigh-Plesset equation [1, 7]:

$$\tau = \sqrt{\frac{3}{2}\frac{\rho}{p_\infty - p_v}}\int_0^{R_0}\frac{dR}{\sqrt{\frac{R_0^3}{R^3} - 1}} \tag{1.5}$$

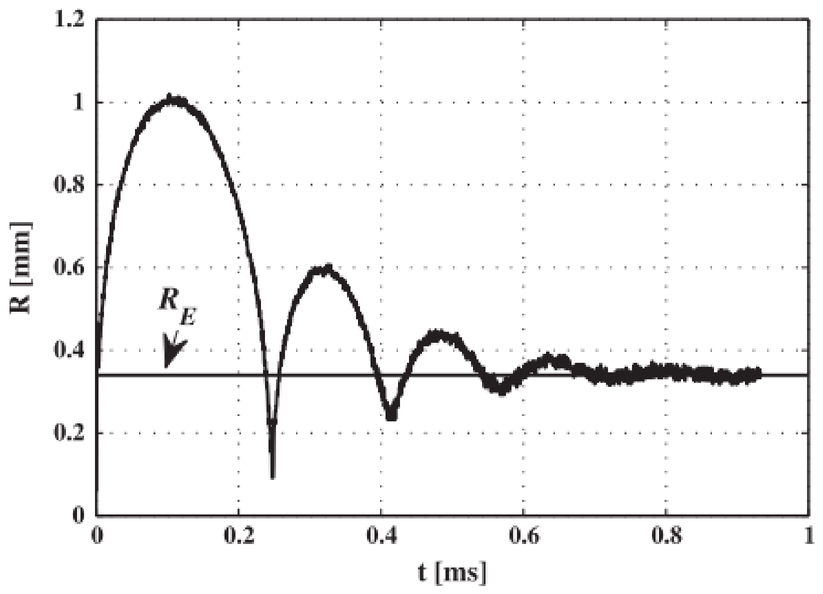


Figure 1.2: Typical bubble radius growth over time [8]

This can be further simplified as:

$$\tau = 0.915 R_0 \sqrt{\frac{\rho}{p_\infty - p_v}} \tag{1.6}$$

as 0.915 is equivalent to $\sqrt{\frac{\pi}{6}\frac{\Gamma(5/6)}{\Gamma(4/3)}}$, where Γ is the gamma function [1]. Through this simplification, it is possible to derive the Rayleigh time from the equilibrium radius R_0, the pressure difference between fluid and vapor $p_\infty - p_v$, and the fluid density ρ.

The evolution of the bubble at the end of this collapse is then estimated as [1]:

$$\frac{R}{R_0} \approx 1.87 \left[\frac{\tau - t}{\tau}\right]^{2/5} \tag{1.7}$$

and from here, the bubble evolves to a top size that will be smaller than when it has reached maximum size from bubble formation. Also, the bubble will no longer maintain a spherical shape after this collapse, as from observations in experiments, the bubble splits into multiple fragments [9]. This explosive cavitation can be attributed to the pressure drop within bubble dropping to the critical pressure, which is the pressure that when a nucleus is subjected below it, equilibrium will not be possible and the bubble at the nucleus will grow and explode after collapse [10].

This figure from Chahine, et al.'s study [10] shows the cavitation curve following the Rayleigh-Plesset equation and the subsequent smaller rebounds.

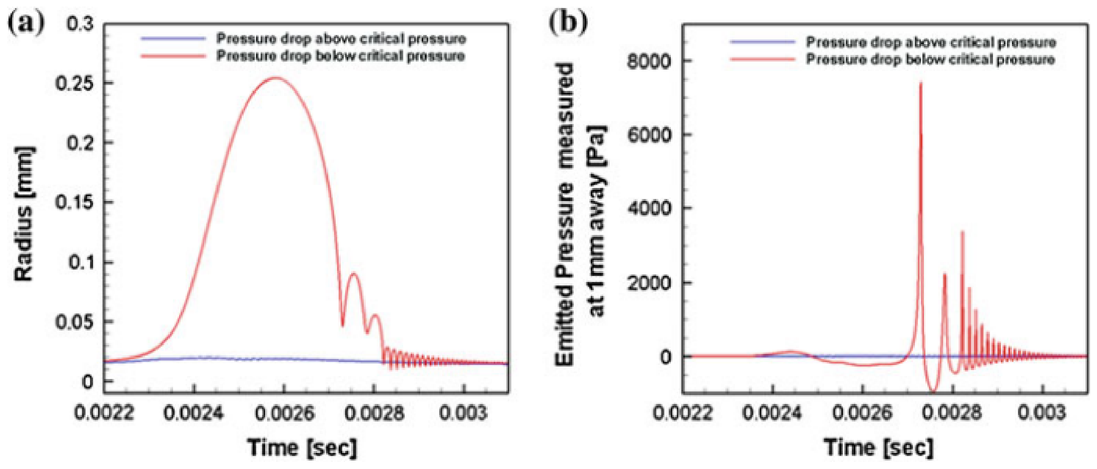


Figure 1.3: Bubble Radius and Emitted Pressure 1 mm away over time [10]

Explosive cavitation occurs upon having a pressure drop smaller than the critical pressure, while when the drop is above critical pressure, there is a flat development of the nucleation site without explosion. Also of note are the pressure levels emitted by the bubble 1 mm away as the graph on the right, which shows thin impulse spikes whenever the rebound of the bubble reaches a maximum.

1.2 Interaction of Bubbles with Solid Walls

The cavitation bubbles, aside from the growth and collapse, interact with solid surface walls differently than when they develop on an infinitely or semi-infinitely considered surrounding of fluid. Whenever a bubble is at collapse near a solid wall, as we will henceforth refer to a solid surface which acts as a boundary denoting the end of the fluid area, its shape is distorted toward the next rebound and a phenomenon known as the re-entrant jet occurs.

1.2.1 Bubble Shape Distortion

As an interface of accelerating gas and liquid, the surface of the bubble undergœs rapid, unstable changes as studied by Rayleigh and Taylor [3, 11]. As a result, as the largest change in radius happens in the collapse, the bubble is also at its most unstable, manifesting as a distortion of the bubble shape.

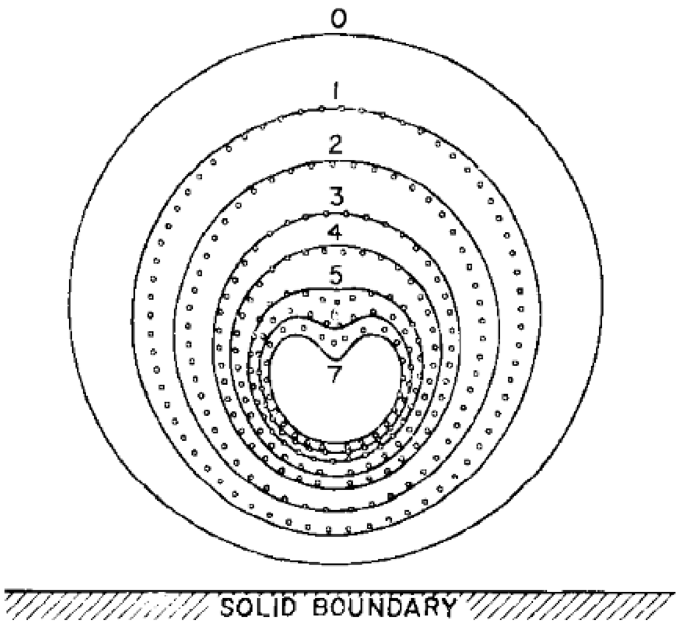


Figure 1.4: Shape Distortion of Bubble, comparison of a study of Lauterborn and Bolle (1975) with the theoretical results of Plesset and Chapman (1971) [7, 3]

The part of the bubble which is farthest away from the solid wall has a higher acceleration towards this wall, which leads to the bubble closing in on itself. This accelerating part culminates into a sharp fluid pillar called a re-entrant jet (or a microjet due to its size).

Bubbles also interact with other bubbles and have their shape distorted in the process. Two bubbles, depending on their distance from each other, either maintain their spherical shapes, or flatten near each other and combine as one bubble in a process described in Bremond, et al. as coalescence [12]. In the latter scenario, denoted as strong coupling, the Rayleigh-Plesset equation no longer describes the shape of the bubble as the sphere shape of the bubbles are lost, and are instead described with the simulation of potential flow boundary integrals [12].

Single and double bubble behaviors can be simulated as it was done in Mao, et al.'s study. This particular simulation and computational study replicates the environment of the experimental study by using an improved version of the lattice Boltzmann Shan-Chen model, in conjunction with the Carnahan-Starling Equation of State and Exact Differential Method [13].

1.2.2 Shock Wave

The formation of the cavitation bubble and its subsequent collapses, due to the high pressure changes involved in the process, produce shock waves that propagate through the liquid medium.

As a bubble grows out of a nucleus and expands during a cavitating flow, it will typically grow to 100 times its initial size, leading to pressure changes multiple orders larger. The bubble initial pressure, when for example starting at 10^-6 bar (assuming that the partial pressure of the gas in the bubble is 1 bar), can jump to a maximum pressure of 10^10 bar during the process. There is observed then, barring the mitigation caused by diffusion of gas, that a shock wave has a high potential to occur during the cavitation process [4].

Early studies in cavitation focused on the formation of this shock wave from bubble collapse, but as the microjet phenomenon is discovered, study focus shifted to investigating this pattern [3].

1.2.3 Re-Entrant Jet

From the shape distortion, the phenomenon of the microjet has then been observed and is one of the main focuses in observing cavitation [3]. A re-entrant jet forms in the distortion of the bubble, causing a disturbance in the liquid medium. This interaction causes an acceleration of fluid which hits the wall, causing damage.

Cavitation damage arises from a collection of jets formed by this process on a multitude of bubbles all interacting with the solid wall in this manner. These

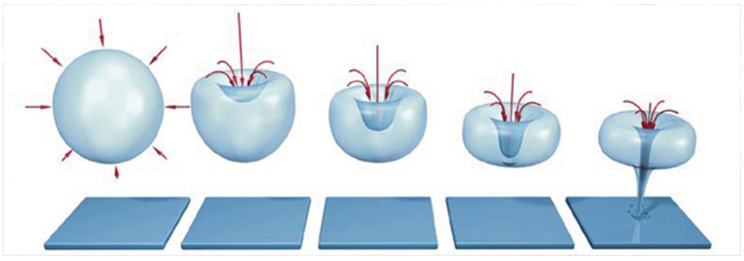


Figure 1.5: Visualization of Shape Distortion and Re-Entrant Jet [14]

jets cause a high amount of stress on the wall, and continuous repetitions of the jet formation eventually cause heavy damage over time [3]. Pressure changes not only induce these microjets, but also high-pressure waves in the fluid. However, in experiments, it has been discovered that the effect of cavitation and re-entrant jets can be mitigated with particles reaching a certain critical size [15].

1.2.4 Collapse Patterns

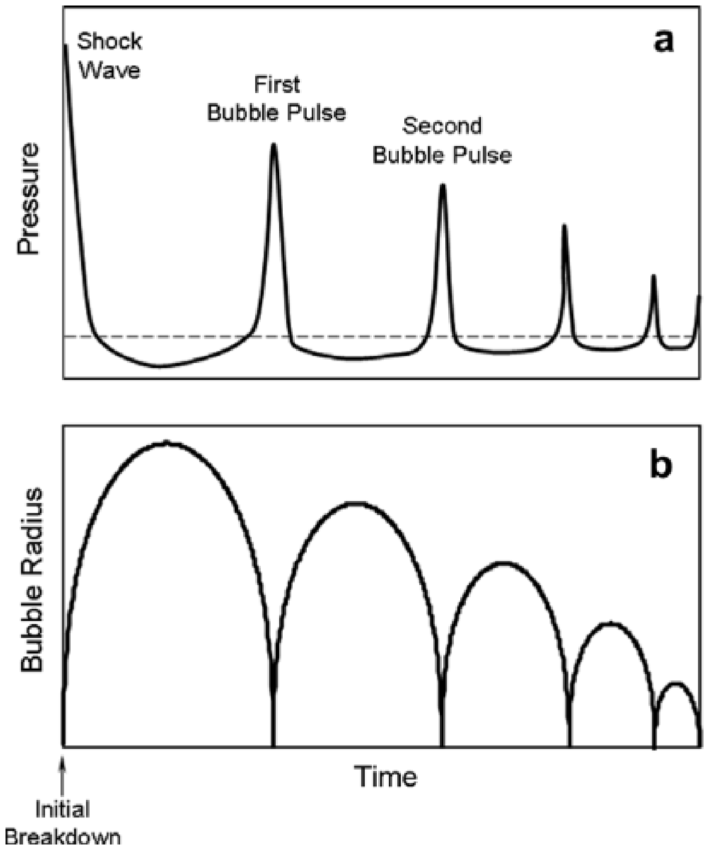


Figure 1.6: Pressure and bubble radius vs. time [9])

Patterns can be observed from the collapse of the cavitation bubble while interacting with a wall. As the shape of the bubble is distorted while it is approaching the solid boundary, the pressure force also has a significant effect on the boundary.

A general pattern is presented in Figure 1.6. Section **a** refers to the behavior of the pressure at a distance away from the bubble, while Section **b** describes the radius of the bubble at this same time.

The bubble radius, as mentioned in the earlier sections, follows the typical Rayleigh-Plesset behavior over time, especially with the first bubble period (from initial breakdown to the first collapse). Bubble radii in subsequent collapses likewise continue this pattern but at a steadily decreasing maximum radius. As there is shape distortion from the wall, the radius is typically more difficult to observe and quantify.

The pressure on the other hand reaches a maximum during the initial breakdown due to the shock wave created during bubble formation. This pressure steadily decreases, then increases again rapidly as the bubble reaches collapse. Likewise, the maximum pressure found a distance away from the bubble also decreases as the bubble continues to evolve until dissipation.

Müller et al.'s study [16] confirms the occurrence of these patterns through measurement via hydrophone attached at far field away from a cavitating bubble, and notices some correlation to the force exerted by this pressure as well.

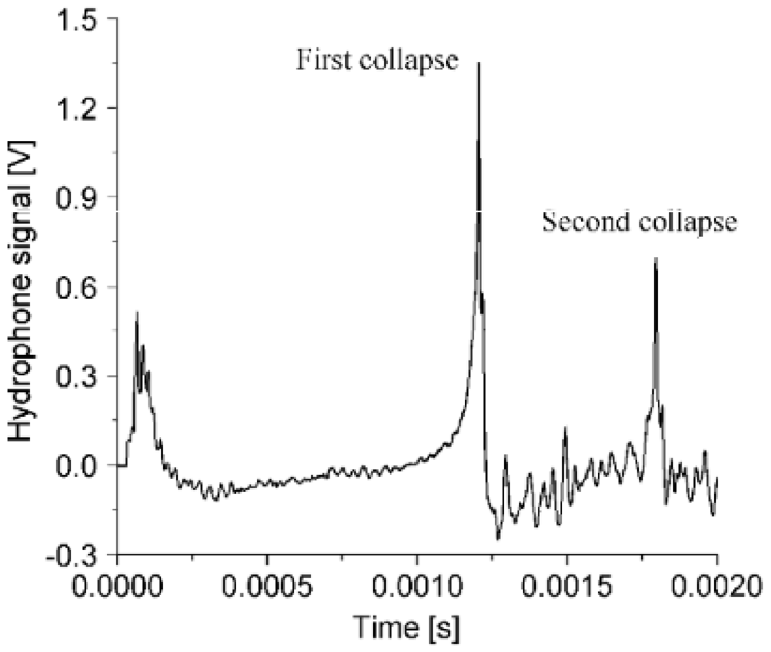


Figure 1.7: Pressure signal (V) vs. time, measured via hydrophone [16])

The pressure during the first bubble period (from formation to first collapse) is also observed to be changing relatively steadily, while more violent oscillations are seen in the second bubble period (from first to second collapse). This corresponds to the bubble growth and the distortion of its shape during this period [16].

The proximity of the bubble to the solid wall is also found to generate collapse patterns that have a noticeable difference in the force applied on the solid wall. Müller et al. identify three such patterns [16]:

- `Pattern 1` – observed when the bubble is farther away from the solid wall. Bubble shape in this area is minimally affected by the wall, as such, it remains relatively nearly spherical. It is typically observed that the peak of the first collapse is higher than the second collapse.
- `Pattern 2` – observed when the bubble is relatively nearer to the solid wall. The bubble will now experience more effects of the bubble and thus will also collapse in a more distorted shape. It is observed that the peaks of the collapses are relatively closer to each other in amplitude.
- `Pattern 3` – observed when the bubble is very close to the solid wall. The bubble breaks into a non-spherical shape after the first collapse, and attaches itself to the wall after the second collapse. A "sucking" effect is observed, as such the peak of the second collapse is observed to be higher than the first.

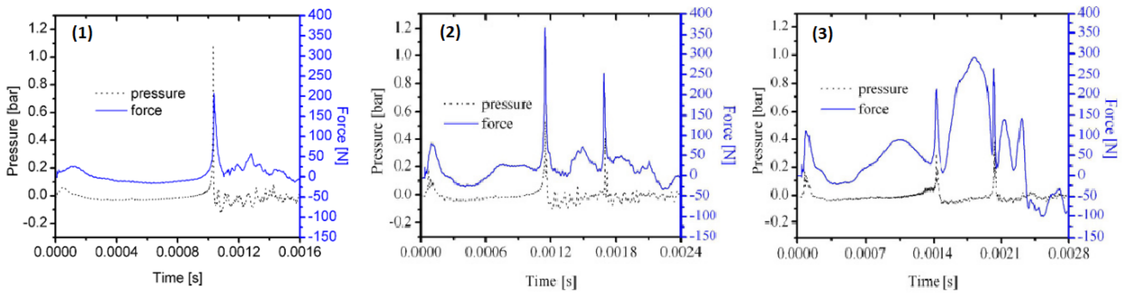


Figure 1.8: Waveforms of collapse Patterns (1), (2), and (3) [16])

These collapse patterns are compared in Figure 1.8.

1.2.5 Proximity Parameter γ

A parameter that is primarily used for investigating the cavitation behavior while interacting with solid walls is the proximity parameter, commonly denoted as γ. One of the earlier observations for the effects of γ is in the study of Shima, et al. [17], where it was found that the mechanism of bubble collapse is dependent on such a parameter, defined as

$$\gamma = \frac{L}{R_{max}} \tag{1.8}$$

where L is the distance between the bubble center (denoted by the crossing point of the electrodes) and the wall, and R_{max} is the maximum radius of the bubble. Subsequent studies use γ as a key parameter for understanding bubble behavior near a solid wall boundary [9].

1.3 Effects of Cavitation

1.3.1 Adverse Effects

Cavitation has often been of note as a physical process due to its destructive nature from the damage dealt by numerous re-entrant jets generated in a fluid medium. This behavior is particularly observed in the weathering of turbines and similar systems, as these involve the high-speed movement of water in order to function.

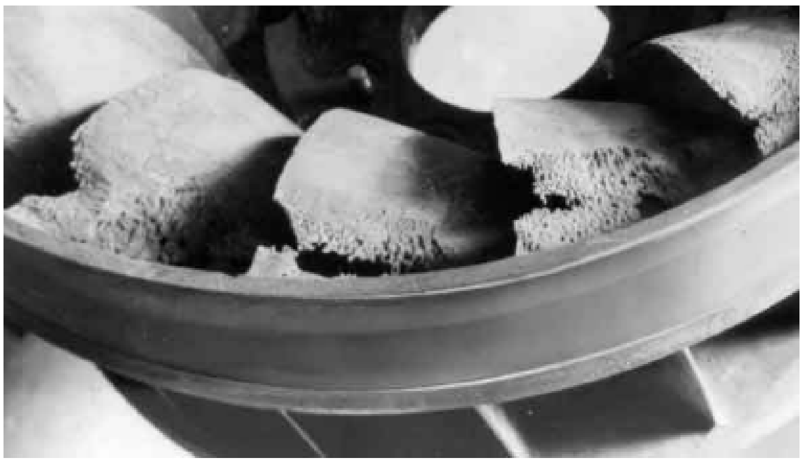


Figure 1.9: Cavitation damage to the blades of a Francis turbine [3]

The collapse of bubbles formed from cavitation has the potential to produce a high-amplitude pulse having pressures comparable to the ultimate and yield strength of some materials, which cause damage to the materials that make up turbines, pumps, or channels [18]. Numerous studies have been conducted through the years to evaluate the extent of damage which can be caused by cavitation, and as such, some rules have been developed from where engineers can estimate and mitigate thereafter the effects of cavitation in a particular system [3].

1.3.2 Beneficial Effects

On the other hand, cavitation has been used in a number of different applications due to the high-stress destructive nature of the microjets.

In the field of medicine, a wide range of applications were seen to use the concept of cavitation. The nature of cavitation activity in clouds of bubbles are studied in its potential to break kidney stones in a certain study [19], while another employs cavitation to play a role in inducing opening of the blood-brain barrier to facilitate treatment of central nervous system diseases [20]. The cavitation mechanism, and understanding its extent and how to control it, is important for fluid applications in a very small scale.

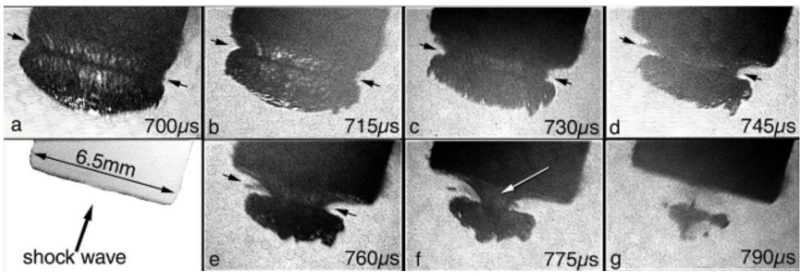


Figure 1.10: Cavitation applied to the surface of an artificial kidney stone [19]

Cavitation can also be used for wastewater treatment operations. Dular et al. [21] posit the use of hydrodynamic cavitation methods in conjunction with other existing ways to treat wastewater in their extensive study, as it is not yet widely used industrially. The methods describe aim to use cavitation to disintegrate organic molecules in the water which may be small enough to evade other treatment methods. Gonzalez-Garcia et al. [22] also review the effectivity of the treatment of wastewater, specifically the degradation of chlorinated compounds, using sonochemical treatment from acoustic cavitation.

2 Objective of the Study

The main goal of the study is to create an analysis tool that will observe and measure what occurs during cavitation and the interaction of bubbles with a solid wall. The main objective of this study then is to benchmark this tool by comparing single and multiple bubble experimental setups to assess the difference between these two.

In order to fulfill this objective, a modifiable experimental setup able to replicate single and multiple (in this case two bubble) bubble cavitation scenarios is constructed. Measurement equipment are used to obtain information about the evolution of the cavitation bubble or bubbles during its lifetime. From the data acquired with the use of these equipment, the analysis tool aims to extract relevant information and assist with the observations in order to differentiate these experimental setups from each other.

The rationale of this study is to improve upon previous related studies in investigating single and multiple bubbles, primarily through the introduction of software analysis to the process. Image recognition is used by the analysis tool in conjunction with the data acquisition from the PVDF sensor to make the process of measurement of the bubbles more efficient.

2.1 Related Studies and Literature

A number of previous studies related to what is aimed to be achieved in this study are assessed so as to evaluate the feasibility of fulfilling this objective.

Single and multiple bubbles have been investigated in a number studies throughout the years, and the study aims to integrate aspects from these into a viable setup which can accommodate such an analysis tool to differentiate notable aspects among single and multiple bubble cavitation scenarios.

The analysis tool also aims to lessen the complications of the measurements and to observe from these studies which are the most relevant measurements to consider.

2.1.1 Wang, et al., 2007 [9]

Wang and Chen [9] investigate the use of the piezœlectric PVDF film to observe impulsive forces that come from a cavitation bubble collapse near a solid wall. From this study, the use of a PVDF film sensor for this purpose is selected due to the material having good bandwidth, high sensitivity, and remarkable durability, making it a good option for collecting information about impulsive forces brought about by cavitation. The dynamics of the cavitation bubble is recorded alongside the PVDF sensor information for comparison.

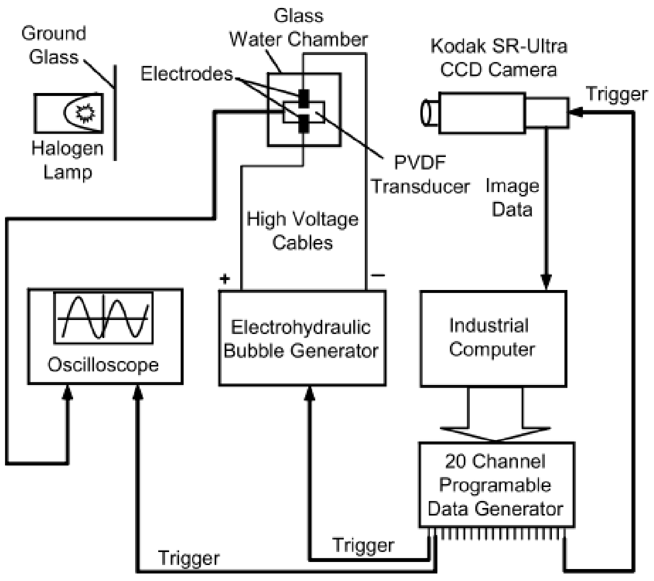


Fig. 2. Experimental setup.

Figure 2.1: Experimental setup of Wang, et al. (2007)

The experimental setup of Wang and Chen relies on two inputs, namely the PVDF signal and a CCD camera, to obtain pressure signals over the time of development of the bubble and to record its dynamics over this time. A singular cavitation bubble was generated by this setup through a high voltage spark (maximum charge of 10 kV) gap discharged by a 1.2 μF capacitor. The PVDF film used is of the model DTI-028 K/L (of thickness 28 μm and digitized through an oscilloscope at a maximum sampling rate of 2.5 gigasamples per second) while the Kodak high-speed CCD camera can record the bubble for up to 10,000 FPS. From each frame, the radius of the bubble is measured, in the following figure a 10 mm reference is included for manual measurement per frame. As such, the growth, maximum radius, collapse and deformation can be observed throughout this process.

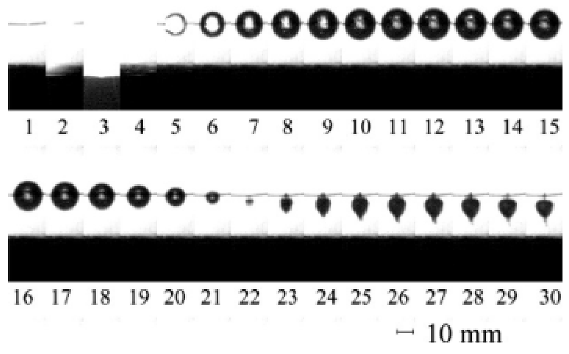


Figure 2.2: Bubble dynamics frame by frame, Wang, et al. (2007)

The proponents discern from the PVDF information some primary parts of the signal. The first part, which is the electrical discharge, is a large spike in the pressure followed by more than 1 ms of noise and ending in a drop in the pressure. This part denotes the effect of electromagnetic interference and the subsequent spark discharge brought about by the spark generation mechanism.

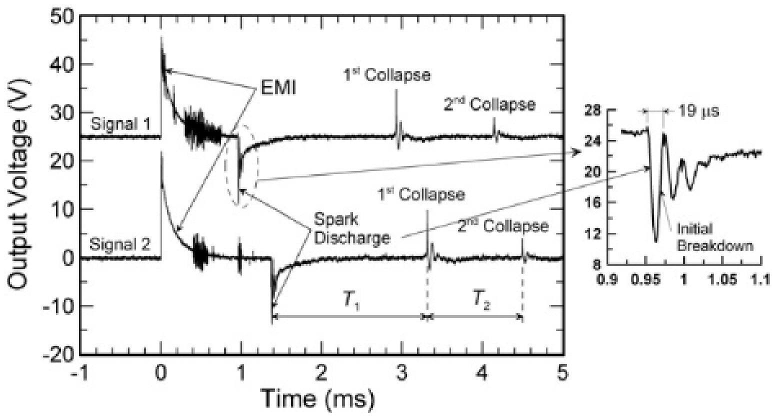


Figure 2.3: Sample PVDF pressure readings for two signals, Wang, et al. (2007)

The subsequent parts are the evolution of the bubble. A steady increase in the pressure waveform indicates the growth of the bubble up to its steady decrease in size, and then culminating into a sharp peak denoting the collapse. Further growth, shrinking, and collapses will occur, each with a peak at its collapse at a lower magnitude than the next. As shown also in the figure, the periods between collapses of the bubble (shown as T_1 and T_2) can be measured, with a replicable difference depending on the value of γ [9].

On the other hand, bubble radius also corresponds the same to the bubble pulses detected by the PVDF film as pressure changes. For each of the peaks in the pressure, the bubble radius is also at its smallest. This corresponds to the collapse of the bubble as it changes its shape throughout its evolution.

The scope of Wang and Chen's study is limited to observing single high voltage spark-generated bubbles, and both the first and second bubble collapse periods for bubbles within the range $\gamma = [0.14, 8]$.

2.1.2 Khoo, et al., 2009 [23] and Fong, et al., 2009 [5]

Khoo, Adhikari, Fong, and Klasebœr [23] studied the interactions of multiple spark-generated bubbles. Their experimental setup is described to be modified from earlier studies of Turangan et al. [24] and Lew et al. [25].

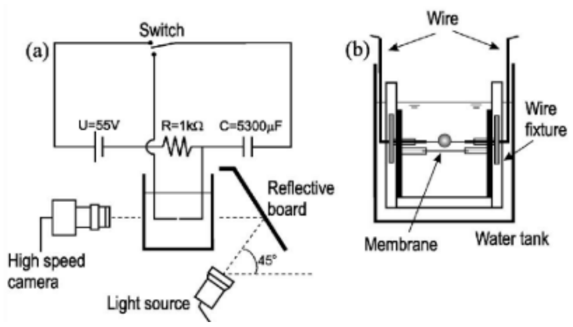


Figure 2.4: Experimental setup of Turangan, et al. (2006)

The spark generation mechanism is a relatively lower voltage (55 V) RC circuit with spark junctions created from fine copper wire (about 0.11 mm diameter). The wire crossings allow for sparks to occur underwater even at low voltages, which is why high voltage spark gaps such as used by studies from Wang et al. [9] and Buogo et al. [26] were no longer necessary; while these wires being thinner than the average size of the generated bubbles will ensure that as much as possible, the behavior of the bubble is not disturbed by them [25].

The modification done by Khoo et al. for this study is to extend the experimental setup for two and three spark-generated bubbles. In their scope, the effects of a solid wall are not covered, and all measurements are done from what is captured from a high-speed camera with a framerate of 12,500 FPS [24, 25].

From their measurements with two bubbles, at least four interactions were recorded: the formation of bubble jet toward the other bubble (for a small phase difference between bubble collapse times), away from the other bubble (when the

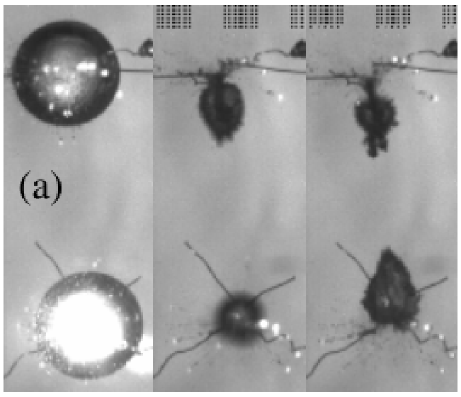


Figure 2.5: Bubbles jetting towards each other, as observed by Khoo, et al. (2009)

bubble collapses become completely out of phase), the ”catapult” effect (where a high-speed jet is created when the bubbles are at a small enough distance from each other), and coalescence (where two bubbles close enough to each other can fuse into a single bubble) [23].

A parallel study by the same authors [5] expands on these presented interactions and also investigated behaviors for more positions of three bubbles, again without the effects of a solid boundary wall. They are shown to have used a similar setup, with insulated wires as electrodes are placed at a distance apart and being held in place by support beams.

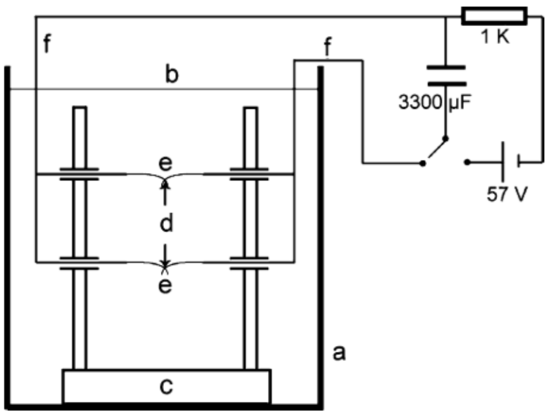


Figure 2.6: Experimental setup of Fong, et al. (2009)

From their experimental results and simulations, the authors constructed a graph of these interactions between bubbles dependent on their phase difference and distance from each other. In this case, the experiments recorded bubbles formed with similar sizes, and such, different bubble radii will exhibit a different phenomenon, which leads to the smaller bubble jetting away and the bigger bubble potentially breaking in half [5].

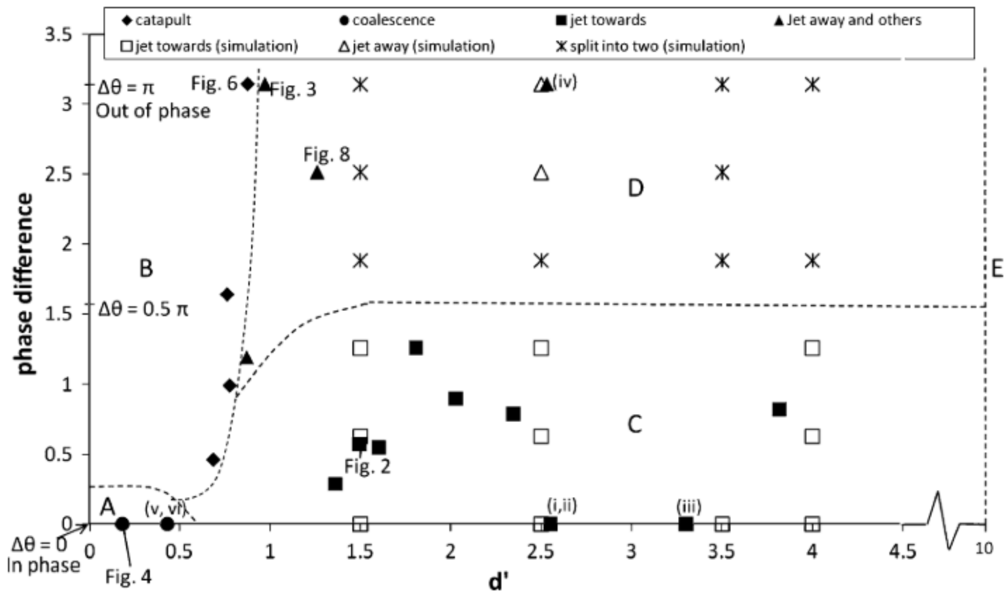


Figure 2.7: Diagram of two bubble interactions, as observed by Fong, et al. (2009)

It is important to note from this review that it will be possible to investigate multiple spark-generated bubble scenarios, but the effect of solid walls and the measurement of pressure on them are both outside the scope of the studies of these authors.

2.1.3 Goh, et al., 2012 [27]

The study of Goh, et al. [27] investigates how to generate the single bubbles with a relatively low input voltage and having consistent sizes with the following setup described.

The researchers use a 60 V input source, a four-part spark generation circuit, and specialized electrodes. The spark generation circuit, instead of being a simple RC circuit as used in other mentioned studies, comprises of a charging circuit, a discharging circuit, a storage circuit, and a sparking circuit. The storage circuit consists of two capacitors in parallel (4700 μF and 2200 μF) where the electrical energy is stored. The charging circuit takes the 60 V input to the capacitors through a 1 kΩ resistor, while the discharging circuit releases the energy through a 1 W, 4.7

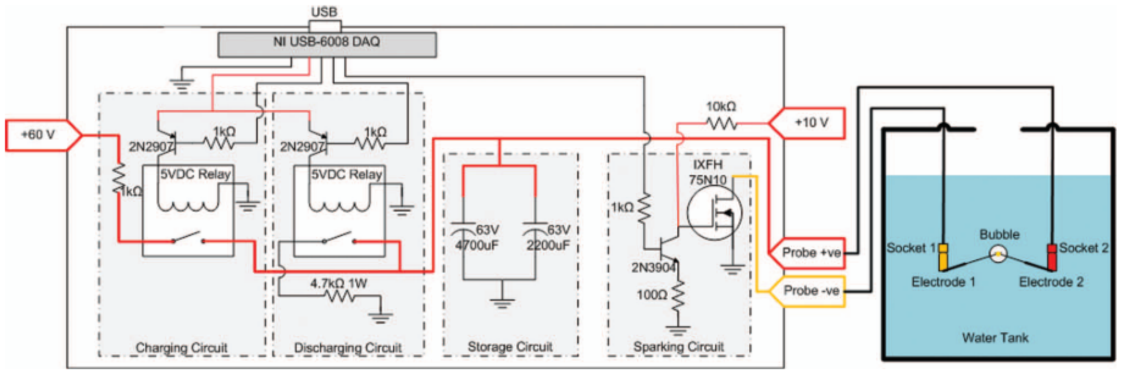


Figure 2.8: Experimental setup of Goh, et al. (2012)

kΩ resistor. Finally, the sparking circuit is controlled by a 10 V switch, from where the output of the capacitors is directed to an n-channel MOSFET rated at 75 A and 100 V which delivers the spark to the electrodes.

The electrodes, which are set up as easily replaceable components, comprised of a banana plug with 0.6 mm gauge wire, with a finer 0.1 mm diameter wire soldered unto it. The finer wires are the ones connected underwater to produce the spark. For this experiment, the dynamics of the bubble produced by the spark is recorded with a high-speed camera at 50,000 FPS.

It was concluded that the resistance of the electrodes, together with the resistance of the discharging circuit, can affect the maximum radius of the bubble. This then can be easily controlled through the monitoring of the length L of the electrodes to be used. The expected maximum radius, as determined experimentally, would be at the range of 3 to 5 mm.

2.1.4 Ji, et al., 2017 [28]

This study of Ji, Li, Zou, and Yang [28] investigates two spark generated bubbles and their interaction with a free surface while being underneath it. As such, instead of with a solid surface, the interaction of the bubbles are checked with the surface of the water the electrodes are submerged in. In this case, the bubbles are placed nearer to the water level of the tank.

The spark generation circuit in this case is a simple RC circuit, with a total capacitance of 5300 μF (from three capacitors arranged in parallel) and resistance of 1 kΩ. A high-speed camera with a framerate of 6,000 FPS records the evolution of the bubbles.

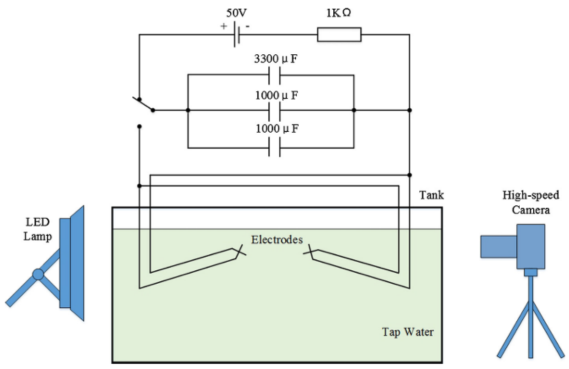


Figure 2.9: Experimental setup of Ji, et al. (2017)

From this study, the two bubbles generated can be made to have almost the same size. The interaction of these bubbles with the free surface, in the form of vertical surface jets, look similar to collapse jets toward a solid wall boundary. Similar phenomena have been observed here as well as from the previous reviewed studies with multiple bubbles [5, 23] such as coalescence and collapse jetting.

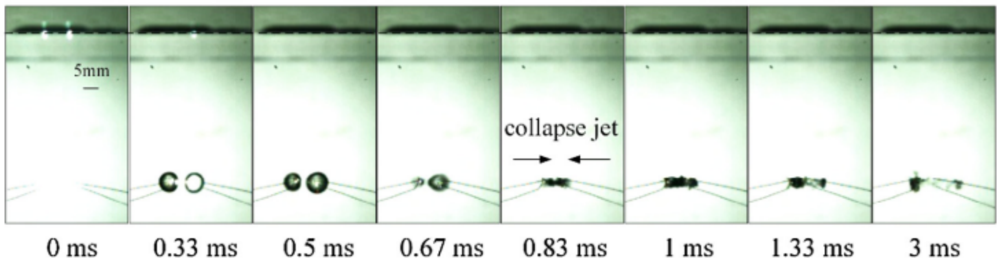


Figure 2.10: Bubble interactions, here especially bubbles jetting toward each other) as observed by Ji, et al. (2017)

2.1.5 Liu, et al., 2017 [29] and Luo, et al., 2019 [30]

Liu, Cui, Ren, and Zhang (2017) and Luo and Niu (2019) investigate in separate studies the interaction of two spark-generated bubbles. Their respective studies involve similar setups to that of Fong et al. [5], mostly differing on their respective parameters of capacitance, resistance, and input voltage.

Liu et al. focuses on the interactions between two bubbles with respect to a free surface (namely the surface of the water in the tank) as well as a solid wall (which is denoted to be the bottom of the tank), and these two distances are considered in their study as h_f and h_w respectively. As is common to these two studies, the

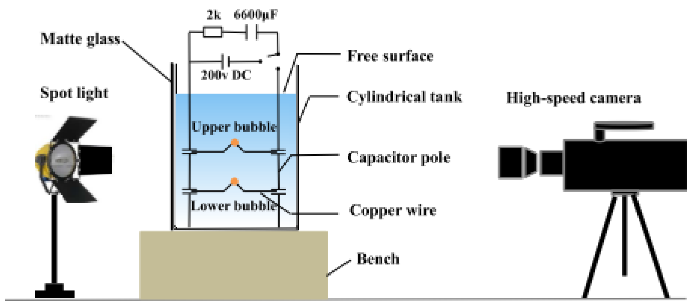


Figure 2.11: Experimental setup of Liu, et al. (2017)

bubbles are oriented vertically, one on top of the other. The spark generating circuit has a 200 V DC input, a resultant capacitance of 6600 μF, and a resistance of 2 kΩ [29]. As such, their experimental setup requires a relatively higher voltage input. The camera used was able to take images of the bubble evolution at a rate of 31,000 FPS [29].

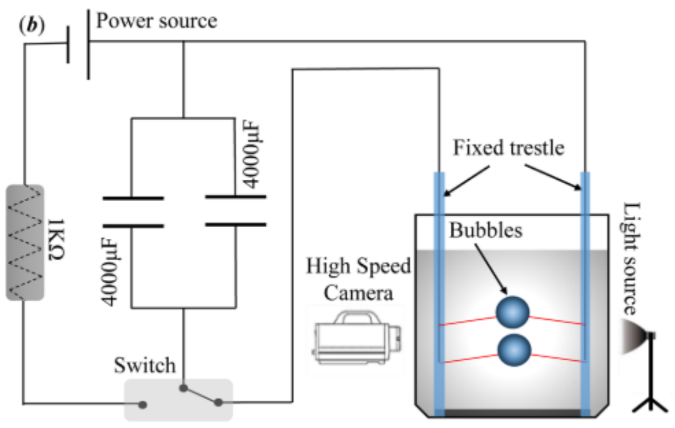


Figure 2.12: Experimental setup of Luo, et al. (2019)

On the other hand, Luo et al. compare the two-bubble interaction across two types of experimental setups, thus spark generation is only one of these types. (The alternate setup they compare with is with a laser-generated double bubble setup). The spark generated bubble setup powers its capacitors to 100 V, as an RC circuit with an equivalent capacitance of 8000 μF and resistance of 1 kΩ. The high-speed camera in use captures images at a framerate of 180,000 FPS [30].

2.1.6 Szala, 2017 [31] and Lv, et al., 2019 [32]

Studies by Szala [31] and Lv et al. [32] explore the potential of image processing for cavitation. Although both focus more on the macroscopic sense of cavitation effects, the state of the art of image processing usage as demonstrated here is a rationale for the same use in cavitation in the microscopic (bubble) scale.

Szala compares two types of proprietary image analysis software, namely ImagePro Plus and MetIlo, for recognizing images of specimen eroded by cavitation of water bubbles due to an ultrasonic horn. The steel specimen was observed after different exposure times to the horn while underwater, and each resulting specimen from these exposures was photographed and processed by the software for comparison of roughness parameters among other attributes found in the images [31].

Lv et al. also makes use of image processing on cavitation erosion exposed specimen, now particularly on the pits formed by cavitation erosion on a hydrofoil during cavitating flow. 8-bit resolution grayscale images of the specimen showing the pits are subjected to an edge detection method, namely the Canny edge-detection function, to recognize the shape of the pits that are in the images. Edges are recognized by this function, which is similar to the first derivative of the Gaussian function of the image, by assessing which value along the surroundings of each point in the image has an intensity which is equivalent to the maximum value [32].

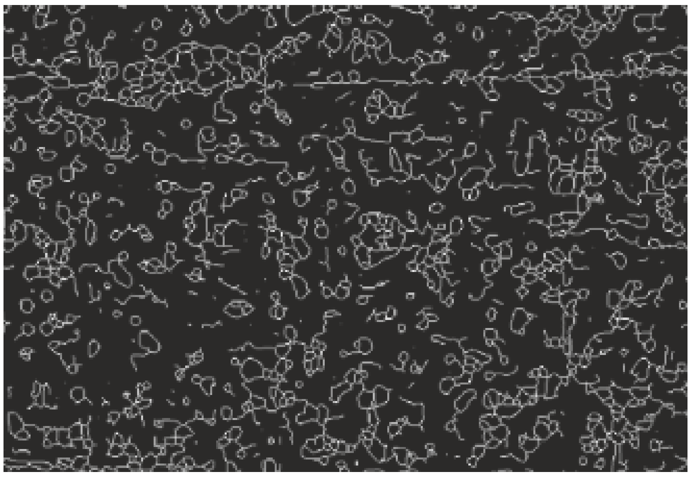


Figure 2.13: Canny edge detection of hydrofoil specimen of Lv, et al. (2018),

Shown here is an example resultant image from Canny edge detection from a specimen in the Lv et al. study [32] displaying contours of detected pits. A similar method will also be used to recognize cavitation bubbles for this masters thesis.

2.2 Scope of the Study

The scope then that will be covered by this particular study is described here.

A spark-generated bubble experimental setup will be used to test the cavitation bubble analysis tool prepared by the author, configured for both single and double bubble setups. The bubbles generated by the experimental setup are of sizes between 3 and 5 mm. The range of bubble size between multiple bubbles is aimed to be kept at around ± 3 percent.

The distance of the bubble centers, as measured from the junctions of fine wire electrodes, are kept consistently at about 8 mm. The distance of the bubbles from the wall are then experimentally determined from two positions: one nearer to the wall (5 mm away) and one farther away from the wall (8 mm away). These positions will allow for γ values equal or greater than 1, as the electrodes are kept at a safe distance away so as to not damage the PVDF film sensor.

This study will primarily be focusing on the effects of a solid wall boundary to the bubbles, and less so to inter-bubble behavior and interactions with a free surface (such as the surface of the water). Bubbles in free liquid (without any walls) are also not under the focus of this study. This is due to the fact that the pressure force inflicted by the cavitating bubbles is the primary point of comparison for this study vis-a-vis the maximum bubble size as determined by high-speed camera.

3 Experimental Setup

The experimental setup used for evaluating the cavitation analysis tool is detailed in this chapter. Fundamentally, the basic setup for spark-generated bubbles can be seen in Figure 2.1, loosely adapted from the experimental setup used in a previous study of Müller et al [16], which uses similar low-voltage spark-generation techniques as described in earlier review.

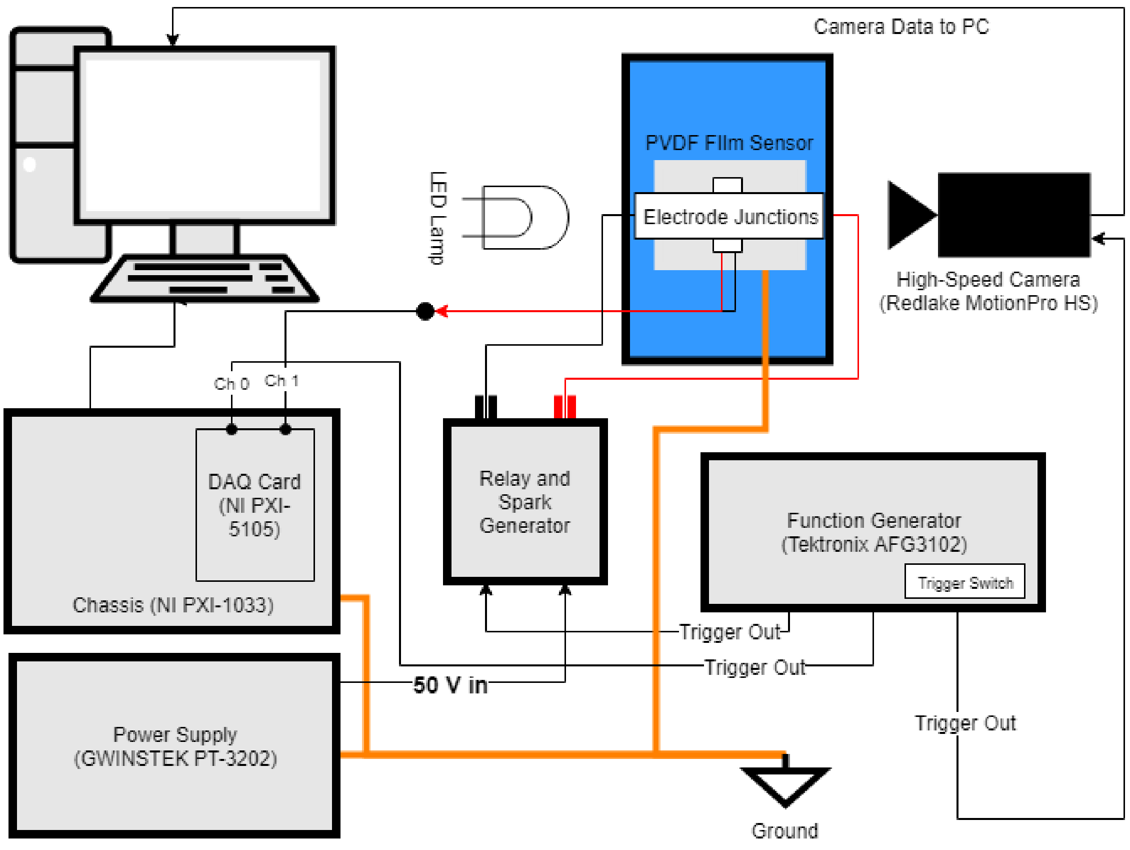


Figure 3.1: Experimental setup

The cavitation bubbles for this experiment are spark-generated, that is, an electrical pulse discharged by a capacitor is sent through a junction (or junctions for multiple wired) of conducting wires submerged underwater. The ensuing spark creating an expansion of the liquid, wherein cavitation will occur. A build-up of charge in the capacitor that is high enough can lead to the formation of bubbles which can be sufficiently analyzed through image capture or with other methods.

This will allow the tool to collect measurements from this setup, and from automatic calculations, will quantify and distinguish the behavior of bubbles in the presence of a solid wall.

In order to minimize the effect of electrical interference from the capacitor setup to the reading of the PVDF film while underwater, proper grounding of all metal components must be observed.

3.1 Bubble Generation

Bubbles are generated from within a liquid medium and are observed by the measurement equipment. In this case, the medium is tap water, maintained at about 26 ± 1 °C. A transparent tank (dimensions 50 cm x 25 cm x 30 cm) is filled more than halfway with the liquid medium, and an additional platform was added to accommodate the pressure sensor and steel (solid wall) surface and put it in the same level as the high-speed camera. The electrode junction block is submerged in this tank just above this platform.

The electrode junction block setup is slightly modified for when single or multiple (in this case, double) bubbles are considered.

3.1.1 Electrode Junction Block

Bubbles are created from the electrode junction block by using a pair of electrodes crossing at a single point underwater where a short circuit will occur. The charge buildup of the capacitor will be discharged, creating a pressure change at this crossing point where a cavitation bubble will evolve. A stronger charge buildup, as provided by the initial voltage across the capacitor, will create a larger bubble. The crossing point can also then be moved, nearer or farther from the wall being investigated. This allows the experimental setup to cover distances from the wall at 5 mm and 8 mm.

The electrode junction is composed of two electrodes, each connected to the Spark Generation block. Each electrode is composed of a thicker gauge wire (about 2 mm) with a finer gauge wire (about 0.1 mm(soldered onto it, screwed in place to the outputs of the spark generation block and held in place right above the PVDF film sensor by a hanging test stand made of hard plastic. For double bubble setups, the electrode will be composed of two fine gauge wires soldered on to the thicker wire on a singular point, slightly separated at this end in order to allow distance between the fine wire crossings.

Among other configurations of the electrode junction block (such as separating the fine wires further through a Y-shaped arrangement of thick wire which had 3 solder points; or separating the wires soldered onto different thick wires which were fastened to the relay output which was split into two), this configuration has worked out the best in terms of producing consistently sized bubbles. It is presumed that it is because the aforementioned configuration only has one soldered junction point before splitting the discharge on two wires, which leads to less random points where the current will decide to flow. The fine wires are ensured to have the same resistance through them (measured at 0.23 ω, and their length are maintained to be the same at all times so as to minimize the effect of wire resistivity per unit length.

3.1.2 Spark Generation Block

The spark generation block is an RC circuit similar in structure to the spark generation circuits reviewed in other studies. This block is governed by relays, having also one external switch that enables or disables the charging of the capacitors, which have a total capacitance of 4700 μF.

There are three sets of external interfaces: firstly, the trigger input, which has a twist-on BNC connector for attachment to the function generator (see 3.5); the second being the voltage input with two connectors (for positive and negative ends) for charging the internal capacitors; and the last being another pair of connectors reserved for the output to the electrode junction block, responsible for the spark. These connectors are fixed to the ends of the electrode thick wires via screw.

3.2 Camera

A high-speed camera captures the evolution of the bubble during the measurement. The camera used for this study is a Redlake MotionPro HS Series CCD Camera, similar to this one pictured.

This camera is set to capture photos at a frame rate of 10,000 FPS. The camera is configured to capture frame by frame images at a size of 148 pixels by 100 pixels, where the analysis tool can distinguish the bubble from the rest of the medium and

Figure 3.2: Redlake MotionPro HS Series Camera [33]

measure the parameters which can be seen from these images, i.e. bubble radius, distance of bubble from the wall, among others.

The time it takes for the evolution of the bubble, from generation to collapse, can be determined by the frame rate of the camera in frames per second, which is known for any particular high-speed camera. For each experiment trial, the tool can also execute the same analysis across different trials of the same experimental setup parameters, tabulating the value of each measurable parameter and allowing for the averaging of the bubble behavior for a particular voltage setting or distance from wall as shown in the figure.

For single bubbles, as measured by the analysis tool, it is found that the input voltage of the spark generation block influences the maximum radius in a linear fashion. This radius can be expressed as $R = C_1V + C_2$ for an input voltage V, where it is estimated that $C_1 = 0.1373$ and $C_2 = -2.4629$ for a regression fit (R-squared) measure of 98.67 percent. These measurements are consistent to what can be seen from the images of single bubbles created by this experimental setup.

To enhance the capability of the camera to produce images of the bubbles formed in the experimental setup, a powerful white LED light focused with a lens is directed at a straight line with the camera lens through the water tank, illuminating the view of the camera.

3.3 PVDF Sensor

The impact force dealt by the generation and collapse of the cavitation bubble is then also measured by a polyvinylidene fluoride film piezœlectric sensor. Through the piezœlectric effect, materials such as polyvinylidene fluoride can store electrical

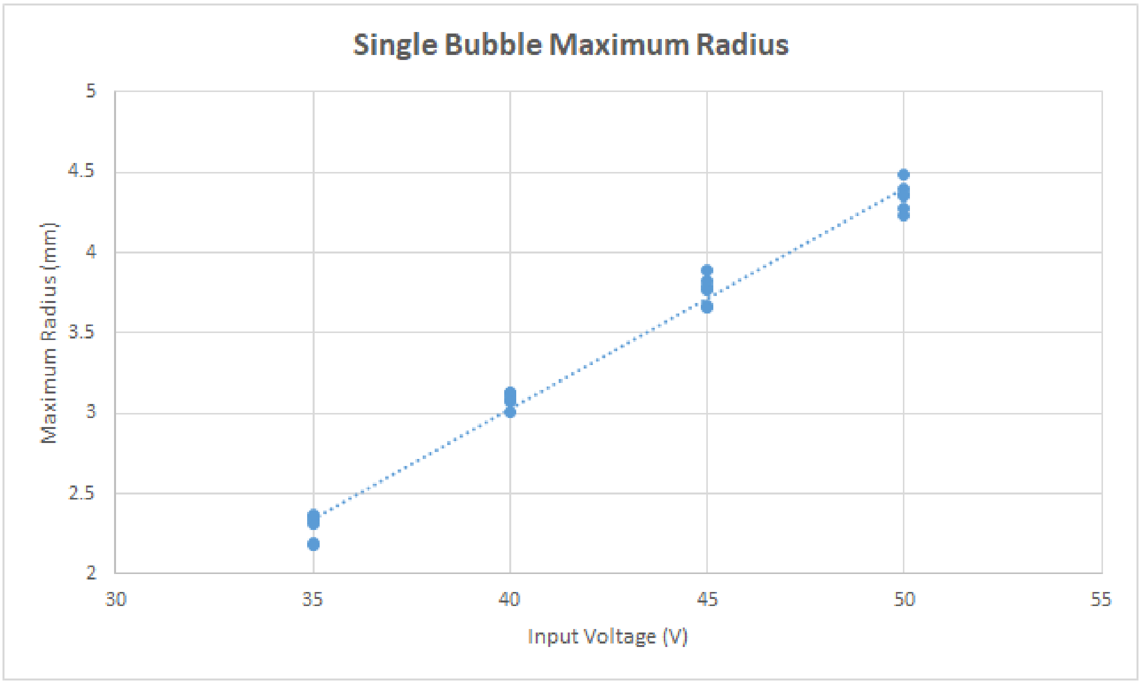


Figure 3.3: Calibration of bubble maximum radius versus input voltage

charges within it upon being subjected to mechanical stress, essentially converting force into electrical energy [34].

As such, the amount of charge collected from the PVDF sensor after the events of cavitation can be seen as a voltage value from an equivalent impact force. As the force applied by the cavitation bubble on a wall (which can be equivalent to the propeller or turbine or the like) must be analyzed as well, the PVDF piezœlectric sensor is a low-cost way of determining this. Polyvinylidene fluoride and its copolymers are favored for fulfilling the task of measuring pressure as it is a lighter, more flexible material, and is relatively easier to process [35].

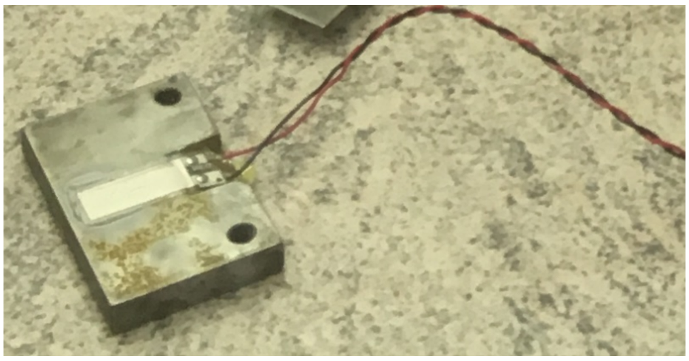


Figure 3.4: PVDF sensor mounted on metal platform

The DTI-028K PVDF film, with a total thickness of 28 μm and nominal capacitance of 1.38 nF [36], is mounted onto a metal platform, which also represents the surface of the solid wall with which the cavitation bubble is interacting. In order to minimize effects of electric interference from the spark generation circuit, this platform is connected to the common earth ground with the rest of the experimental setup. The PVDF film is glued with ethyl-2-cyanoacrylate superglue onto the platform, with a layer of tape and a two-component epoxy compound insulating the metal contacts, as shown in the figure. The area where the contacts are on the platform is also slightly tapered such that the film surface will be perceived as flat as possible. This entire sensor setup is placed under the short-circuit junctions, where it will act as the wall with which the bubbles generated by the junctions are interacting.

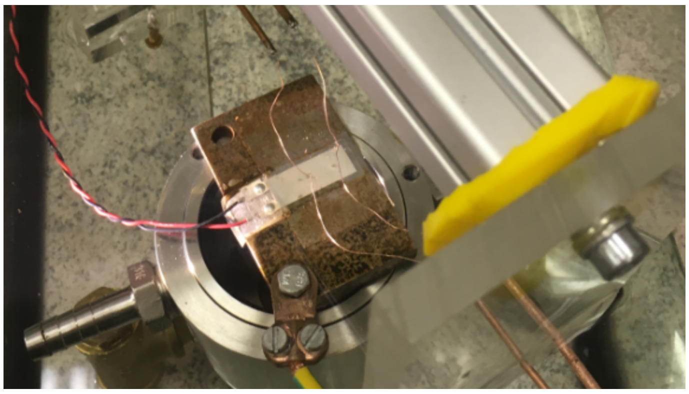


Figure 3.5: PVDF sensor submerged under short-circuit spark generators

The PVDF sensor is aligned at an angle near 45° so that for double bubble configurations, both bubbles will be visible to the camera, with the small distances between the bubble assumed to have minimal contribution to the size of the bubble seen on the image.

While it is submerged in water beneath the spark-generating probes, is then connected to a DAQ device, the National Instruments PXI-5105 12-bit Digitizer mounted on the NI PXI-1033 chassis. The data logging is achieved with the proprietary SignalExpress software also provided by National Instruments, as the PXI-1033 is connected to an external computer via PCI card. The digitizer can record data for a speed of up to 60 megasamples per second.

The start of data capture by the PVDF film is also synchronized to the rest of the circuit, as such, it will be possible to determine the evolution of this impact force from the bubble over time. Peaks in these data readings signify the highest force readings during the lifetime of the bubble.

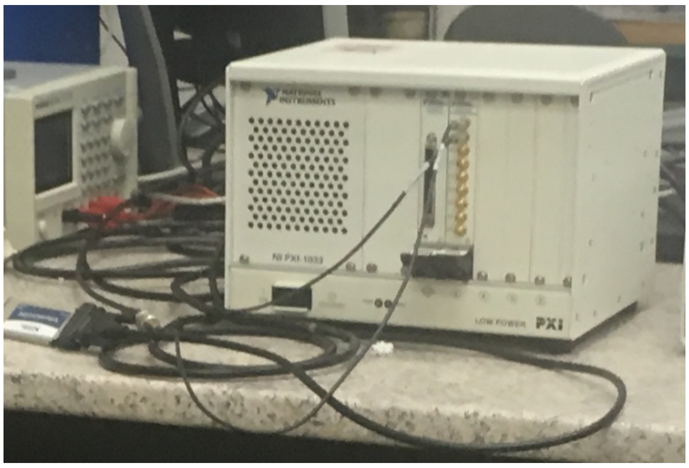


Figure 3.6: NI PXI-5105 Digitizer in NI PXI-1033 Chassis

The aim of the tool then is also to consolidate the PVDF sensor force data (in newtons) with the high-speed camera image data, and also apply digital signal processing to differentiate situations where there are different scenarios (more bubbles, bubbles forming closer to the walls, etc.) in the experimental setup.

3.4 Calibration

Before using the PVDF sensor, it must be made sure that the sensor has undergone calibration. Using Hujer's study [37] on the calibration of the PVDF sensor, the voltage reading can be converted into an equivalent value of force.

The measurement of this equivalent force is done by obtaining the conversion cœfficient from volts to newtons from the ball drop method. This involves measuring the height of a ball dropped on top of the PVDF film, and its subsequent rebound height. In this way, the force inflicted by the ball can be calculated for a corresponding voltage response of the sensor.

For this particular set of PVDF sensors, this value is determined to be at an average of 0.00771 V/N, calculated at a standard deviation of 0.0048 percent in 20 trials. This is at a value less than the nominal value of 0.013 V/N for a DTI-028K PVDF film sensor used in a similar study [9] as tape was added to the sensor to protect it from damage or effects from aggressive cavitation bubbles.

As can be seen in Figure 3.7, the voltage has strong linear correlation with the force on the PVDF film. From here, the calibration of the equivalent force for a particular voltage is realized according to the study of Hujer et al. [37]

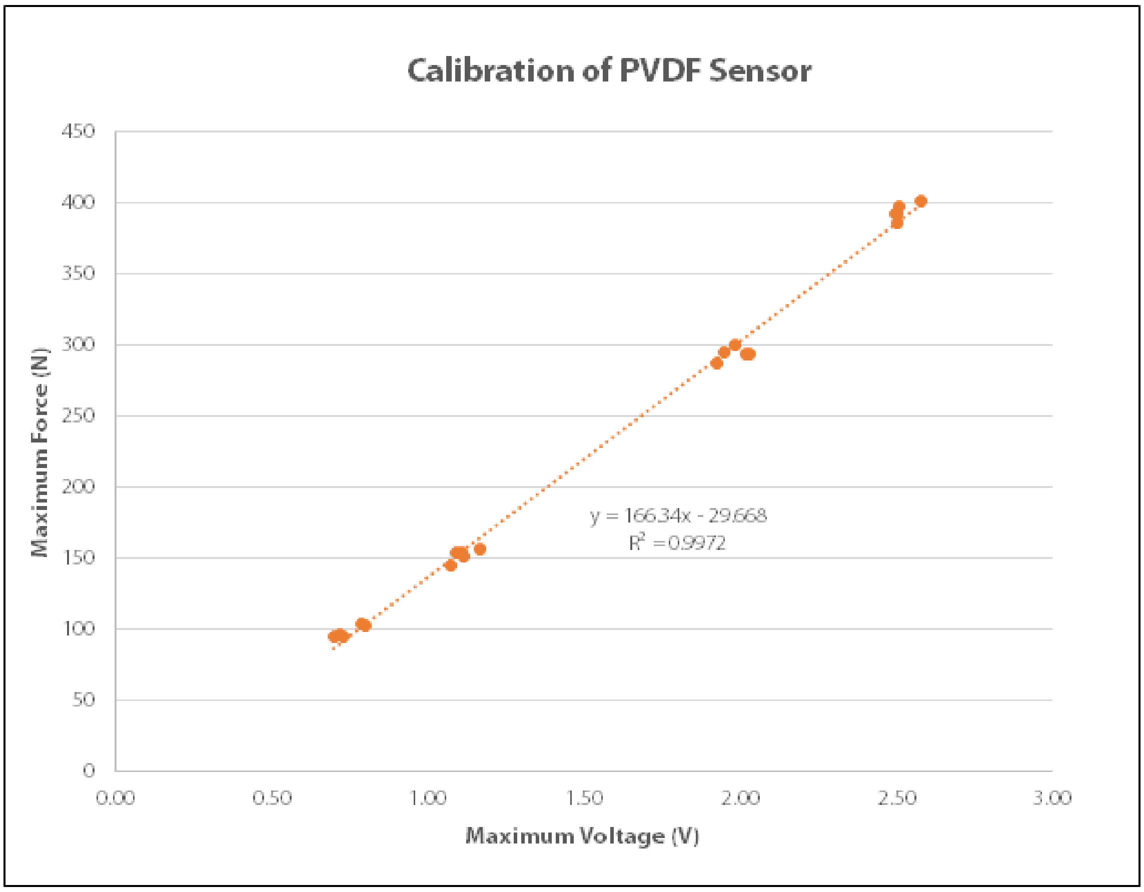


Figure 3.7: PVDF Calibration Results

3.5 Synchronization

All the elements must activate at the same time so that it is ensured that all data gathered will correspond to the same observed time frame.

The timing of the electrical spark, a pulse function, is determined by the function setting of a signal generator (the one used in this setup is the Tektronix AFG3102). This trigger signal is then synchronized with an external physical switch accessible from the function generator.

One output of the main trigger is connected to the camera. From the function generator, the triggering signal for this output is a continuous pulse train after a delay of 3 ms, enabling the camera to take images at a frequency of 10 kHz matching its framerate. The camera is set by its software to capture 140 frames at this rate.

Another output is connected to the spark generation block as its switch. The triggering signal is a simple pulse function set at an amplitude of 5 V, without any delay. This output is the switch for the relays in this block to discharge electricity on its output probes.

The third output, placed at the non-modifiable trigger output of the function generator, is connected to the DAQ block. As the DAQ would only need to know when the change of the voltage occurs (at positive edge) and the voltage threshold is modifiable on the SignalExpress software, there is no need to denote a specific voltage output for this trigger. The DAQ software is then set to have a 5 ms delay after the trigger in order to get the important readings, as in this experimental setup, bubbles are found to start developing about 6 ms after sparks.

4 CavTools Analysis Tool

This chapter details the analysis tool itself, which allows for the automation of radius measurement and consolidation of the PVDF measurements for comparison and analysis of the single and double bubble scenarios to be studied.

4.1 Procedure

In order to use this cavitation analysis tool, an executable file, `CavTools.exe`, is simply run in the computer. It is found only to function when the requisite DLLs are also in the same directory as the executable, as it is in the provided copy of the executable attached in this thesis.

An Open File Dialog opens once the executable is run, asking for a TIFF file (`.tif`). One of the camera images in a directory should be selected (typically `ImgA000000.tif`, but it should not matter which one is chosen).

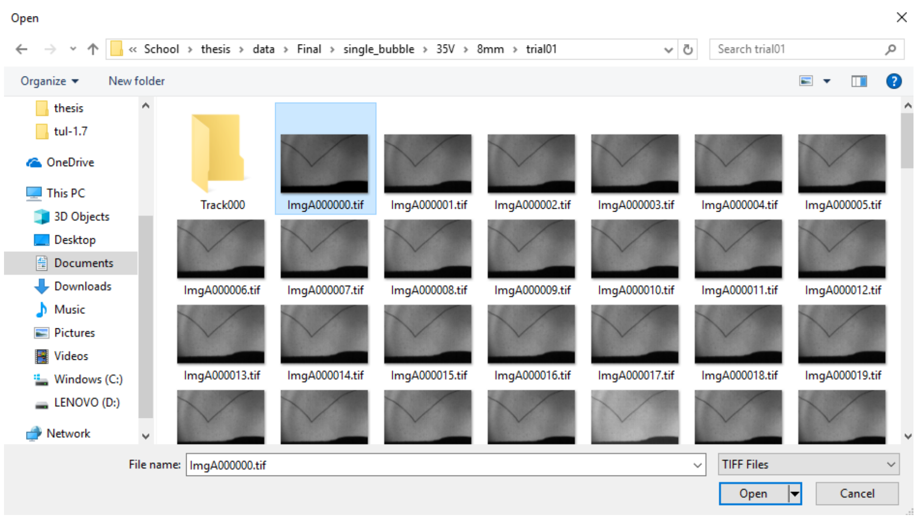
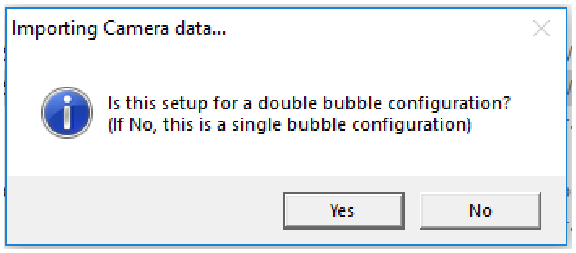


Figure 4.1: Selecting a camera frame in CavTools

The data in the working directory of a particular test run must be arranged in the following configuration. An example trial folder (`trialXX`) is given, and its contents are based on the actual output of the camera when a set of camera images are saved:

- `trialXX`
  - `ImgA000000.tif`
  - `ImgA000001.tif`
  - `ImgA000002.tif`
  - `...`
  - `pvdf_data.csv`

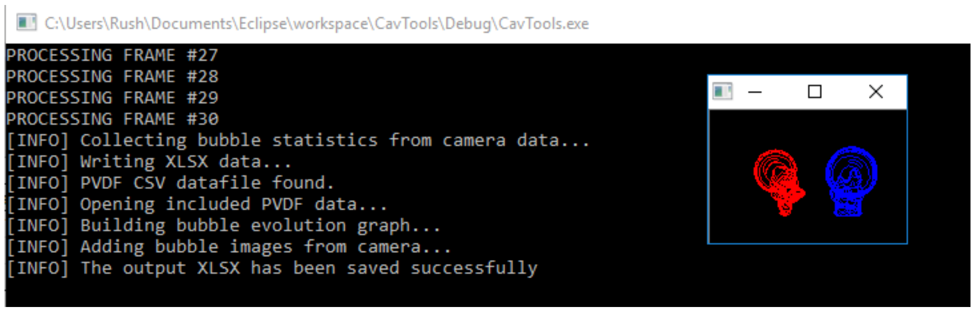
The analysis tool will then ask the user if the dataset being analyzed is of a double or single bubble setup. A proposal in the extension of this software in succeeding versions is to instead set up a graphical user interface with modifiable controls so that the distance from the wall and other such parameters will also be easily controlled.



Is this setup for a double bubble configuration?
(If No, this is a single bubble configuration)

Figure 4.2: Denoting either single or double bubble setup

The console will display the progress of the tool in analyzing the TIFF images and the CSV file of data obtained from the PVDF. Informational messages prefaced `[INFO]` are shown with the particular step the analysis tool is in.


C:\Users\Rush\Documents\Eclipse\workspace\CavTools\Debug\CavTools.exe

```
PROCESSING FRAME #27
PROCESSING FRAME #28
PROCESSING FRAME #29
PROCESSING FRAME #30
[INFO] Collecting bubble statistics from camera data...
[INFO] Writing XLSX data...
[INFO] PVDF CSV datafile found.
[INFO] Opening included PVDF data...
[INFO] Building bubble evolution graph...
[INFO] Adding bubble images from camera...
[INFO] The output XLSX has been saved successfully
```

Figure 4.3: Example outputs at the end of a run

At the end of the run, an image displaying the contours the image recognition algorithm recognizes as the bubble will be displayed, superimposed throughout the development of the bubble in time. Single bubbles are portrayed with red color, while double bubbles are individually portrayed in red and blue color.

In the end of this process, an XLSX output file (`data_output_xlsx_graph.xlsx` is created in the same directory as all the other files. This spreadsheet, accessible with recent versions of Microsoft Excel, is composed of the following sheets displaying information about the bubble:

- `Statistics` – contains statistics from the camera image information.
- `Calculations` – contains parameters calculated from the PVDF data based on the estimations made from the camera images.
- `Camera` – contains the camera data parameters in detail, tabulated for each frame considered during the evolution of the bubble.
- `PVDF` – contains the tabulated PVDF data which corresponds to the same time period obtained in Camera.
- `Bubble Evolution` – contains a graph juxtaposing Bubble Radius (as obtained from Camera) and the pressure signal (as obtained from PVDF).
- `Bubble Dynamics Images` – contains a frame-by-frame display of the camera images within this same time period in Camera and PVDF, with an equivalent time stamp for each frame.

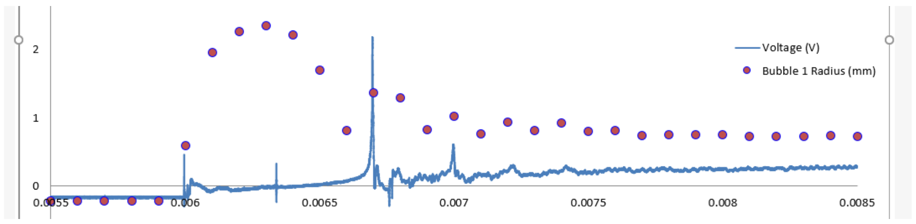
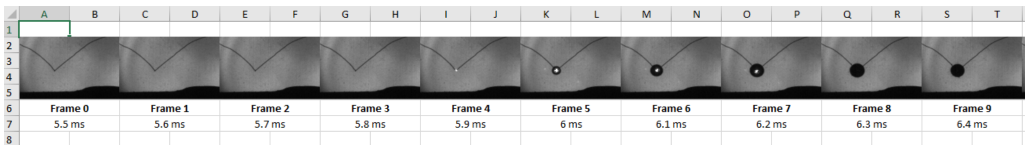


Figure 4.4: An example of Bubble Evolution graph



Frame 0	Frame 1	Frame 2	Frame 3	Frame 4	Frame 5	Frame 6	Frame 7	Frame 8	Frame 9
5.5 ms	5.6 ms	5.7 ms	5.8 ms	5.9 ms	6 ms	6.1 ms	6.2 ms	6.3 ms	6.4 ms

Figure 4.5: An example of Bubble Dynamics Images sheet

4.2 Programming Environment

The environment used for creating the analysis tool is described in this section.

4.2.1 Language

The analysis tool is coded using the C++ programming language. Building off of Dennis Ritchie's C language, C++ was first developed in 1979 primarily through the addition of the functionality of classes [38]. Classes are an important aspect of Object Oriented Programming, wherein similar entities known as "objects" get their characteristics from these classes [39]. Classes are constructed as a framework from where objects can be created, enabling the programmer to create more instances of similar such objects. As such, it is seen to be a good candidate for accomplishing the tasks that the analysis tool is built for - for analyzing multiple forms of data, passing them around throughout the program, and creating similar instances of such data. There are many available libraries as well for the modular tasks needed for the analysis tool, primarily image recognition (through `OpenCV`) among others.

4.2.2 IDE

The IDE used for developing this analysis tool in C++ is Eclipse, a freeware, open-source software development kit produced by the Eclipse Foundation. This programming environment was selected for its wide availability, ease of user interface, and ability to integrate with third-party libraries such as OpenCV.

4.3 Image Recognition

The image recognition is performed by the OpenCV library. OpenCV is an open-source library used for computer vision applications, particularly for a computer to process images and extract meaningful data from them [40]. This includes operations such as recognizing objects or part of objects, determining their 2D or 3D shape, or associating meaning to some image data. OpenCV contains more than 500 functions constructed in C/C++ [40].

The particular functions most used in the image recognition aspect of the CavTools analysis tool is `findContours()`, a function used to find contours in an image. A contour is defined as a figure drawn by a line, with such a line drawn to separate two contrasting colors on each side. To perform this function, a threshold is applied to each image first, converting it into an entirely black and white image on a certain threshold level. All pixels of the image which have a color value less than this threshold are zerœd out, while all pixels which have a higher color value are set to the maximum color value (255, 255, 255). As such, a recognizable contour will be drawn around the bubble. Functions such as getting contour area, perimeter, and others are then used once this contour is found.

4.4 The `CavTools` Library

The `CavTools` library is a simple software interface built to read the two sources of information from the experimental setup. It expects a set of images from the high-speed camera and a table of CSV information from the DAQ block in a specific configuration. From here, `CavTools` will analyze these inputs and prepare on its own a spreadsheet containing these information, the bubble images in sequence, and a graph comparing the bubble maximum radius and the pressure on the solid wall.

`CavTools` consists of the following components, separated according to their function in the bubble analysis process:

- `bload` – contains functions pertaining to the loading of bubble images (in TIFF format) onto the analysis tool.
- `bread` – contains functions, primarily from the OpenCV library, for detecting the bubble in each frame of the image
- `bcomps` – contains functions for the computation of various parameters and measurements from the camera and PVDF data
- `bsave` – contains functions for consolidating the outputs of the analysis onto an easy-to-read format (as an Excel XLSX spreadsheet).
- `bwin` – contains functions for window interactions with the user. Can then be extended as containing functions for the GUI of the tool in future versions.

The `CavTools` main program itself (`CavTools.cpp`) for now performs all of the analysis pulling from the functions available on these components. On future versions, this can comprise a new component by itself.

The preliminary application with the `CavTools` library implementation, as of now numbered as version 1.3, performs the basic functions as described in the components.

4.5 Data Output and Plotting

The `SimpleXlsx` Library developed by Pavel Akimov and Alexandr Belyak [41] is an open-source freeware C++ library used to interface with Microsoft Excel to create the data storage and graph plotting used by the tool to present the measurements.

Microsoft Excel is used as an important component for the analysis tool, due to its ubiquity in use with common personal computers and its ability to compute through large sums of data using cell-based formulas for executing certain tasks. As such, these functions were deemed important for the analysis, coupled with the powerful ability of creating presentable graphs for investigating trends in the data.

As some functions are calculated via Excel formulas, modifications can also be made for correcting mistakes of the algorithm of the image recognition, for instance, in estimating collapse times (see 5.2).

5 The Cavitation Bubble Dynamic

Observations on the dynamics of the cavitation bubble, as determined with the help of the analysis tool, are to be described here. The initial testing trials are described, followed by the parameters measured by the tool from the camera images and the PVDF sensors. From here, the nature of the cavitation bubble dynamic and some trends that occur throughout the test runs are taken note of.

5.1 Test Conditions

The experimental run consists of two phases - measuring single bubbles and measuring double bubbles. The aim is to use the input interpreted from single bubble waveforms and shape structures in order to make sense of those that will be obtained from a double bubble setup. As the conditions of the single bubble are covered in literature in more detail, the single bubble measurements are to be the main basis with regards to detecting patterns occurring in the double bubble measurements.

As noted in the experimental setup, the single bubble measurement phase considers two variables, namely the input voltage of the spark generating block and the distance of the electrodes from the wall. Four input voltages (35, 40, 45, and 50 V) and two distances (5 mm and 8 mm away from the wall) are considered. Each particular configuration of these variables have three trials for precision. This lends to a total of 24 available test points for single bubble characteristics.

The double bubble measurement phase considers bubbles created simultaneously with the experimental setup, with the spark generation circuit powered with a 50 V input, as this has been discovered to make the most visible bubbles while reaching the limit of the power supply output. Three trials are taken for two configurations: 5 mm and 8 mm away from the solid wall.

5.2 Bubble Parameters

The parameters measured by the tool are either deduced through the image analysis of the frame-by-frame bubble evolution, or from the output waveform of the PVDF sensor. The latter of these are obtained with assistance from early estimations made from the image data.

5.2.1 Parameters from Camera Data

The primary parameters included in the output spreadsheet which are obtained from image processing are:

- `Position` – the position of the bubble at a particular frame. X and Y values are obtained for the bubble position.
- `Area` – the area of the contour which the analysis tool detects as a particular bubble. Given through a function within OpenCV.
- `Perimeter` – the length of the perimeter of the contour which the analysis tool detects as a particular bubble. Given through a function within OpenCV.
- `Radius` – the radius of the bubble as if it were treated as a circle (as it is for most of the bubble behavior pre-collapse). Obtained by applying the method on how to get the area of a circle: $R = \sqrt{A/\pi}$.
- `Circularity` – the measure of how near the current bubble shape is to a circle. Given by comparing the area measurement of a circle to its circumference (perimeter). A circularity of 1 denotes a perfect circle, while higher values of circularity correspond to shapes further away from the circle (chiefly due to shape distortion).
- `Distance from Wall` – the measure of how near a bubble is initially to the wall. This value gets smaller throughout the lifetime of the bubble as it tends to move nearer to the wall. (In the current version, this parameter is currently deprecated as the wall detection is still being tested, and as such, γ is calculated directly with the distance from the wall (5 mm or 8 mm) divided by the `Radius` as described here.)
- `Gamma` – the proximity parameter γ, given as L/R_{max} where L is the distance from the wall and R_{max} is the maximum radius. (In the current version, this

parameter is currently deprecated as the wall detection is still being tested, and as such, γ is calculated directly with the distance from the wall (5 mm or 8 mm) divided by the `Radius` as described here.)

- `Estimated Rayleigh Collapse Time (ERCT)` – the Rayleigh collapse time of the bubble. This period is taken from the time where maximum radius is achieved up to the time where collapse occurs.
- `Estimated Time x (ETx)` – the measure of the specific time where events occur in the waveform. Four are detected in particular: ET0 (denoting the start of the spark), ET0_5 (denoting the time the bubble has reached its maximum radius), ET1 (denoting the time for the first bubble collapse) and ET2 (for the second bubble collapse).
- `Conversion Factor` – the measurement with which to convert the voltage into readings of pressure force. Obtained from calibration (see Section 3.4).

5.2.2 Parameters from PVDF Data

The following data are obtained from the PVDF signal, with assistance from the estimations obtained from the camera. This is done to ensure that the data between the two sources are synchronized at the same time.

- `Start Time (T0)` – the time when the spark generation circuit discharges and the formation of the bubble starts. This is found by searching the maximum force value between the start of the signal up to ET0.
- `Maximum Radius Time (T0_5)` – the time when the maximum radius of the bubble is reached. This time is found by searching the maximum force value found in the PVDF data from ET0 to ET1.
- `Collapse Time 1 (T1)` – the time when the first collapse occurs. This time is found by searching the maximum force value found in the PVDF data from ET1 to ET2.
- `Collapse Time 2 (T2)` – the time when the second collapse occurs. This time is found by searching the maximum force value found in the PVDF data from ET2 to within two frame periods worth of ET2, where a third collapse time is estimated to occur when a local minimum is not observed from the camera images at this time.

- `Bubble Period 1 (BP1)` – the time it takes for the bubble to develop from the start of the spark up to the first bubble collapse. Analogous to the measurement obtained in [9]. Equivalent to $T1 - T0$.

- `Bubble Period 2 (BP2)` – the time it takes for the bubble to develop from the first bubble collapse up to the second bubble collapse. Analogous to the measurement obtained in [9]. Equivalent to $T2 - T1$.

- `Rayleigh Collapse Time (RCT)` – the time it takes for the bubble to develop from the maximum size to the first bubble collapse [3]. Equivalent to $T1{-}T0\_5$.

- `Collapse Voltage (CVx)` – the voltage (in volts) of the peak occurring at the collapse time referred to (T1 or T2, denoted as Tx). This is obtained by getting the equivalent PVDF signal value at the time denoted by Tx.

- `Collapse Force (CFx)` – the force (in newtons) exerted during the peak at the collapse time referred to (T1 or T2, denoted as Tx). This is obtained by getting the equivalent PVDF signal value at the time denoted by Tx, and multiplying it with the obtained constant through Calibration of the PVDF sensor.

- `Collapse Width (CWx)` – the width (in seconds) of the peak occurring at the collapse time referred to (T1 or T2, denoted as Tx). This is obtained as the Full Width at Half Maximum, which is the width spanned by where the signal is at half the peak force to the left (at a time less than Tx) up to where the signal is at half the peak force to right (at a time greater than Tx).

- `Collapse Area (CAx)` – the area under the curve of the peak at the collapse time referred to (T1 or T2, denoted as Tx). This is obtained by integrating the voltage value over the time where the signal is at a minimum to the left (less than Tx) up to the time where the signal is at a minimum to the right (greater than Tx).

- `Collapse Impulse (CJx)` – the impulse at the peak (in newton-seconds) at the collapse time referred to (T1 or T2, denoted as Tx). This is obtained by integrating the force value (converted from voltage value) over the time where the signal is at a minimum to the left (less than Tx) up to the time where the signal is at a minimum to the right (greater than Tx).

These parameters are visually demonstrated in the following figure.

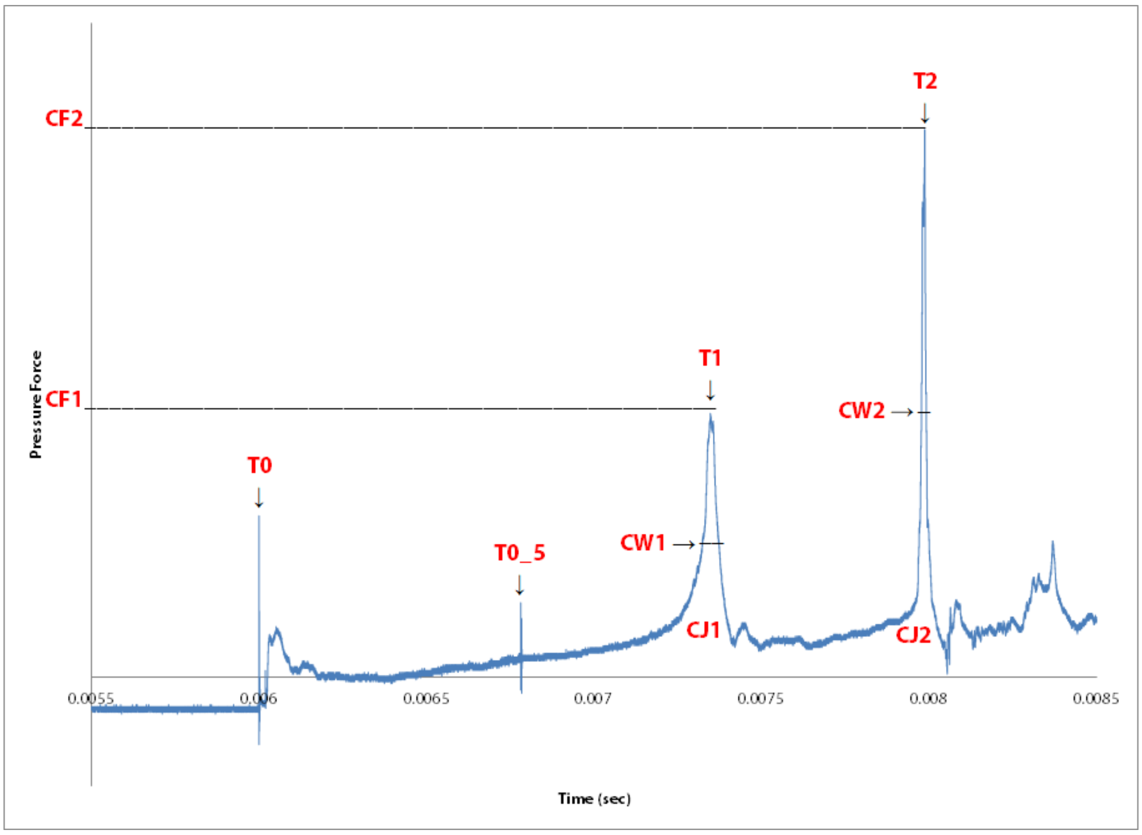


Figure 5.1: Parameters from PVDF Data

At times, the tool is not able to make the correct assessment of the parameters for a rare amount of cases (5 from the total 24 cases in the current version) without assistance. This is mainly due to the errors in estimating the second collapse time, as a collapse may not be seen by the camera given the disparity in the framerate. However, these cases can be resolved with slight corrections by the user. Estimations in searching for the correct peaks in the waveform may be corrected by the user in the output XLSX spreadsheet, as a part of the calculations are performed in Excel via formulas. Also, a camera with a faster framerate may mitigate the occurrence of these errors.

5.3 Pressure Force Waveform Regions

The pressure waveform obtained from the PVDF sensor confirms earlier results as to the behavior of the cavitating bubble onto the solid surface. Some regions have been determined to appear consistently in the PVDF waveform in single bubble setup, and are identified as such:

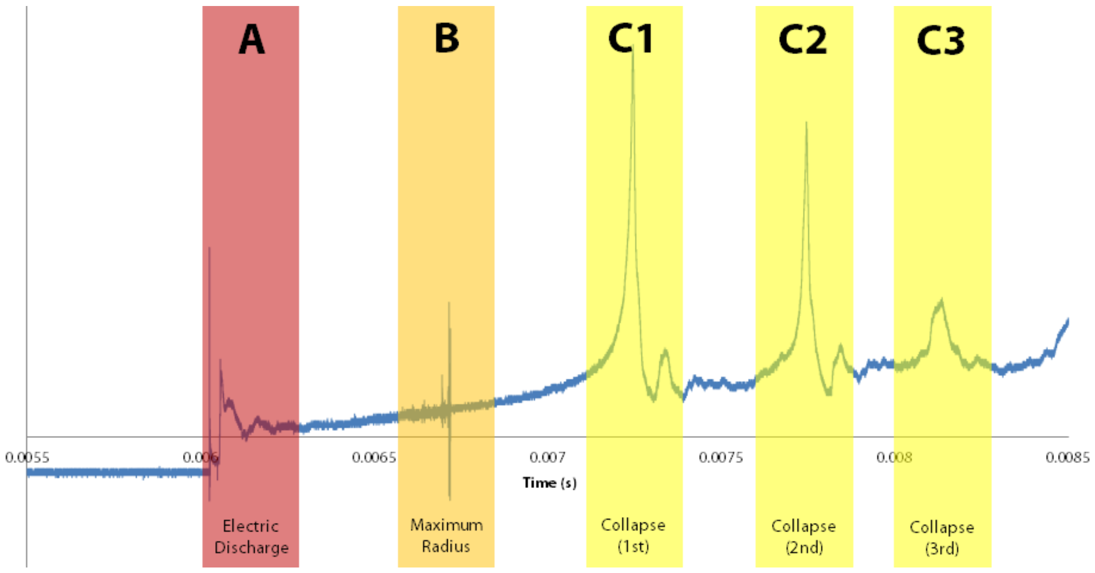


Figure 5.2: The identified Pressure Waveform Regions

- `Electric Discharge Region (A)` – the region when the spark generation circuit discharges and the formation of the bubble starts. As shown in [9], this signal in the pressure sensor results from the electrical discharge of the spark generation circuit. There is a short period of electrical interference which then clears out, and from here on it can be observed that the pressure is increasing smoothly and slowly until the next region is reached.

- `Maximum Radius Region (B)` – the region where the maximum radius of the bubble is reached. A very thin spike occurs at this part, presumably as this is where the radius changes direction from going larger to going smaller, causing a disturbance in the liquid medium recorded by the pressure sensor.

- `Collapse 1 (C1)` – the region where the Collapse Time T1 occurs. This region is the effect of the initial collapse of the bubble, and also where the bubble shape starts to be distorted from its original spherical form.

- `Collapse 2 (C2)` – the region where the Collapse Time T2 occurs. It occurs within the second bubble period, which is approximated as about half of the first bubble period, and thus comes relatively sooner right after the first collapse.

- `Further Collapse regions (C3, etc.)` – are regions beyond C1 and C2 where collapses occur. They are not covered by the scope of this study.

6 Results and Analysis

The results obtained from the experimental setup in Chapter 3 following the test conditions mentioned in Chapter 5 are then presented here.

6.1 Parameter Trends

From the single bubble setup, the parameters of Collapse Time, Collapse Force, Collapse Width, and Collapse Impulse may exhibit certain trends as the parameters of maximum radius and distance from the wall are changed, consolidated in the proximity parameter γ.

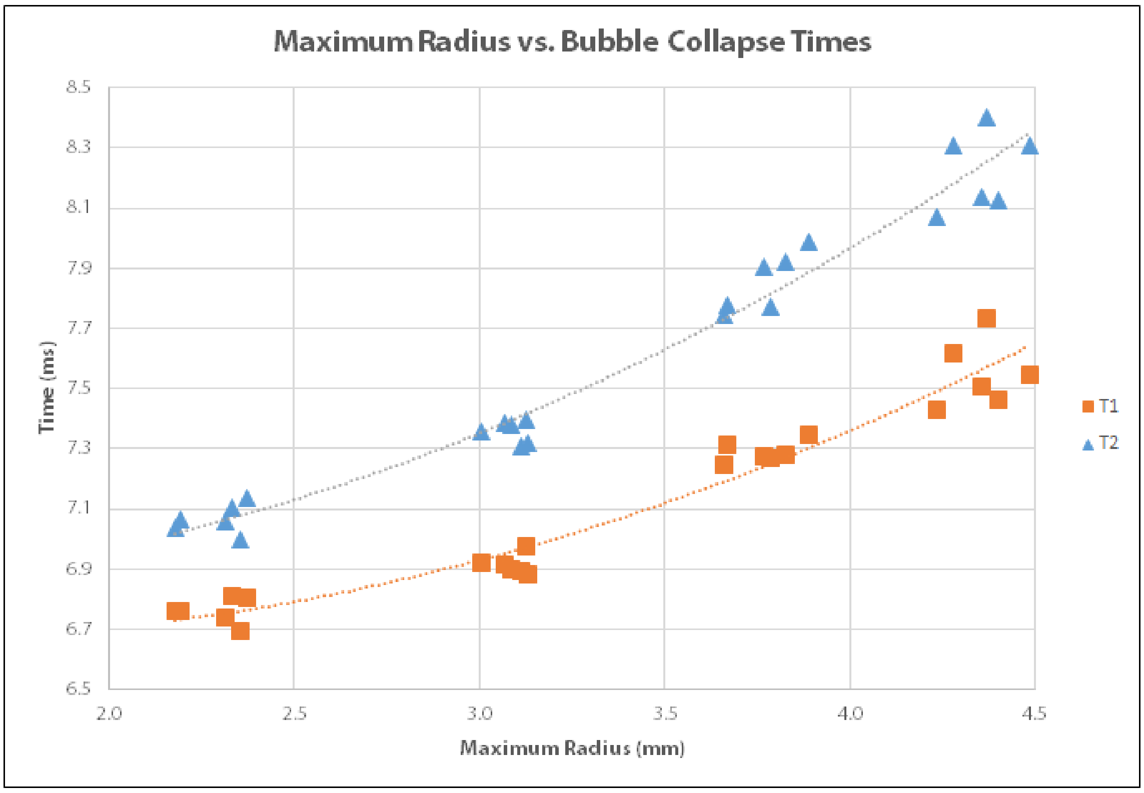


Figure 6.1: Maximum Bubble Radius vs. Collapse Times

The first parameter to be examined is Collapse Time (Tx) shown in Figure 6.1.

From the test conditions on configurations of single bubbles, it appears that the collapse times change irrespective of the distance from the wall, but are instead solely reliant on the maximum bubble radius. There is a strong square polynomial correlation (at 95 percent R^2 and 97 percent R^2 for T1 and T2 respectively) between Maximum Bubble Radius and Collapse Time as shown in the figures. Collapse Times t (in ms) can be expressed as $t = a_1R^2 + a_2R + a_3$ for any maximum radius R in mm. For this particular experimental setup, a_1, a_2, and a_3 are 0.1037, -0.2951, and 6.88 for T1 respectively, and 0.1169, -0.2021, and 6.90 for T2 respectively. In this case, most of the effect is based on the value of a_2, while a_3 is relatively constant.

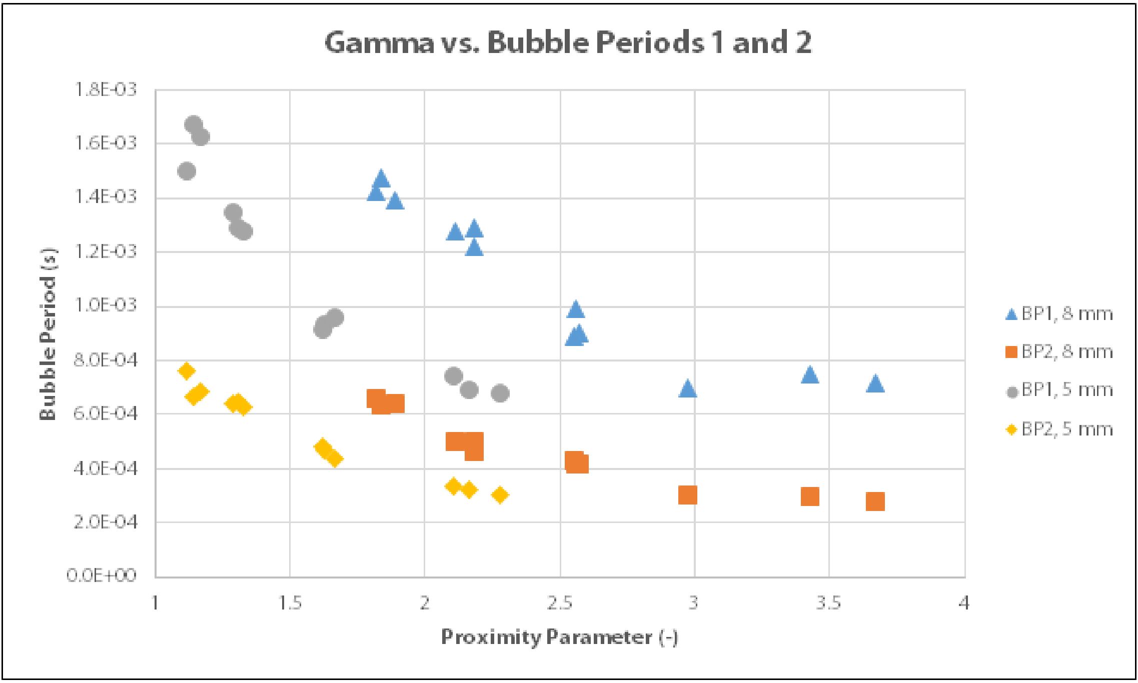


Figure 6.2: Gamma vs. Bubble Periods

As the relationship between Bubble Period and Collapse Time (shown in figure 6.2) is only through difference, it follows that this correlation will also occur for Bubble Periods 1 and 2. The non-reliance on bubble distance from wall is emphasized when BPx is compared with γ, as the curve of the line is more or less the same for the same radii but when the distance is considered, the line gets separated into the two divisions for 5 mm or 8 mm.

The relationship of the first collapse force CF1 with maximum bubble radius as shown in Figure 6.3 appears likewise not to be dependent as well on the distance from the wall, as is visible on the figure. Data points for both 5 mm and 8 mm distances are grouped together in a singular curve. The relationship from this curve is not very clear, but it seems to be a downward pointing 2nd degree polynomial

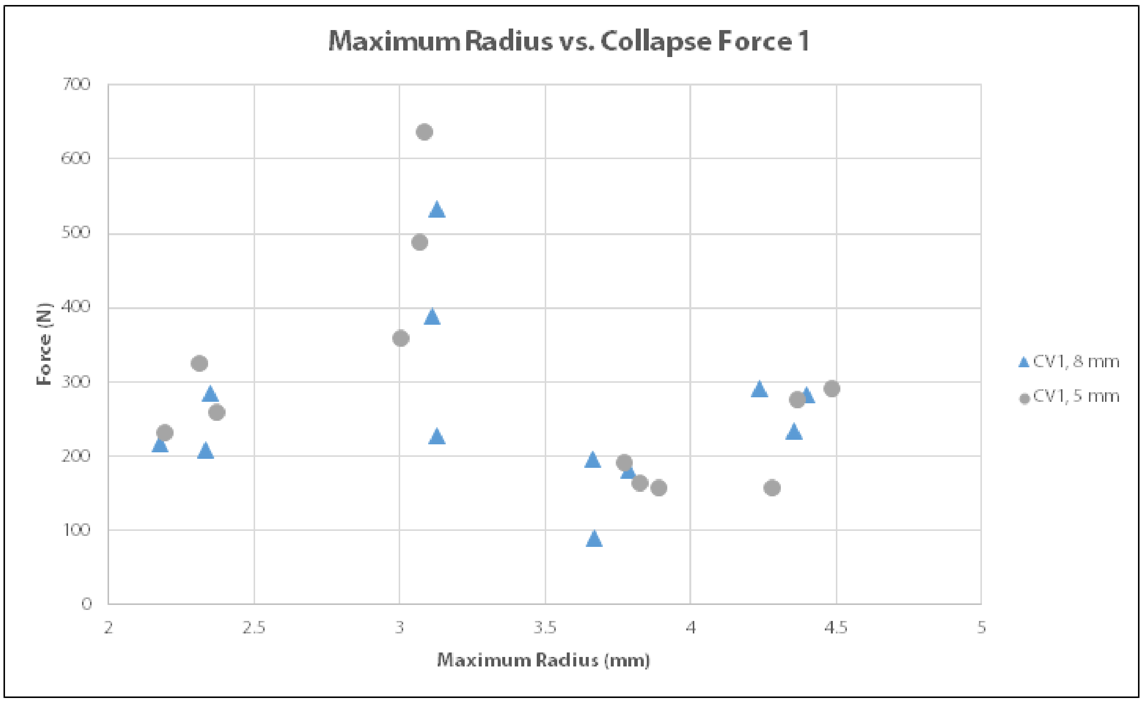


Figure 6.3: Maximum Radius vs. Collapse Force 1

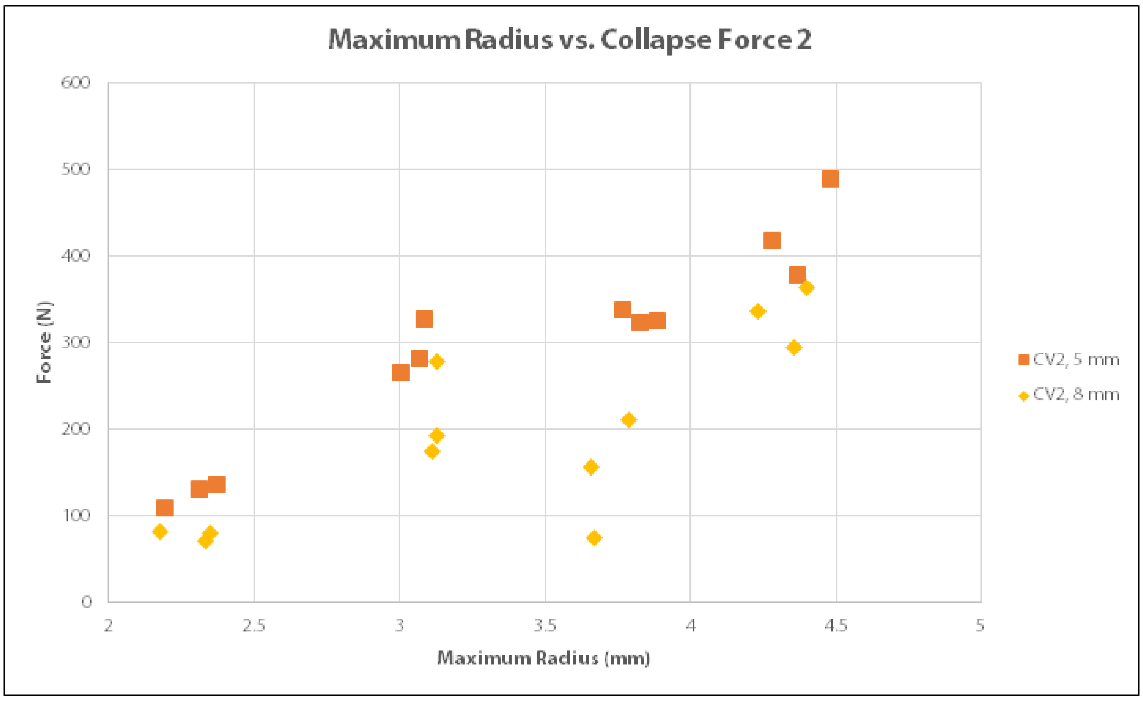


Figure 6.4: Maximum Radius vs. Collapse Force 2

curve, with maximum collapse forces occurring for maximum radii nearer to 3 mm.

The relationship of the second collapse force CF2 with maximum bubble radius as shown in Figure 6.4 seems to be more straightforward, as from what is shown here, CF2 increases as maximum radius increases. Likewise, it seems to be only slightly by the change in distance from the wall, as both datasets coincide with a similar linear correlation curve, as CF2 tends to be higher for the 5 mm data points than from the 8 mm data points. This can be attributed to the attachment of the second collapse to the solid wall, thus a higher force is exerted to the wall.

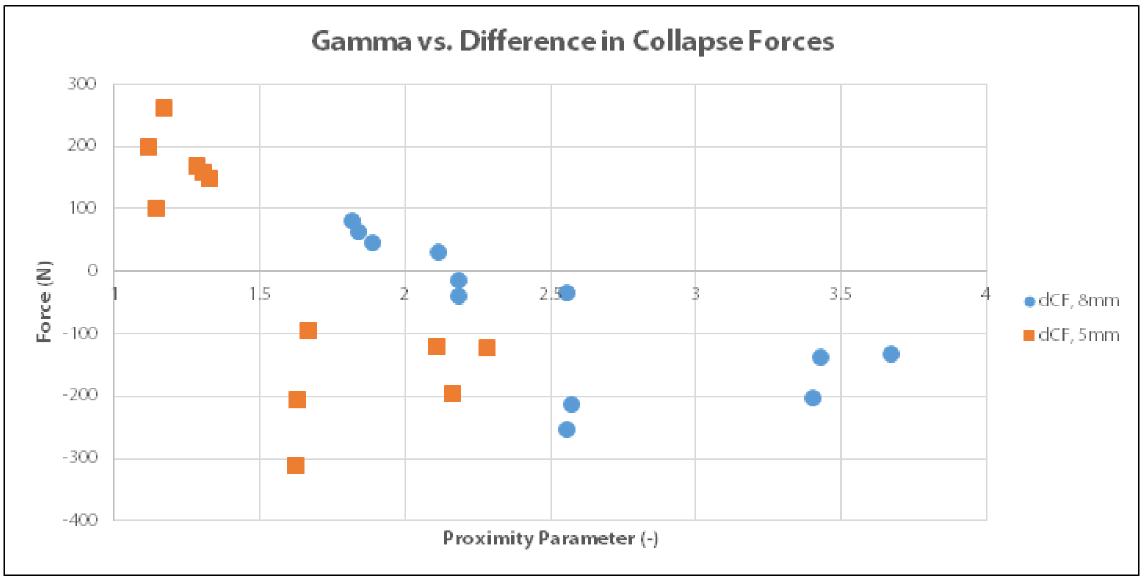


Figure 6.5: Gamma vs. Difference in Collapse Forces

The difference between the forces of the two collapse peaks is then assessed according to the proximity parameter γ, shown in Figure 6.5. The behavior described in Müller et al. [16], where the collapse patterns of these peaks depend on the distance of the bubble from the wall, is confirmed. As shown in the graph, there is a seemingly linear dependence of the peak force difference to the proximity. When γ is smaller, the force exerted by the second collapse tends to be higher than the first. As γ then increases, the first collapse tends to be higher than the second. The mentioned effect on CF2 is exhibited likewise here, since the difference dCF is expressed as CF2 - CF1.

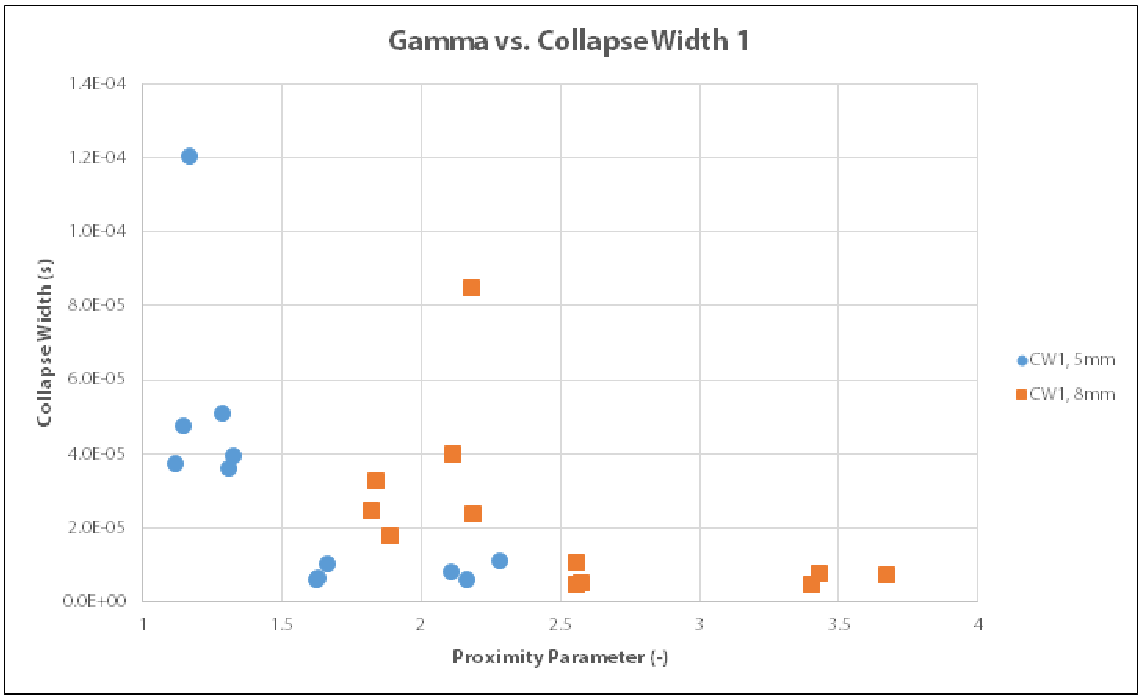


Figure 6.6: Gamma vs. Collapse Width 1

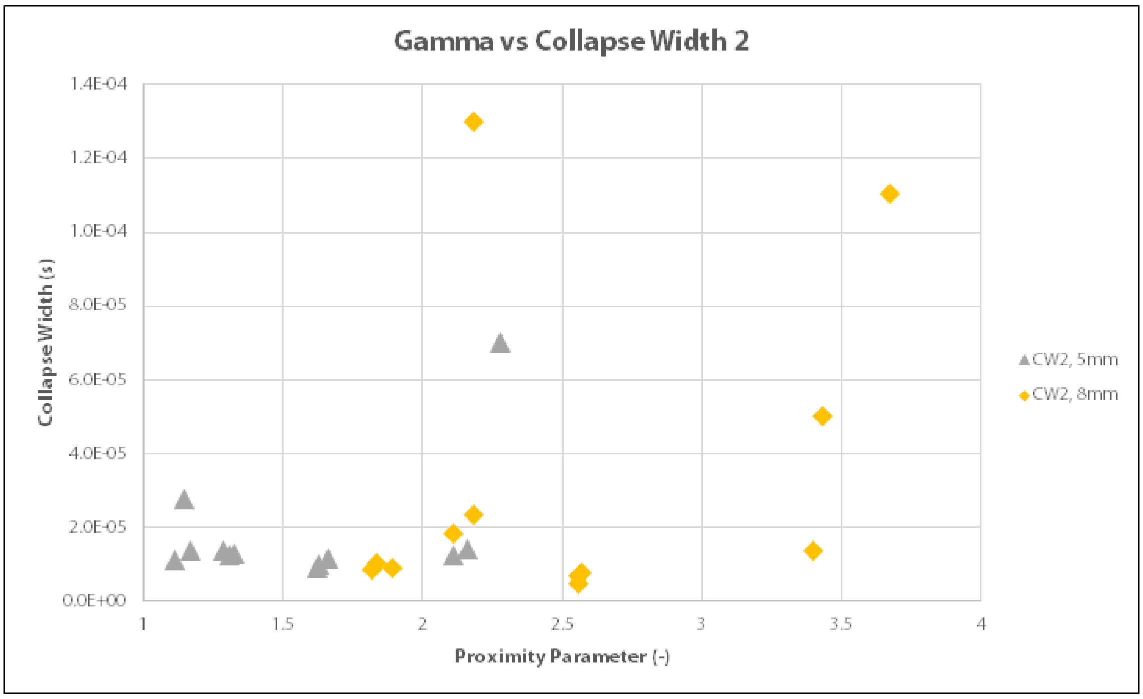


Figure 6.7: Gamma vs. Collapse Width 2

The width of the first collapse (CW1) is compared with γ in Figure 6.6, and it appears that the width of the collapse peak is also related to the proximity parameter. CW1 appears to decrease as γ increases, in what seems to be exponential decline which evens out at some point. This means that the impulse from the first collapse might seem to have a minimum duration on the solid wall.

The width of the second collapse (CW2) also appears to be affected by the proximity parameter as shown in Figure 6.7. A few outliers have been detected on this graph based on the nature of the signal and how CW2 is computed (as a full width at half maximum of the collapse peak). However, it is still quite evident that CW2 increases as γ increases, in a seemingly exponential growth.

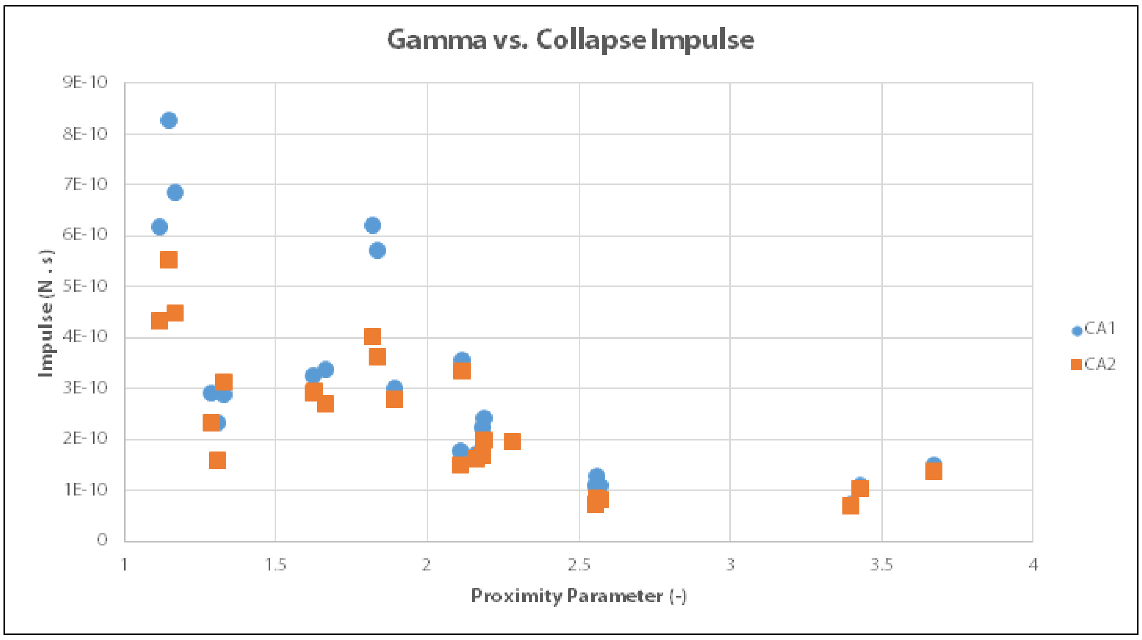


Figure 6.8: Gamma vs. Collapse Impulse

Finally, as demonstrated in Figure 6.8, the area under the curve for each peak, and subsequently, the impulse exerted by each peak, is also affected by the proximity. Both impulses CJ1 and CJ2 decrease as γ increases. This is consistent to the definition of the Collapse Impulse being related to the overall effect of the collapse on the solid wall. When γ is higher (and thus the bubble is smaller or more far away), the effect of the collapse is weaker on the wall. Otherwise, for lower values of γ (where the bubble is larger or nearer to the wall), the impact of the cavitation collapse over this time is stronger.

6.2 The Two-Bubble Waveform

The double bubble setup displays a waveform that is similar to that of the single bubble pressure waveform, but as expected, similar to two superimposed pressure waveforms of this type.

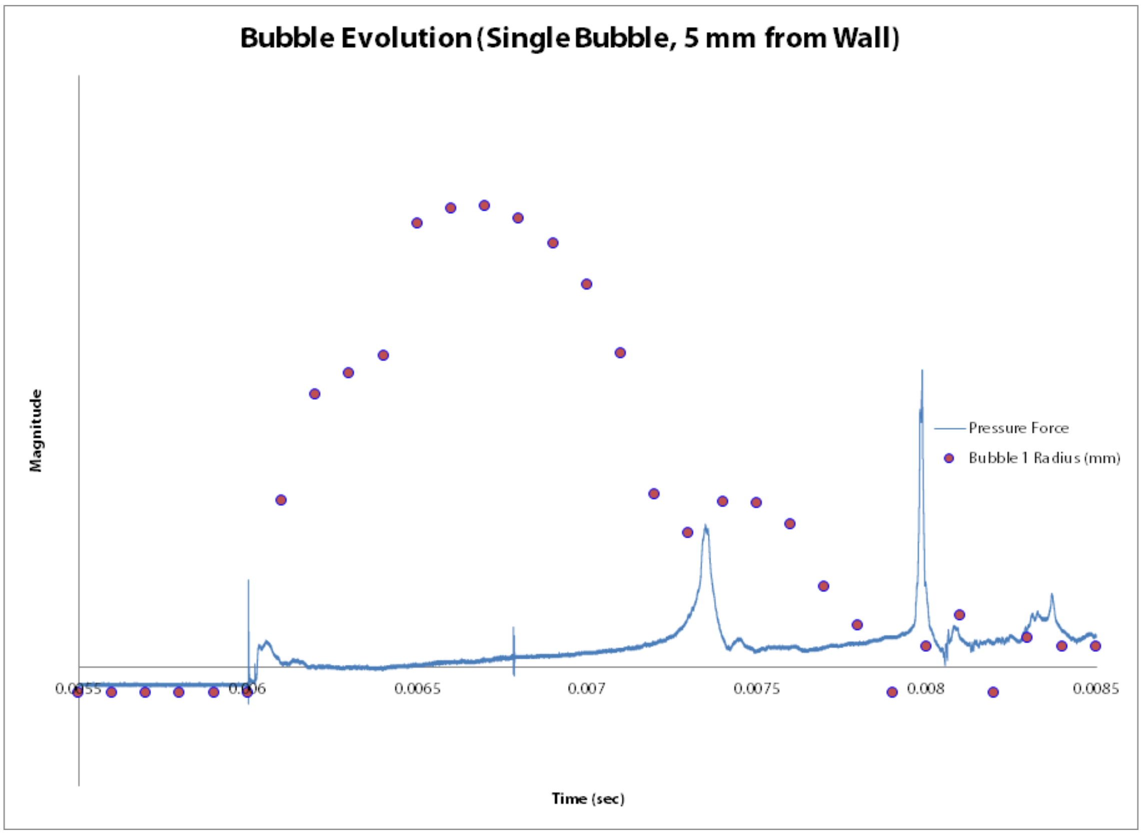
A comparison of the double bubble setup waveform to the single bubble setup waveform is shown in Figure 6.9. In the single bubble waveform, the pressure force waveform regions as discussed in Section 5.3 is clearly delineated from the few peaks that are detected by the sensor. The analysis tool is also able to trace the radius of the bubble as it develops, and it follows the shape of the Rayleigh-Plesset equation, collapses into a small size, and rises again to collapse in the next bubble period.

There are significantly more peaks in the waveform since two bubbles have been detected to undergo the cavitation process. As a result, there will be additional peaks but since the collapses occur at roughly the same period, the collapse forces may be affected by each other. In this analysis, the output of the analysis tool is modified in such a way it will also calculate for another set of peaks based on the Estimated Collapse Times independently calculated by the tool for the second bubble. This will be modified to be done automatically in the second version of the analysis tool.

In Figures 6.10 to 13, the two bubbles detected during the test run **(2b)** are compared to the nearest found data points as seen from the single bubble data in terms of size **(1b)**, as well as the data of the average found for the same trials for a certain input source voltage and distance from the wall **(1b**, **average)**. It will then be seen if the same trends can be found on each of the bubbles for each of the parameters bubble period (BPx), collapse times (Tx), collapse forces (CFx), difference between collapse forces (dCF), collapse widths (CWx), and collapse impulses (CJx).

For the double bubble setup 8 mm away from the wall, several things can be observed. First, each of the individual bubbles, as expected, share the same start times and collapse times with the single bubbles with whom they share the approximate size with. As a result, the bubble periods are also consistent to what are expected, with BP2 still being near the value of half of BP1.

The collapse force observed is relatively consistent for CF1 and CF2 (being in the middle of the values expected for the nearest average sizes), the left bubble value for CF2 having a considerable difference in magnitude. It is possible that this magnitude is also affected by some energy from the first bubble collapse of the right bubble, which has a collapse time (7.249 ms) close to the second bubble collapse time of the left bubble (7.364 ms). The difference in CF1 and CF2 are therefore also



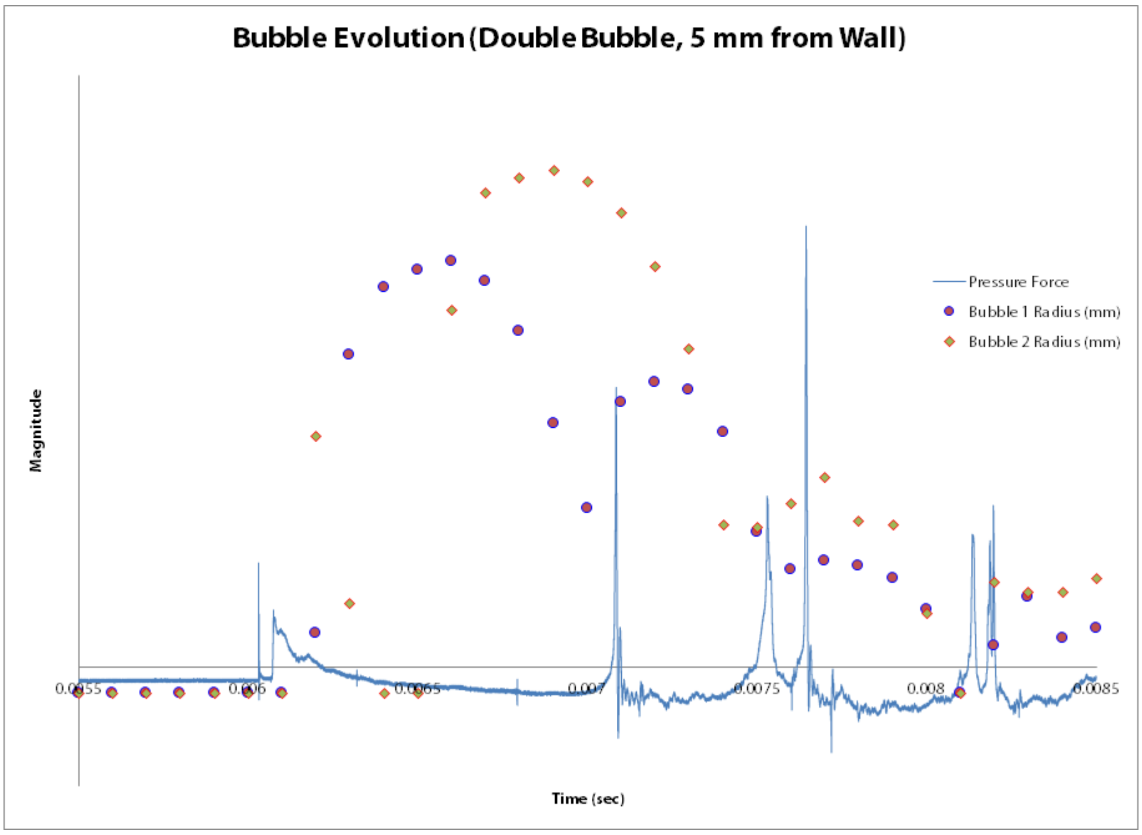
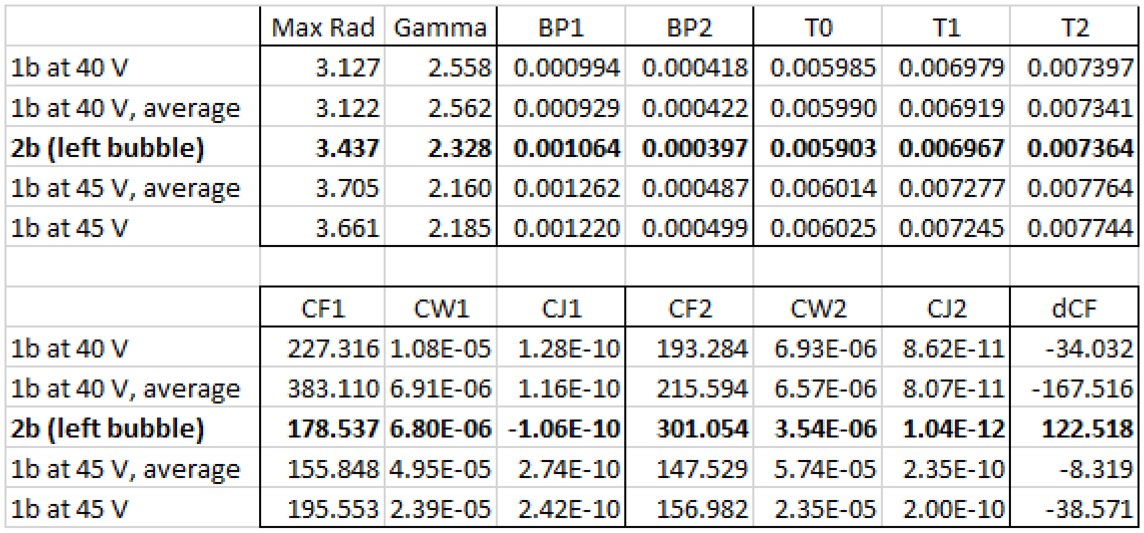


Figure 6.9: Single Bubble vs. Double Bubble Waveforms

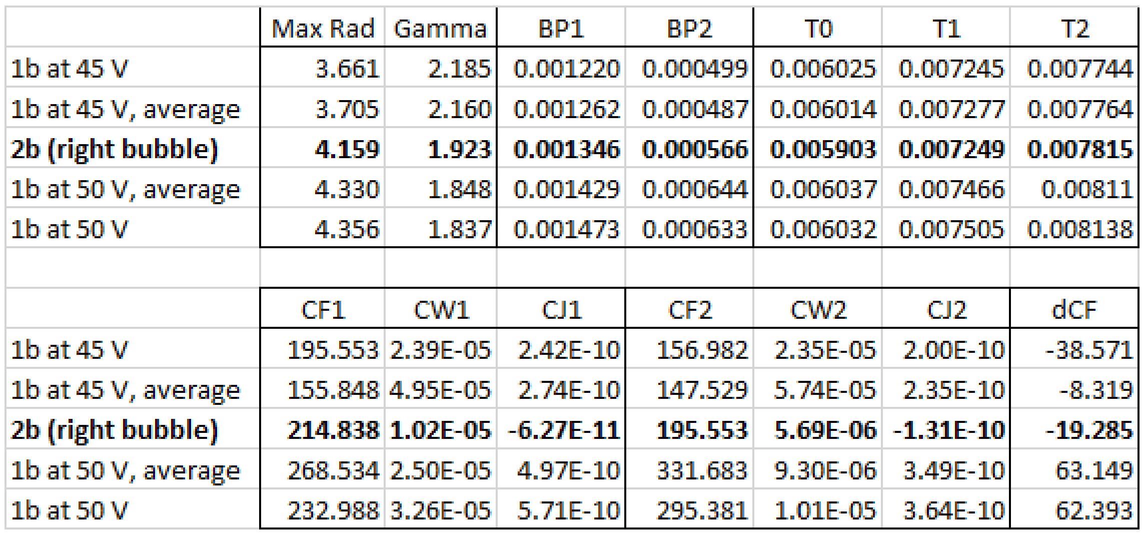
affected by this, as dCF for the left bubble seems to be inconsistent to the earlier trend discussed.



	Max Rad	Gamma	BP1	BP2	T0	T1	T2
1b at 40 V	3.127	2.558	0.000994	0.000418	0.005985	0.006979	0.007397
1b at 40 V, average	3.122	2.562	0.000929	0.000422	0.005990	0.006919	0.007341
2b (left bubble)	**3.437**	**2.328**	**0.001064**	**0.000397**	**0.005903**	**0.006967**	**0.007364**
1b at 45 V, average	3.705	2.160	0.001262	0.000487	0.006014	0.007277	0.007764
1b at 45 V	3.661	2.185	0.001220	0.000499	0.006025	0.007245	0.007744

	CF1	CW1	CJ1	CF2	CW2	CJ2	dCF
1b at 40 V	227.316	1.08E-05	1.28E-10	193.284	6.93E-06	8.62E-11	-34.032
1b at 40 V, average	383.110	6.91E-06	1.16E-10	215.594	6.57E-06	8.07E-11	-167.516
2b (left bubble)	**178.537**	**6.80E-06**	**-1.06E-10**	**301.054**	**3.54E-06**	**1.04E-12**	**122.518**
1b at 45 V, average	155.848	4.95E-05	2.74E-10	147.529	5.74E-05	2.35E-10	-8.319
1b at 45 V	195.553	2.39E-05	2.42E-10	156.982	2.35E-05	2.00E-10	-38.571

Figure 6.10: Parameter Data for Double Bubble (left), 8 mm from Wall



	Max Rad	Gamma	BP1	BP2	T0	T1	T2
1b at 45 V	3.661	2.185	0.001220	0.000499	0.006025	0.007245	0.007744
1b at 45 V, average	3.705	2.160	0.001262	0.000487	0.006014	0.007277	0.007764
2b (right bubble)	**4.159**	**1.923**	**0.001346**	**0.000566**	**0.005903**	**0.007249**	**0.007815**
1b at 50 V, average	4.330	1.848	0.001429	0.000644	0.006037	0.007466	0.00811
1b at 50 V	4.356	1.837	0.001473	0.000633	0.006032	0.007505	0.008138

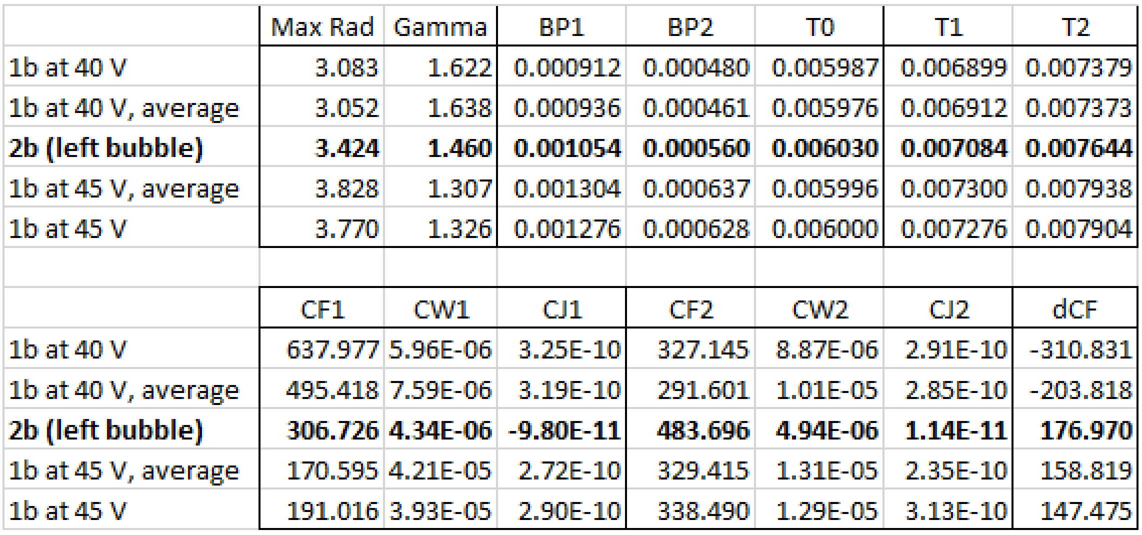
	CF1	CW1	CJ1	CF2	CW2	CJ2	dCF
1b at 45 V	195.553	2.39E-05	2.42E-10	156.982	2.35E-05	2.00E-10	-38.571
1b at 45 V, average	155.848	4.95E-05	2.74E-10	147.529	5.74E-05	2.35E-10	-8.319
2b (right bubble)	**214.838**	**1.02E-05**	**-6.27E-11**	**195.553**	**5.69E-06**	**-1.31E-10**	**-19.285**
1b at 50 V, average	268.534	2.50E-05	4.97E-10	331.683	9.30E-06	3.49E-10	63.149
1b at 50 V	232.988	3.26E-05	5.71E-10	295.381	1.01E-05	3.64E-10	62.393

Figure 6.11: Parameter Data for Double Bubble (right), 8 mm from Wall

Likewise, collapse widths are also observed to be consistent with the expected single bubble sizes, except for the different first collapse width (CW1) of the right bubble. The new value is relatively smaller in magnitude, implying that the effect of the first bubble collapse of this bubble is cut short due to another bubble collapse happening at the same time as itself.

The collapse areas for both bubbles are not consistent in measure to the collapse areas computed for single bubbles of the same relative size. This is most probably due to the effect of the electrical signal detecting too many function values which are negative (see Recommendations for a possible remedy to this effect). However, when this signal is compared to the signal of the double bubble from 5 mm away, the CJx values for these are of comparable range, and the CJ1 and CJ2 for the 8 mm double bubble is considerably lower. This is consistent to what is expected as bubbles which develop farther from the wall are likely to have a lesser impulse effect on the wall, and thus the integral for their peaks (area under the curve) is of a smaller value.

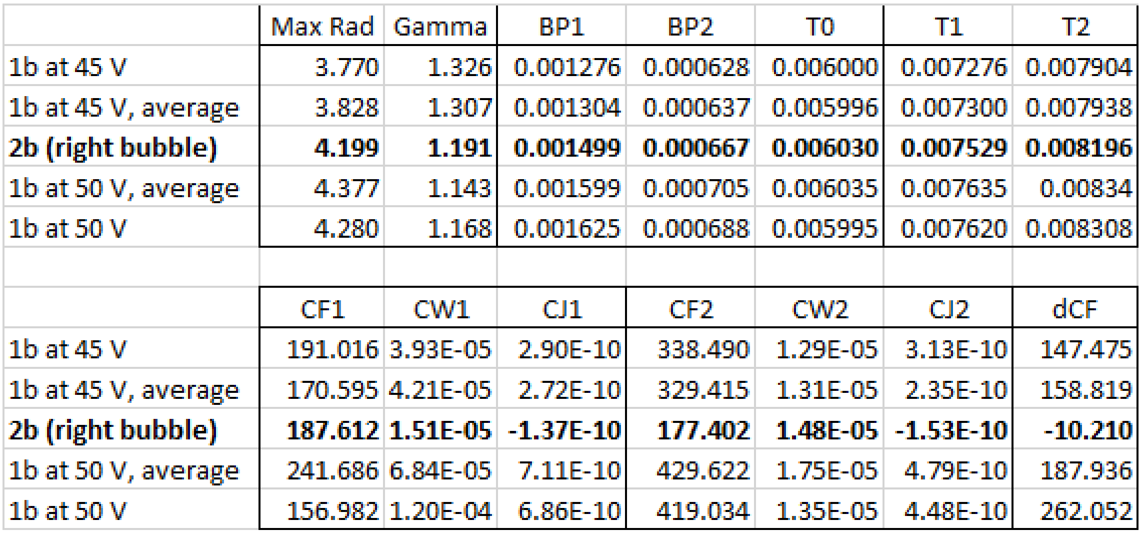
Similar points were noticed for the setup at 5 mm away from the wall.



	Max Rad	Gamma	BP1	BP2	T0	T1	T2
1b at 40 V	3.083	1.622	0.000912	0.000480	0.005987	0.006899	0.007379
1b at 40 V, average	3.052	1.638	0.000936	0.000461	0.005976	0.006912	0.007373
2b (left bubble)	**3.424**	**1.460**	**0.001054**	**0.000560**	**0.006030**	**0.007084**	**0.007644**
1b at 45 V, average	3.828	1.307	0.001304	0.000637	0.005996	0.007300	0.007938
1b at 45 V	3.770	1.326	0.001276	0.000628	0.006000	0.007276	0.007904

	CF1	CW1	CJ1	CF2	CW2	CJ2	dCF
1b at 40 V	637.977	5.96E-06	3.25E-10	327.145	8.87E-06	2.91E-10	-310.831
1b at 40 V, average	495.418	7.59E-06	3.19E-10	291.601	1.01E-05	2.85E-10	-203.818
2b (left bubble)	**306.726**	**4.34E-06**	**-9.80E-11**	**483.696**	**4.94E-06**	**1.14E-11**	**176.970**
1b at 45 V, average	170.595	4.21E-05	2.72E-10	329.415	1.31E-05	2.35E-10	158.819
1b at 45 V	191.016	3.93E-05	2.90E-10	338.490	1.29E-05	3.13E-10	147.475

Figure 6.12: Parameter Data for Double Bubble (left), 5 mm from Wall



	Max Rad	Gamma	BP1	BP2	T0	T1	T2
1b at 45 V	3.770	1.326	0.001276	0.000628	0.006000	0.007276	0.007904
1b at 45 V, average	3.828	1.307	0.001304	0.000637	0.005996	0.007300	0.007938
2b (right bubble)	**4.199**	**1.191**	**0.001499**	**0.000667**	**0.006030**	**0.007529**	**0.008196**
1b at 50 V, average	4.377	1.143	0.001599	0.000705	0.006035	0.007635	0.00834
1b at 50 V	4.280	1.168	0.001625	0.000688	0.005995	0.007620	0.008308

	CF1	CW1	CJ1	CF2	CW2	CJ2	dCF
1b at 45 V	191.016	3.93E-05	2.90E-10	338.490	1.29E-05	3.13E-10	147.475
1b at 45 V, average	170.595	4.21E-05	2.72E-10	329.415	1.31E-05	2.35E-10	158.819
2b (right bubble)	**187.612**	**1.51E-05**	**-1.37E-10**	**177.402**	**1.48E-05**	**-1.53E-10**	**-10.210**
1b at 50 V, average	241.686	6.84E-05	7.11E-10	429.622	1.75E-05	4.79E-10	187.936
1b at 50 V	156.982	1.20E-04	6.86E-10	419.034	1.35E-05	4.48E-10	262.052

Figure 6.13: Parameter Data for Double Bubble (right), 5 mm from Wall

CF2 of the left side bubble and CW1 of the right side bubble are also different from the expected values, as second collapse time of left bubble (7.644 ms) is close to the first collapse time of the right bubble (7.529 ms). The CF2 of the right side bubble is also observed to be affected, as it will be shown in the graph of the waveforms, CF3 of the left side bubble is also likely to occur at this same time. As mentioned, comparing CJx for this configuration with that of CJx for the 8 mm away configuration will yield a consistent result. CJ1 and CJ2 for the 5 mm double bubble configuration is higher in magnitude, as bubbles which develop nearer to the wall are likely to have a greater impulse effect on the wall, resulting in a greater value for area under the curve.

The waveforms for the two double bubble setups are then compared. The first and second collapse times are also zoomed in to view the peaks that occur at this point. The camera images for the bubble evolution in both are also shown below.

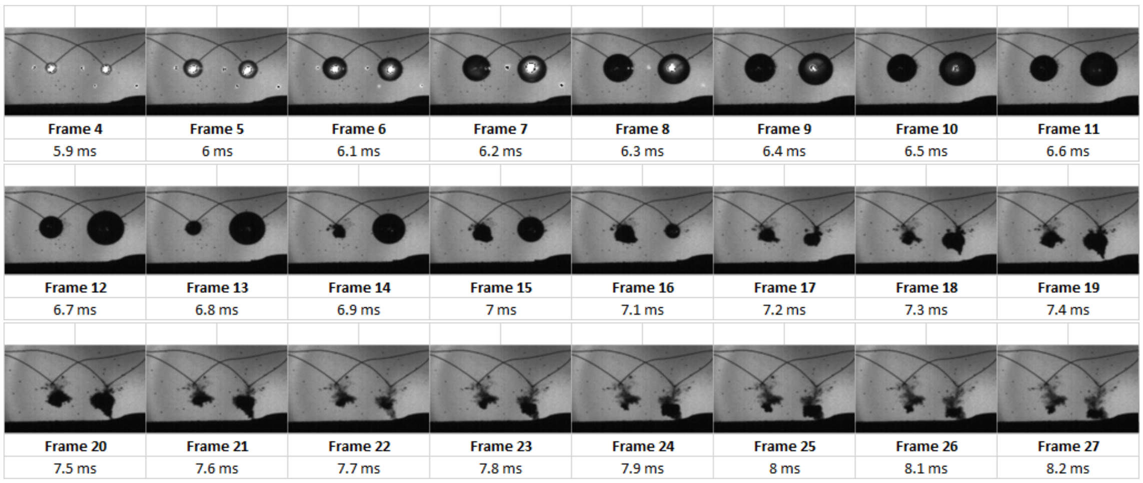


Figure 6.14: Bubble Evolution of Double Bubble (left), 8 mm from Wall

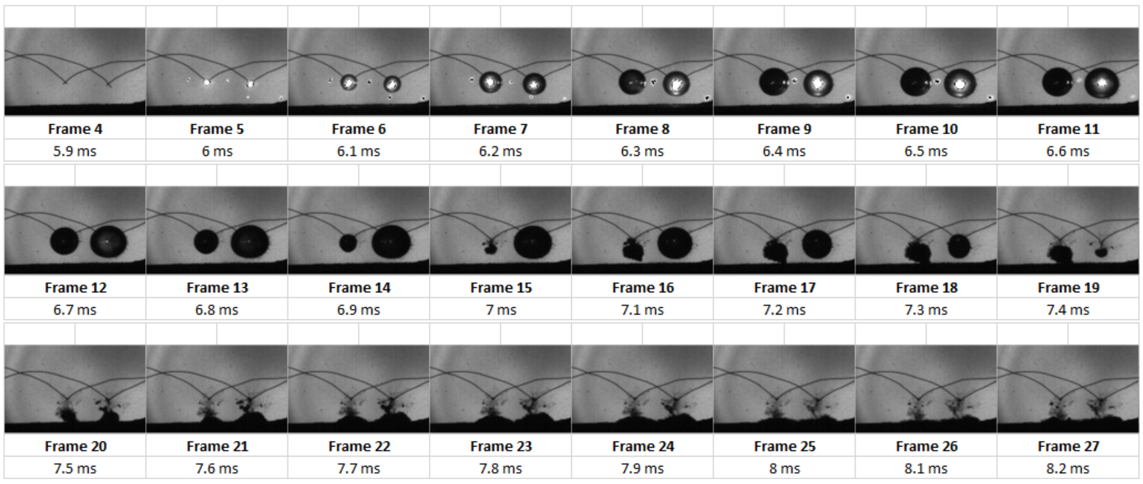
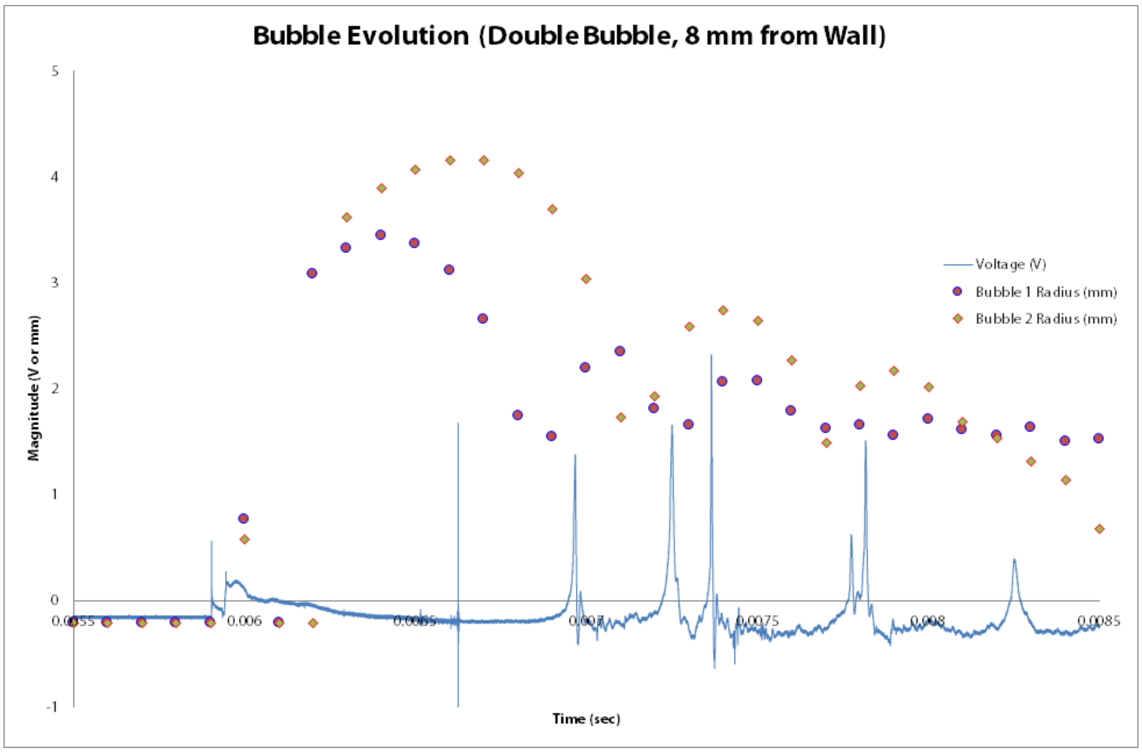


Figure 6.15: Bubble Evolution of Double Bubble (left), 5 mm from Wall



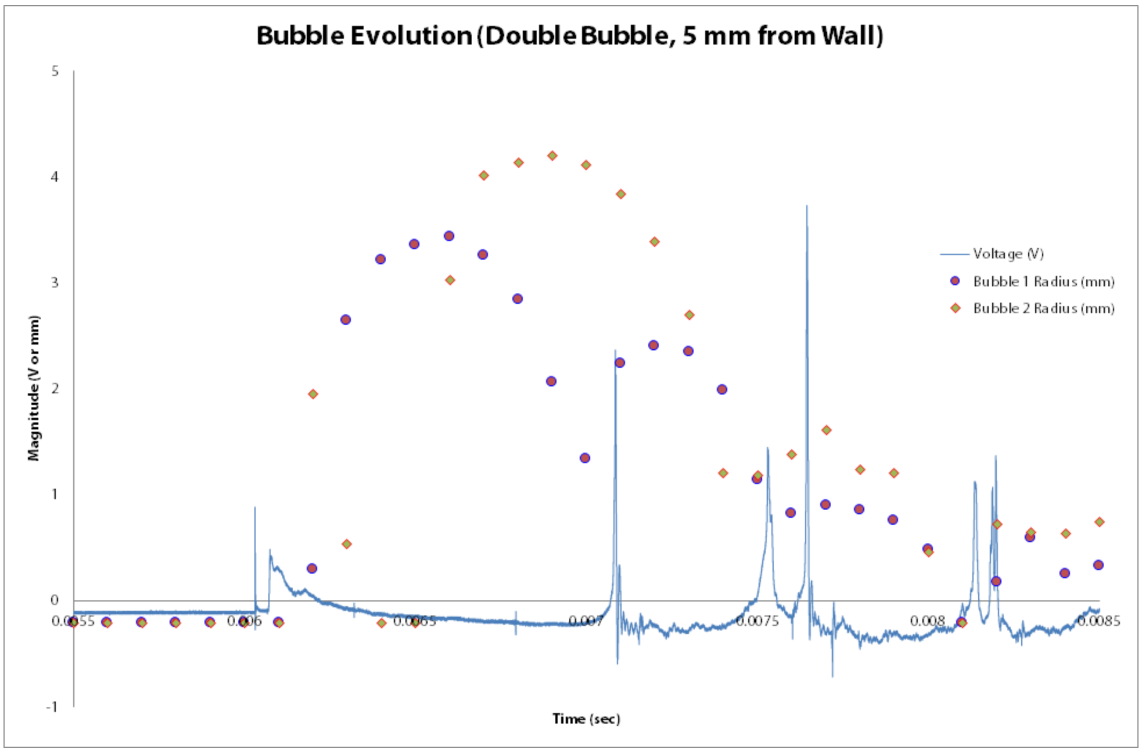
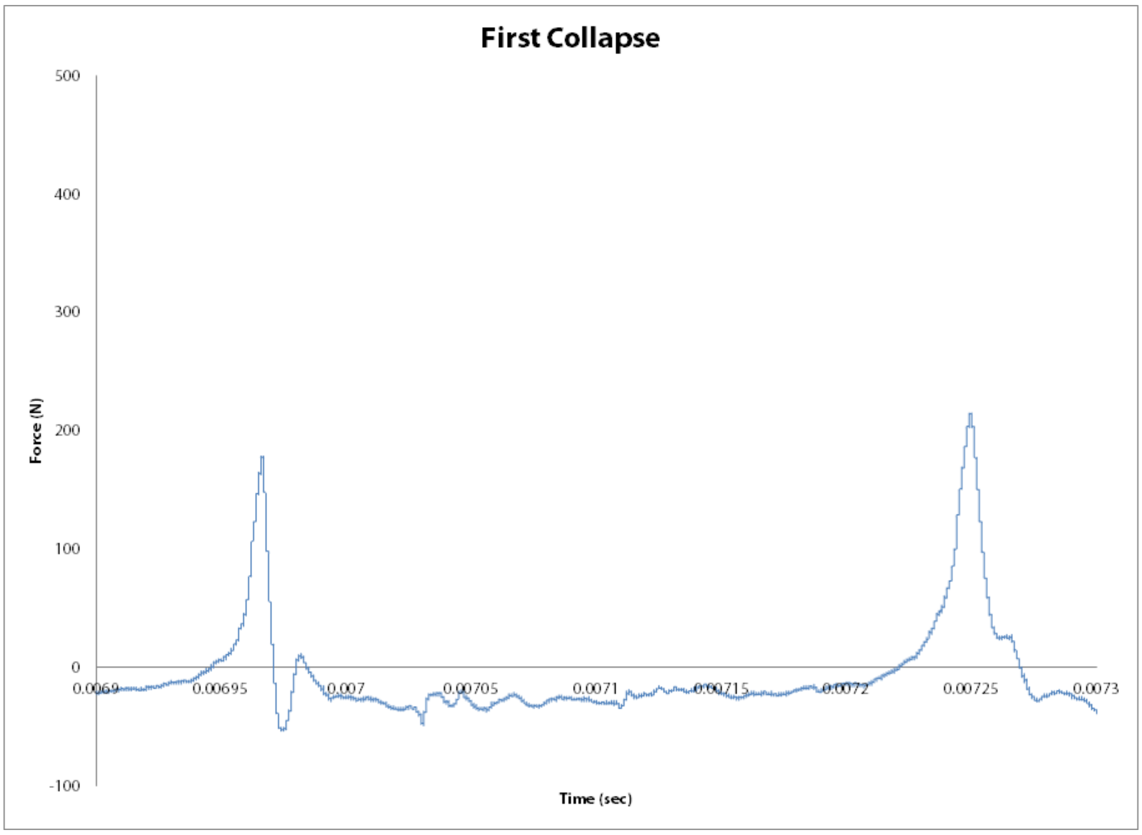


Figure 6.16: Bubble Radius vs. Force Graphs for Double Bubble Setup

As can be seen in the waveforms, multiple peaks occur throughout the evolution of the bubble. The bubble radii are consistent to what are expected, especially the spherical evolution of the bubble within BP1 following the Rayleigh-Plesset equation. The local minima as observed from the radius sizes obtained from the camera (shown as dots and diamonds) appear to correspond to certain peaks in the pressure signal waveform. Also of note that there are some frames which do not render well due to the algorithm of searching for contours, and as such, some outliers or undetected radii can be seen in the graph, although the shape is very much still visible and can be followed.

Figures 6.17 and 6.18 show the first and second collapses of the 8 mm and 5 mm distance from wall double-bubble setups respectively. The proximity of the collapses to each other sometimes affect the reading and the peaks appear to join together into a single peak with a split top. This is most noticeable in 6.18 at around the set of second collapses. This peak appears to be one peak with a deep splitting on its top. The peak in a similar position at 6.17 is presumed to even be two peaks already joined together, but the amplitude of this peak is quite low and not much difference is discerned from the camera data.

The maximum radius region (Region B) still appears as a very short and very narrow spike in the middle of BP1. It appears that the amplitude is less when the bubble is nearer to the wall as there is less distance from the wall to affect from the change of growth direction of the bubble.



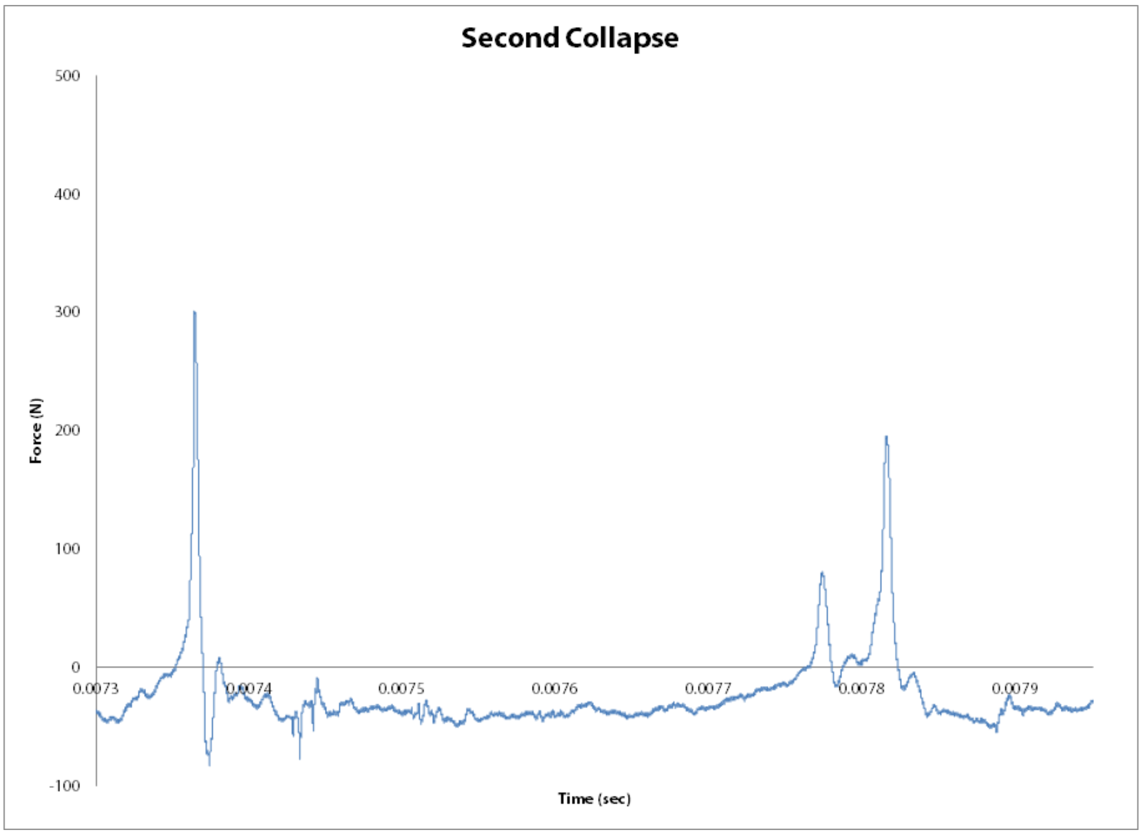
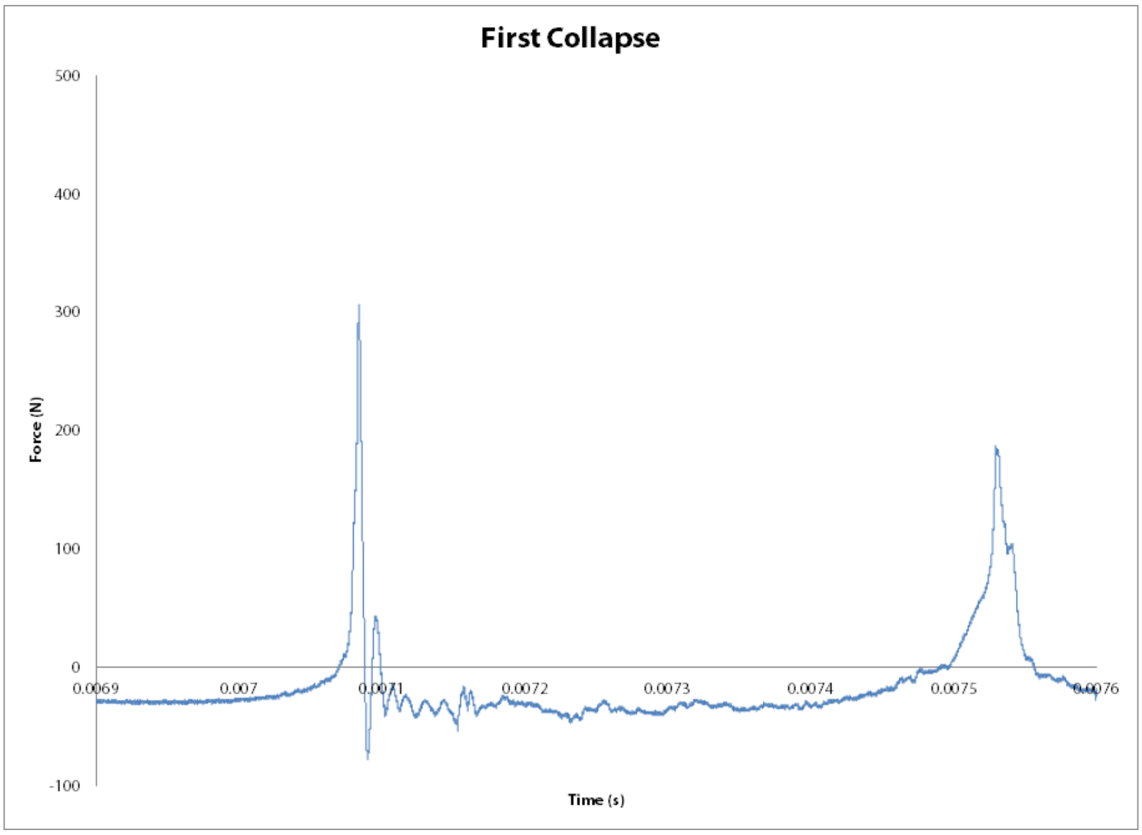


Figure 6.17: First and Second Bubble Collapses for Double Bubble, 8 mm



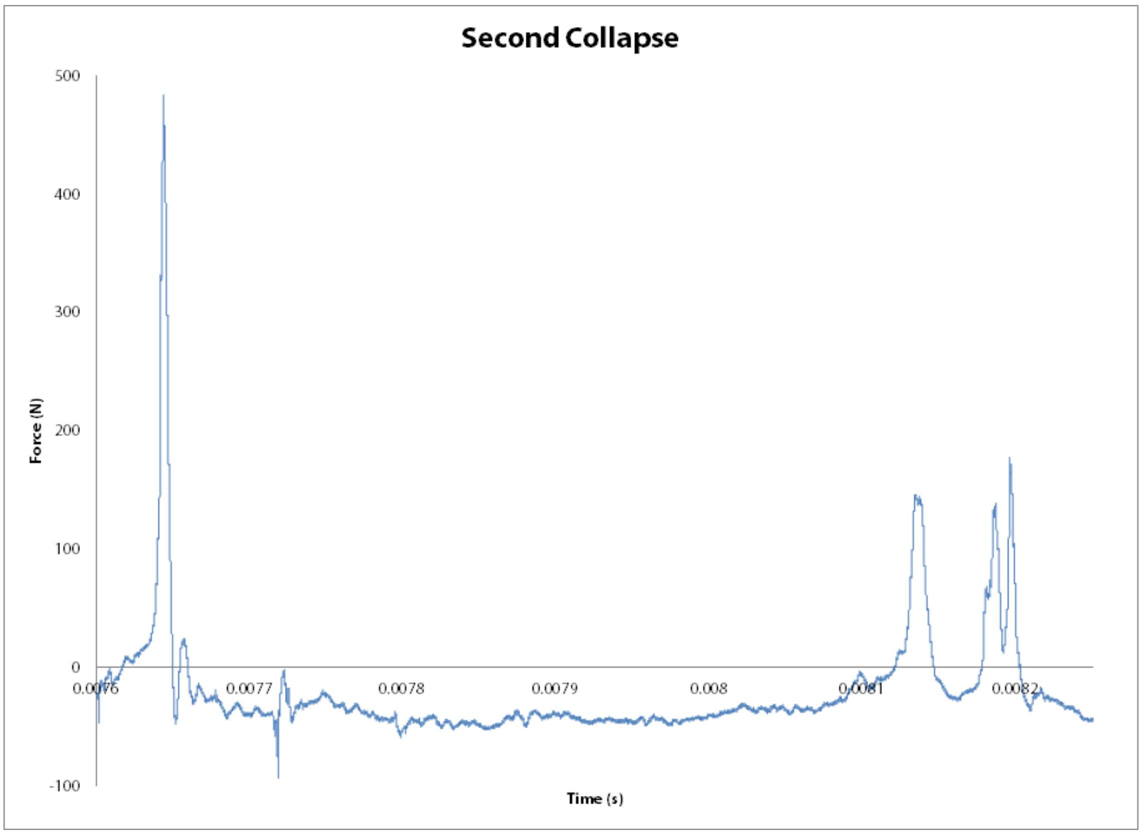


Figure 6.18: First and Second Bubble Collapses for Double Bubble, 5 mm

7 Conclusions and Recommendations

7.1 Summary

Throughout this study, the following tasks have been performed toward the realization of the objectives set in investigating cavitation bubble dynamics.

- The state of the art in investigating the behavior of bubbles produced during the cavitation process and the interaction of bubble structures near a solid wall is reviewed through the comparison of related literature. The potential of image recognition for cavitation analysis is also explored.
- The experimental setup for recreating the cavitation phenomenon was defined, prepared, and implemented. Single bubbles and bubble structures (in this case, two-bubble systems) are created with the experimental setup. The bubble dynamic thereafter was measured optically (with high-speed CCD camera) and acoustically (with the use of PVDF sensors).
- An analysis tool for processing the data from the bubble dynamics measurement was developed in the open source Eclipse environment, written in C++.
- A testing plan was executed, running trials with the experimental setup for two distances (5 mm and 8 mm) and four approximate bubble sizes (generated from 35, 40, 45, and 50 V for single bubble, and 50 V for double bubble).
- The PVDF (acoustic) and camera (optical) information was processed with the analysis tool, leading to understanding of selected parameters and the trends that occurred when bubble size and distance from solid wall were modified.
- From these obtained results, the bubble structures of single and multiple bubbles were analyzed and some conclusions and observations were explored from the analysis process.

7.2 Conclusions from Analysis

The analysis tool is able to perform calculations from the camera and PVDF sensor data which observe the evolution of the cavitating bubble. Through repeated use of the tool throughout a significantly-sized volume of data, comparisons can be made and trends regarding the different bubble parameters can be observed in a reasonably shorter time.

As such, it was possible to make a precursory analysis of the double bubble waveform. The estimated collapse times extrapolated from the camera image are able to match respective waveforms in the double bubble signal. It is also possible to compare the two bubbles in this kind of setup to configurations of single bubbles with similar size and see their effects on the solid wall. The added capability of the analysis tool will even further accelerate this measuring process and give precision which is better and taking less effort than when measurements are done manually.

The double bubble waveform is then characterized as a superimposition of two single bubbles acting on different sides of the film, as it will be possible to match waveforms in the signal corresponding to the pressure waveform regions described earlier on. When the bubbles activate at the exact same time, it is also possible to detect split peaks as it is as if two separate bubbles exert their respective energies onto the film at the same time. However, at the same time, the collapses may have effects on each other, which can potentially result in the combination of the collapse peaks into a single peak exerting force onto the solid wall.

The efforts presented here, especially the analysis tool, are exhibited in this study to be an initial step in the understanding of multiple bubble cavitation as affected on a solid wall.

7.3 Recommendations

7.3.1 Expansion of Test Conditions

In order to test the accuracy and precision of the presented data, more configurations are proposed for the test conditions. When other configurations are explored, such as extending the range of tests $\gamma < 1$ and $\gamma > 4$, a broader scope of understanding of the cavitating bubble regime can be reached.

7.3.2 Controlling of Bubble Size

The best efforts performed for this study was able to maintain the bubble size between the two bubbles in the double bubble setup to around within 10-20 percent of the average between them. This is presumably due to the nature of the electrical discharge selecting randomly between the two paths of conduction, namely the two spark junctions underwater. It has been found throughout the course of the study that the effect can be mitigated by having the least amount of branching off points for the electricity to pass through. Previous attempts for producing two bubbles at once involved the use of separate thick electrodes as the conductor where the fine wires are soldered on, in turn being connected together by screws (which might lead to a considerable amount of randomness to the junction), and through Y-shaped conductors where the thin wires were soldered on (relatively better, but have three solder branching off points in total as pictured).

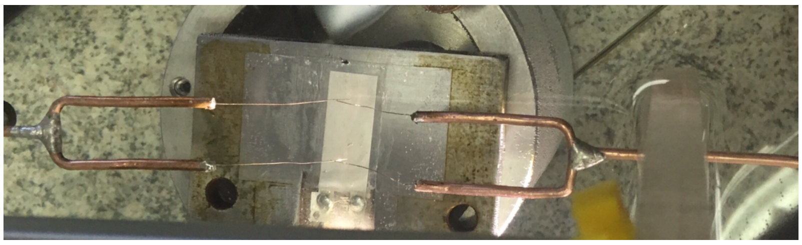


Figure 7.1: Old Electric Junction block with Y-shaped conductors

It is also a possibility to explore having separate power sources generating each a single bubble spark underwater so that the sizes can be controlled independently of one another. In this way, it is presumed that the maximum radii of each of the single bubbles will not be reliant on the splitting of electrical energy through the branching off points.

7.3.3 High Pass Filter for PVDF Data

PVDF data are also prone to slight errors due to the slight offset drift of the voltage. After the spark discharge, the whole voltage output tends to be raised by some hundreds of millivolts away from the 0 V level. This is presumed to be caused by the discharge of the capacitor somehow affecting the voltage readings of the submerged PVDF sensor, either through the water or other conductive parts of the experimental setup. Aside from proper grounding of the experimental setup, it is discovered that passing the PVDF waveform can mitigate these effects, as such, the 0 level of the signal will remain consistent at 0 V where it needs to be, thus potentially improving the PVDF signal reading.

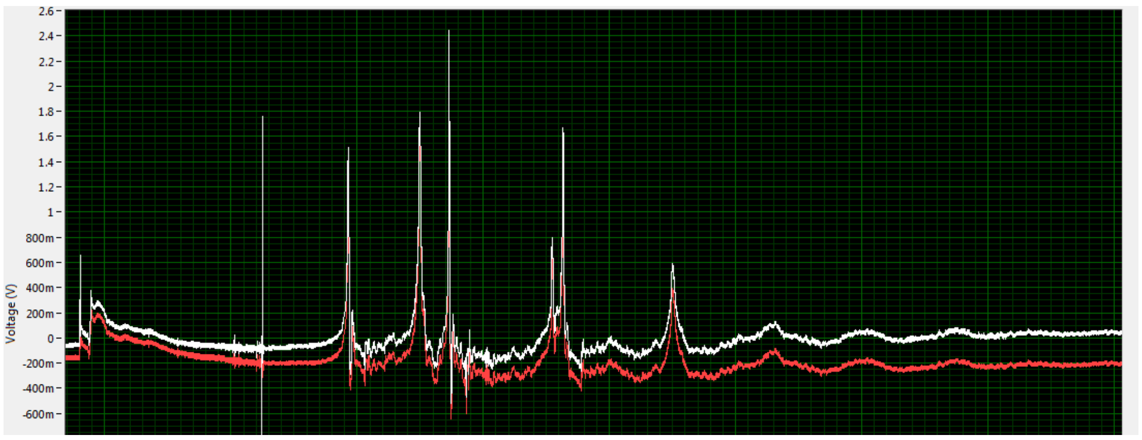


Figure 7.2: Comparison of unfiltered (red) to filtered (white) signal

The phenomenon is pictured in the above figure, upon applying a Butterworth High Pass Filter to the recorded PVDF signal in NI SignalExpress. Notice that the original unfiltered signal is offset at some voltage 200 mV below 0. When the high pass filter is applied (in this example at 100 Hz cutoff), it can be seen that as intended, the response of the film without cavitation levels out at 0 V, which is what we intended (as for this time, only the effect of the cavitation is visible to the sensor).

8 References

1. FRANC, Jean-Pierre; MICHEL, Jean-Marie. *Fundamentals of Cavitation*. Dordrecht: Kluwer Academic Publishers, 2004. ISBN 1-4020-2232-8.
2. HOGENDOORN, Willian. *Cavitation: Experimental investigation of cavitation regimes in a coverging-diverging nozzle*. 2017. Available also from: https://repository.tudelft.nl/islandora/object/uuid:823a18f0-66a8-4ffd-a688-c3dadf62c4da?collection=education. Masters. Delft University of Technology.
3. BRENNEN, Christopher Earls. An Introduction to Cavitation Fundamentals. *WIMRC Forum*. 2011, no. July, pp. 1–17. Available also from: http://authors.library.caltech.edu/28373/.
4. BRENNEN, Christopher Earls. *Cavitation and Bubble Dynamics*. Cavitation and bubble dynamics. 2013. ISBN 9781107338760. Available from DOI: 10.1017/CBO9781107338760.
5. FONG, Siew Wan; ADHIKARI, Deepak; KLASEBŒR, Evert; KHOO, Boo Cheong. Interactions of multiple spark-generated bubbles with phase differences. *Experiments in Fluids*. 2009, vol. 46, no. 4, pp. 705–724. ISSN 07234864. Available from DOI: 10.1007/s00348-008-0603-4.
6. RAYLEIGH, Lord. VIII. On the pressure developed in a liquid during the collapse of a spherical cavity. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*. 1917, vol. 34, no. 200, pp. 94–98. ISSN 1941-5982. Available from DOI: 10.1080/14786440808635681.
7. PLESSET, Milton S; PROSPERETTI, Andrea. Bubble Dynamics and Cavitation. *Annual Review of Fluid Mechanics*. 1977, vol. 9, no. 1, pp. 145.
8. HEGEDUS, Ferenc; KOCH, Sandra; GAREN, Walter; PANDULA, Zoltán; PAÁL, György; KULLMANN, László; TEUBNER, Ulrich. The effect of high viscosity on compressible and incompressible Rayleigh-Plesset-type bubble models. *International Journal of Heat and Fluid Flow*. 2013, vol. 42, pp. 200–

208. ISSN 0142727X. Available from DOI: 10.1016/j.ijheatfluidflow.2013.04.004.

9. WANG, Yi Chun; CHEN, Yu Wen. Application of piezœlectric PVDF film to the measurement of impulsive forces generated by cavitation bubble collapse near a solid boundary. *Experimental Thermal and Fluid Science*. 2007, vol. 32, no. 2, pp. 403–414. ISSN 08941777. Available from DOI: 10.1016/j.expthermflusci.2007.05.003.

10. KIM, Ki-Han; CHAHINE, Georges L.; FRANC, Jean-Pierre; KARIMI, Ayat. *Advanced Experimental and Numerical Techniques for Cavitation Erosion Prediction*. Ed. by KIM, Ki-Han; CHAHINE, Georges L.; FRANC, Jean-Pierre; KARIMI, Ayat. Dordrecht: Springer, 2014. ISBN 978-94-017-8538-9. ISSN 2215-0056. Available from DOI: 10.1007/978-94-017-8539-6.

11. PLESSET, Milton S; MITCHELL, T. P. On the Stability of the Spherical Shape of a Vapor Cavity in a Liquid. *Quarterly of Applied Mathematics*. 1956, vol. 13, no. 4, pp. 419–430. Available also from: http://www.jstor.org/stable/43634276.

12. BREMOND, Nicolas; ARORA, Manish; DAMMER, Stephan M.; LOHSE, Detlef. Interaction of cavitation bubbles on a wall. *Physics of Fluids*. 2006, vol. 18, no. 12. ISSN 10706631. Available from DOI: 10.1063/1.2396922.

13. MAO, Yunfei; PENG, Yong; ZHANG, Jianmin. Study of cavitation bubble collapse near awall by the modified Lattice Boltzmann method. *Water (Switzerland)*. 2018, vol. 10, no. 10, pp. 1–15. ISSN 20734441. Available from DOI: 10.3390/w10101439.

14. MCOR ADVANCED MECHANICAL POLYMERS. *Cavitation Solution* [online] [visited on 2019-04-18]. Available from: http://www.mcor.net/mcorabilities/mcorabilites-for-pumps-fluid-flow/cavitation-solution/.

15. TERAN, Leonel A.; RODRÍGUEZ, Sara A.; LAÍN, Santiago; JUNG, Sunghwan. Interaction of particles with a cavitation bubble near a solid wall. *Physics of Fluids*. 2018, vol. 30, no. 12, pp. 1–10. ISSN 10897666. Available from DOI: 10.1063/1.5063472.

16. MÜLLER, Miloš; HUJER, Jan; KOTEK, M.; ZIMA, Patrik. Identification of collapse patterns of cavitation bubbles close to a solid wall. *EPJ Web of Con-*

ferences. 2013, vol. 45, pp. 01120. Available from DOI: 10.1051/epjconf/20134501120.

17. SHIMA, A.; TAKAYAMA, K.; TOMITA, Y.; OHSAWA, N. Mechanism of impact pressure generation from spark-generated bubble collapse near a wall. *AIAA Journal*. 1983, vol. 21, no. 1, pp. 55–59. ISSN 0001-1452. Available from DOI: 10.2514/3.8027.

18. FRANC, Jean-Pierre. Physics and Control of Cavitation. In: *Design and Analysis of High Speed Pumps*. Neuilly-sur-Seine: RTO, 2006, pp. 2–1 – 2–36. No. 44. Available also from: https://www.sto.nato.int/publications/STO%20Educational%20Notes/RTO-EN-AVT-143/EN-AVT-143-02.pdf.

19. PISHCHALNIKOV, Yuriy A.; SAPOZHNIKOV, Oleg A.; WILLIAMS, James C.; EVAN, Andrew P.; MCATEER, James A.; CLEVELAND, Robin O.; COLONIUS, Tim; BAILEY, Michæl R.; CRUM, Lawrence A. Cavitation bubble cluster activity in the breakage of stones by shock wave lithotripsy. *The Journal of the Acoustical Society of America*. 2013, vol. 111, no. 5, pp. 2461. ISBN 6173535866. ISSN 00014966. Available from DOI: 10.1121/1.4778494.

20. CHU, Po Chun; CHAI, Wen Yen; TSAI, Chih Hung; KANG, Shih Tsung; YEH, Chih Kuang; LIU, Hao Li. Focused Ultrasound-Induced Blood-Brain Barrier Opening: Association with Mechanical Index and Cavitation Index Analyzed by Dynamic Contrast-Enhanced Magnetic-Resonance Imaging. *Scientific Reports*. 2016, vol. 6, no. September, pp. 1–13. ISSN 20452322. Available from DOI: 10.1038/srep33264.

21. DULAR, Matevž et al. Use of hydrodynamic cavitation in (waste)water treatment. *Ultrasonics Sonochemistry*. 2016, vol. 29, pp. 577–588. ISSN 18732828. Available from DOI: 10.1016/j.ultsonch.2015.10.010.

22. GONZÁLEZ-GARCÍA, José; SÁEZ, Verónica; TUDELA, Ignacio; DÍEZ-GARCIA, María Isabel; ESCLAPEZ, María Deseada; LOUISNARD, Olivier. Sonochemical treatment of water polluted by chlorinated organocompounds. A review. *Water (Switzerland)*. 2010, vol. 2, no. 1, pp. 28–74. ISSN 20734441. Available from DOI: 10.3390/w2010028.

23. KHOO, Boo Cheong; ADHIKARI, Deepak; FONG, Siew Wan; KLASEBŒR, Evert. Multiple Spark-Generated Bubble Interactions. *Modern Physics Letters B*. 2009, vol. 23, no. 03, pp. 229–232. ISSN 0217-9849. Available from DOI: 10.1142/s0217984909018072.

24. TURANGAN, C. K.; ONG, G. P.; KLASEBŒR, Evert; KHOO, Boo Cheong. Experimental and numerical study of transient bubble-elastic membrane interaction. *Journal of Applied Physics*. 2006, vol. 100, no. 5. ISSN 00218979. Available from DOI: 10.1063/1.2338125.

25. LEW, Kelly Siew Fong; KLASEBŒR, Evert; KHOO, Boo Cheong. A collapsing bubble-induced micropump: An experimental study. *Sensors and Actuators, A: Physical*. 2007, vol. 133, no. 1, pp. 161–172. ISSN 09244247. Available from DOI: 10.1016/j.sna.2006.03.023.

26. BUOGO, Silvano; CANNELLI, Giovanni B. Implosion of an underwater spark-generated bubble and acoustic energy evaluation using the Rayleigh model. *The Journal of the Acoustical Society of America*. 2002, vol. 111, no. 6, pp. 2594–2600. ISSN 0001-4966. Available from DOI: 10.1121/1.1476919.

27. GOH, Bing Hui Terence; OH, Y. D. A.; KLASEBŒR, Evert; OHL, Siew-Wan; KHOO, Boo Cheong. A low-voltage spark-discharge method for generation of consistent oscillating bubbles. *Review of Scientific Instruments*. 2013, vol. 84, no. 1. ISSN 00346748. Available from DOI: 10.1063/1.4776187.

28. JI, Chen; LI, Bo; ZOU, Jun; YANG, Huayong. Interaction of two spark generated bubbles beneath free surface. *Experimental Thermal and Fluid Science*. 2017, vol. 81, pp. 76–83. ISSN 08941777. Available from DOI: 10.1016/j.expthermflusci.2016.10.007.

29. LIU, N. N.; CUI, P.; REN, S. F.; ZHANGA, A. M. Study on the interactions between two identical oscillation bubbles and a free surface in a tank. *Physics of Fluids*. 2017, vol. 29, no. 5, pp. 052104. ISSN 10897666. Available from DOI: 10.1063/1.4984080.

30. LUO, Jing; NIU, Zhipan. Jet and Shock Wave from Collapse of Two Cavitation Bubbles. *Scientific Reports*. 2019, vol. 9, no. 1, pp. 1–13. ISBN 4159801837868. ISSN 20452322. Available from DOI: 10.1038/s41598-018-37868-x.

31. SZALA, Mirosław. Application of computer image analysis software for determining incubation period of cavitation erosion – preliminary results. *ITM Web of Conferences*. 2017, vol. 15, no. December, pp. 06003. Available from DOI: 10.1051/itmconf/20171506003.

32. LV, Dongli; LIAN, Zhanghua; ZHANG, Tao. Study of Cavitation and Cavitation Erosion Quantitative Method Based on Image Processing Technique.

Advances in Civil Engineering. 2018, vol. 2018, pp. 1–10. ISSN 1687-8086. Available from DOI: 10.1155/2018/5317578.

33. DEL IMAGING. *Redlake MotionPro HS Series* [online]. 2019 [visited on 2019-04-09]. Available from: http://www.delimaging.com/camera/redlake-motionpro-hs-series/.

34. KIM, Tæ Hoon; ARIAS, Ana Claudia. *Characterization and applications of piezoelectric polymers*. Berkeley, 2015. Available also from: http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-253.html.

35. ROOPAA, T. S.; NARASIMHA MURTHY, H. N.; PRAVEEN KUMAR, V. V.; KRISHNA, M. Development and Characterization of PVDF Thin Films for pressure sensors. *Materials Today: Proceedings*. 2018, vol. 5, no. 10, pp. 21082–21090. ISSN 22147853. Available from DOI: 10.1016/j.matpr.2018.6.503.

36. TE CONNECTIVITY. *DT Sensors - Specifications*. TE Connectivity Sensor Solutions, 2015. Available also from: http://www.farnell.com/datasheets/2310782.pdf.

37. HUJER, Jan; MÜLLER, Miloš. Calibration of PVDF Film Transducers for the Cavitation Impact Measurement. *EPJ Web of Conferences*. 2018, vol. 180, pp. 02036. Available from DOI: 10.1051/epjconf/201818002036.

38. RONGALA, Arvind. *Blog - Invensis Technologies*. Benefits of C / C++ over Other Programming Languages [online]. 2015 [visited on 2019-04-09]. Available from: https://www.invensis.net/blog/it/benefits-of-c-c-plus-plus-over-other-programming-languages/.

39. URDHWARESHE, Ashwin. Object-Oriented Programming and its Concepts. *International Journal of Innovation and Scientific Research*. 2016, vol. 26, no. 1, pp. 1–6. Available also from: http://www.ijisr.issr-journals.org/.

40. BRADSKI, Gary. The OpenCV library. *Dr. Dobb's Journal of Software Tools*. 2000, vol. 25, no. 120, pp. 122–125. Available also from: http://www.drdobbs.com/open-source/the-opencv-library/184404319.

41. AKIMOV, Pavel; BELYAK, Alexandr. *SourceForge.net*. SimpleXlsxWriter - C++ library for creating XLSX files for MS Excel 2007 and above. [online]. 2012 [visited on 2019-04-24]. Available from: https://sourceforge.net/projects/simplexlsx/.

A Sensor Equipment Settings

The settings used for the sensor equipment from the experimental setup are described in the following appendix.

A.1 Camera Settings

The high-speed camera data is read from the MotionPro X Studio software. In order to get the clearest images from the camera, calibration is first performed. A calibration operation is present in this software.

The camera settings can be configured through a console in MotionPro X Studio so that the bubble can be recorded properly. The ROI is manually set in a separate window into a size small enough to accommodate a 10 kHz sampling rate (specifically 148 pixels by 100 pixels) and make the bubble and wall area visible. The other settings are as follows:

- `Sensor Gain` – No Gain
- `Rate (Hz)` – 10000
- `Exposure (`μ`s)` – 15
- `Exposure Mode` – Single
- `Binning` – 1x1
- `Pixel Depth` – 8 Bit
- `ROI` – Manually set
- `Live Timeout (s)` – 5.000
- `Sample Rate (S/s)` – 10M
- `Record Length` – 120000

- `Acquire` – N Samples

The recording settings are configured on the next tab such that the entire lifetime of the bubble is sufficiently observed and recorded, as follows:

- `Record Mode` – Normal
- `Frames` – 140
- `Frame Sync` – External
- `Sync Cfg` – Edge-High
- `Trigger Cfg` – Edge-High

The software will then be able to save a collection of 140 8-bit TIFF images taken at 10000 FPS within a folder it automatically creates upon definition by the user. One of the images in this nominated folder is then used as input by the analysis tool, which will then measure the relevant images which are at the same location as itself.

A.2 PVDF Sensor DAQ Settings

The PVDF data as mentioned earlier are acquired from the NI PXI-5105 card to the NI SignalExpress software. Two channels of the PXI-5105 are used; Channel 0 for the 3 V trigger input from synchronization, and Channel 1 for the output of the PVDF sensor connected here. The following are the settings used for data acquisition in order to acquire the proper lifetime of the bubble, from inception to continued collapses, after the spark:

- `Device` – PXI-5105
- `Channel(s)` – Channel 0 and Channel 7
- `Range (V)` – 6 (Channel 0), 30 (Channel 7)
- `Offset (V)` – 0 (Channels 0 and 7)
- `Probe attenuation` – 1 (Channels 0 and 7)
- `Coupling` – DC (Channels 0 and 7)
- `Input Impedance` – 1 Mohm (Channels 0 and 7)

- `Bandwidth (Hz)` – 0 (Channels 0 and 7)
- `Sample Rate (S/s)` – 60M
- `Record Length` – 120000
- `Acquire` – N Samples

A five millisecond delay is provided from an experimental observation that at this time, the bubble will only start development after the spark has occurred. Thus, the trigger from the Trigger tab is configured as such:

- `Type` – Edge
- `Source` – Channel 0
- `Slope` – Positive
- `Coupling` – DC
- `Ref position` – 0
- `Level (V)` – 0
- `Max time (s)` – 10
- `Delay (s)` – 5m

Upon successful running of the test, a waveform of Channel 1 information sampled at 60 MHz can be obtained at the Display Pane of SignalExpress. There are options to export the information as an Excel file (which can be converted as a CSV file later for input to the analysis tool) or as a collection of data in the clipboard to be pasted later.

B Code Snippets

B.1 CavTools Main Module

Listing B.1: CavTools.cpp

```
//==========================================================
// Name        : CavTools.cpp
// Author      : R. S. Salonga
// Version     : v1.2
// Copyright   : 30.08.2018
// Description : CavTools Main program
//==========================================================

#include "Common/includes.h"
#include "Common/defines.h"

int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int
    nCmdShow )
{
    OPENFILENAME ofn;
    char szFile[200];

    // open a file name
    ZeroMemory(&ofn , sizeof(ofn));
    ofn.lStructSize = sizeof(ofn);
    ofn.hwndOwner = NULL;
    ofn.lpstrFile = szFile ;
    ofn.lpstrFile[0] = '\0';
    ofn.nMaxFile = sizeof(szFile);
    ofn.lpstrFilter = "TIFF Files\0*.tif\0";
    ofn.nFilterIndex = 1;
    ofn.lpstrFileTitle = NULL;
    ofn.nMaxFileTitle = 0 ;
    ofn.lpstrInitialDir = NULL;
    ofn.Flags = OFN_PATHMUSTEXIST | OFN_FILEMUSTEXIST;

    if (GetOpenFileName(&ofn))
    {
        // Get the file path location
        string openPath = ofn.lpstrFile;
        string work_dir = openPath.substr(0, openPath.size() - 14); // remove ImgA000000
            .tif from filename
        cout << work_dir << endl;
        // Ask if the setup is of Single or Double Bubble
        int answer = AskSingleBubble();
        cout << "[INFO] " << answer << " bubble configuration selected." << endl;

        // Build the images
        cout << "[INFO] Loading bubble set from Camera data..." << endl;
        Mat k = buildBubbleFrame(ofn.lpstrFile);
        Mat j[NUM_OF_CAMERA_POINTS];
        buildBubbleSet(j, INITIAL_FRAME_CAMERA, NUM_OF_CAMERA_POINTS, ofn.lpstrFile);

        cout << "[INFO] Attempting to locate wall..." << endl;
        int wallLoc = 80;
        Mat j2[NUM_OF_CAMERA_POINTS];

        Mat drawing = Mat::zeros( k.size(), CV_8UC3 );

        // Prepare the stats object
        BubbleStats bstats;
        bstats.fsize = NUM_OF_CAMERA_POINTS;

        // Read a safe bubble (from a middle frame)
        cout << "[INFO] Reading a safe bubble location..." << endl;
        Mat safe = j[SAFE_BUBBLE_IDX];
        vector<vector<Point>> cont_safe;
        vector<Vec4i> hier_safe;
        int idx1_safe, idx2_safe;
        bubbleDetectFeatures(safe, &cont_safe, &hier_safe);
        idx1_safe = bubbleIsolate(safe, &cont_safe);
        if (answer == 2) {
            idx2_safe = bubble2Isolate(safe, &cont_safe, idx1_safe);
            BubblePair b_safe = set2BubbleIndices(&cont_safe, idx1_safe, idx2_safe);
            idx1_safe = b_safe.left;
            idx2_safe = b_safe.right;
        }

        int b1_safe_cX, b1_safe_cY, b2_safe_cX, b2_safe_cY;
        cout << "SAMPLING MIDDLE FRAME:" << endl;
        cout << "B1 Pos X: " << computePositionX(&cont_safe, idx1_safe) << endl;
        cout << "B1 Pos Y: " << computePositionY(&cont_safe, idx1_safe) << endl;
        cout << "B1 Area: " << computeArea(&cont_safe, idx1_safe) << endl;
        cout << "B1 Perim: " << computePerimeter(&cont_safe, idx1_safe) << endl;
        cout << "B1 Radius: " << computeRadius(&cont_safe, idx1_safe) << endl;
        cout << "B1 Circularity: " << computeCircularity(&cont_safe, idx1_safe) << endl;
        cout << "B1 Dist from Wall: " << (computePositionY(&cont_safe, idx1_safe) -
            wallLoc) << endl;
        if (answer == 2) {
            cout << "B2 Pos X: " << computePositionX(&cont_safe, idx2_safe) << endl;
            cout << "B2 Pos Y: " << computePositionY(&cont_safe, idx2_safe) << endl;
            cout << "B2 Area: " << computeArea(&cont_safe, idx2_safe) << endl;
            cout << "B2 Perim: " << computePerimeter(&cont_safe, idx2_safe) << endl;
            cout << "B2 Radius: " << computeRadius(&cont_safe, idx2_safe) << endl;
            cout << "B2 Circularity: " << computeCircularity(&cont_safe, idx2_safe) <<
                endl;
            cout << "B2 Dist from Wall: " << (computePositionY(&cont_safe, idx2_safe) - wallLoc) << endl;
            b2_safe_cX = computePositionX(&cont_safe, idx2_safe);
            b2_safe_cY = computePositionY(&cont_safe, idx2_safe);
        }

        b1_safe_cX = computePositionX(&cont_safe, idx1_safe);
        b1_safe_cY = computePositionY(&cont_safe, idx1_safe);

        for (int i = 0; i < NUM_OF_CAMERA_POINTS; i++) {
            cout << "PROCESSING FRAME #" << i << endl;
            bstats.ftime.push_back(i);

            vector<vector<Point>> contours;
            vector<Vec4i> hierarchy;

            j2[i] = bubbleWallDraw(j[i], wallLoc);

            bubbleDetectFeatures(j2[i], &contours, &hierarchy);
            int idx = bubble1Isolate_CenteredAt(j2[i], &contours, b1_safe_cX, b1_safe_cY);
            int idx2;
            if (answer == 2) idx2 = bubble2Isolate_CenteredAt(j2[i], &contours, b2_safe_cX, b2_safe_cY, idx);
            else idx2 = -1;

            if ((idx > -1) && (idx2 > -1))
            {
                BubblePair b;
                b = set2BubbleIndices(&contours, idx, idx2);
                idx = b.left;
                idx2 = b.right;
            }
            else if (idx == -1)
            {
                cout << "There are no bubbles found." << endl;
            }
            else if (idx2 == -1)
            {
                cout << "The 2nd bubble is not found." << endl;
            }

            Scalar color = Scalar(0,0,255);
            {
                if (idx > -1) {
                    drawContours( drawing, contours, idx, color, 1, 8, hierarchy, 0, Point() );
                    bstats.bubble1Area.push_back(computeArea(&contours, idx));
                    bstats.bubble1Perimeter.push_back(computePerimeter(&contours, idx));
                    bstats.bubble1Radius.push_back(computeRadius(&contours, idx));
                    bstats.bubble1Circularity.push_back(computeCircularity(&contours, idx));
                    bstats.bubble1PosX.push_back(computePositionX(&contours, idx));
                    bstats.bubble1PosY.push_back(computePositionY(&contours, idx));
```

```cpp
                    bstats.bubble1DistanceFromWall.push_back(abs(computePositionY(&contours, idx) - wallLoc));
                }
                else {
                    bstats.bubble1Area.push_back(-1);
                    bstats.bubble1Perimeter.push_back(-1);
                    bstats.bubble1Radius.push_back(-1);
                    bstats.bubble1Circularity.push_back(-1);
                    bstats.bubble1PosX.push_back(-1);
                    bstats.bubble1PosY.push_back(-1);
                    bstats.bubble1DistanceFromWall.push_back(-1);
                }
            }
            Scalar color2 = Scalar(255,0,0);
            {
                if (idx2 > -1) {
                    drawContours( drawing, contours, idx2, color2, 1, 8, hierarchy, 0, Point() );
                    bstats.bubble2Area.push_back(computeArea(&contours, idx2));
                    bstats.bubble2Perimeter.push_back(computePerimeter(&contours, idx2));
                    bstats.bubble2Radius.push_back(computeRadius(&contours, idx2));
                    bstats.bubble2Circularity.push_back(computeCircularity(&contours, idx2));
                    bstats.bubble2PosX.push_back(computePositionX(&contours, idx2));
                    bstats.bubble2PosY.push_back(computePositionY(&contours, idx2));
                    bstats.bubble2DistanceFromWall.push_back(abs(computePositionY(&contours, idx2) - wallLoc));
                }
                else {
                    bstats.bubble2Area.push_back(-1);
                    bstats.bubble2Perimeter.push_back(-1);
                    bstats.bubble2Radius.push_back(-1);
                    bstats.bubble2Circularity.push_back(-1);
                    bstats.bubble2PosX.push_back(-1);
                    bstats.bubble2PosY.push_back(-1);
                    bstats.bubble2DistanceFromWall.push_back(-1);
                }
            }
        }

        saveSheet(bstats, answer);
        imshow( "Result window", drawing );
    }
    else
    {
        MessageBox ( NULL , "Sorry, I ran into some error :(" , "Wait a sec..." , MB_OK);
    }

    waitKey(0);
    return 0;
}
```

B.2 bload Module

Listing B.2: AutoBubble/bload.h

```cpp
#ifndef AUTOBUBBLE_BLOAD_H_
#define AUTOBUBBLE_BLOAD_H_

#include <opencv2/opencv.hpp>
#include <opencv/highgui.h>

using namespace cv;
using namespace std;

string buildBubbleFileNames(string bubbleSetName, int index);
void buildBubbleSet(Mat bs[], int bsStart, int bsSize, string bsFileName);
Mat buildBubbleFrame(string bsFileName);

#endif /* AUTOBUBBLE_BLOAD_H_ */
```

Listing B.3: AutoBubble/bload.cpp

```cpp

#include "AutoBubble/bload.h"

string buildBubbleFileNames(string bubbleSetName, int index)
{
    string fileName;
    if (index < 10)
        { fileName = bubbleSetName + "00000" + to_string(index) + ".tif"; }
    else if (index < 100 && index >= 10)
        { fileName = bubbleSetName + "0000" + to_string(index) + ".tif"; }
    else if (index < 1000 && index >= 100)
        { fileName = bubbleSetName + "000" + to_string(index) + ".tif"; }
    else if (index < 10000 && index >= 1000)
        { fileName = bubbleSetName + "00" + to_string(index) + ".tif"; }
    return fileName;
}

void buildBubbleSet(Mat bs[], int bsStart, int bsSize, string bsFileName)
{
    string bsFileNameTrunc = bsFileName.substr(0, bsFileName.size() - 10);  // remove
        000000.tif from filename
    string bsFileNameNew;
    for (int i = bsStart; i < bsStart + bsSize; i++) {
        bsFileNameNew = buildBubbleFileNames(bsFileNameTrunc, i);
        cout << "[INFO] Loading bubble frame file " << bsFileNameNew << endl;
        bs[i-bsStart] = imread(bsFileNameNew);
    }
    cout << "Loading complete." << endl;
}

Mat buildBubbleFrame(string bsFileName) { return imread(bsFileName); }
```

B.3 bread Module

Listing B.4: AutoBubble/bread.h

```cpp
#ifndef AUTOBUBBLE_BREAD_H_
#define AUTOBUBBLE_BREAD_H_

#include <opencv2/opencv.hpp>
#include <opencv/highgui.h>

using namespace cv;

Mat bubbleWall(Mat img);
int bubbleWallBounds(Mat img);
Mat bubbleWallDraw(Mat img, int wallLocation);
void bubbleDetectFeatures(Mat img, vector<vector<Point>> * cntr, vector<Vec4i> * hier);
int bubbleIsolate(Mat img, vector<vector<Point>> * cntr);
int bubble2Isolate(Mat img, vector<vector<Point>> * cntr, int bubble1Index);
int bubble1Isolate_CenteredAt(Mat img, vector<vector<Point>> * cntr, int center1X, int
    center1Y);
int bubble2Isolate_CenteredAt(Mat img, vector<vector<Point>> * cntr, int center1X, int
    center1Y, int bubble1Index);

#endif /* AUTOBUBBLE_BREAD_H_ */
```

Listing B.5: AutoBubble/bread.cpp

```cpp
#include "AutoBubble/bread.h"
#include "AutoBubble/bcomps.h"

#include "Common/defines.h"

Mat bubbleWall(Mat img)
{
    Mat step1, step2, step3;
    cvtColor(img, step1, CV_BGR2GRAY);
    blur( step1, step1, Size(3,3) );
    threshold(step1, step2, 50, 255, THRESH_BINARY);    // threshold for contours
    step3 = img.clone();

    vector<vector<Point>> contours;
    vector<Vec4i> hierarchy;
    findContours(step2, contours, hierarchy, CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE, Point
        (0,0));
    Rect wallBounds = boundingRect(contours.at(1));

    Point pt1, pt2;
        pt1.x = 0;
        pt1.y = wallBounds.height;
        pt2.x = step3.size().width;
        pt2.y = wallBounds.height;
    line(step3, pt1, pt2, cv::Scalar(255, 255, 255), 2, CV_AA, 0);
    return step3;
}
```

```

int bubbleWallBounds(Mat img)
{
    Mat step1, step2, step3;
    cvtColor(img, step1, CV_BGR2GRAY);
    blur( step1, step1, Size(3,3) );
    threshold(step1, step2, 50, 255, THRESH_BINARY);    // threshold for contours
    step3 = img.clone();

    vector<vector<Point>> contours;
    vector<Vec4i> hierarchy;
    findContours(step2, contours, hierarchy, CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE, Point
        (0,0));
    Rect wallBounds = boundingRect(contours.at(1));
    return wallBounds.height;
}

Mat bubbleWallDraw(Mat img, int wallLocation)
{
    Mat imgNew = img.clone();
    Point pt1, pt2;
        pt1.x = 0;
        pt1.y = wallLocation;
        pt2.x = imgNew.size().width;
        pt2.y = wallLocation;
    line(imgNew, pt1, pt2, cv::Scalar(255, 255, 255), 1, CV_AA, 0);
    return imgNew;
}

void bubbleDetectFeatures(Mat img, vector<vector<Point>> * cntr, vector<Vec4i> * hier)
{
    Mat step1, step2, step3;
    cvtColor(img, step1, CV_BGR2GRAY);
    blur( step1, step1, Size(3,3) );
    threshold(step1, step2, 65, 255, THRESH_BINARY);    // threshold for circles
    threshold(step1, step3, 42, 255, THRESH_BINARY);    // threshold for contours
    Mat thresh = step2.clone();
    Mat thresh2 = step3.clone();
    Mat dwg, dwg2;
    cvtColor(thresh2, dwg, CV_GRAY2BGR);

    vector<vector<Point>> contours;
    vector<Vec4i> hierarchy;

    // Contours
    findContours(thresh2, contours, *(hier), CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE, Point
        (0,0));
    Scalar color = Scalar(0,0,255);
    for (int c = 0; c < (int)contours.size(); c++ ) {
        drawContours( dwg, contours, c, color, 1, 8, hierarchy, 0, Point() ); }

    findContours(step3, *(cntr), *(hier), CV_RETR_TREE, CV_CHAIN_APPROX_SIMPLE, Point
        (0,0));
}

```

```cpp
int bubbleIsolate(Mat img, vector<vector<Point>> * cntr)
{
    // Isolate the largest contour which is reasonably sized and "circular"
    int bubble1Index = -1;
    vector<vector<Point>> contour = *(cntr);

    int c = BUBBLE_CNTR_IDX_START;

    while (c < (int)contour.size()) {
        if (arcLength(contour.at(c), true) <= MAX_BUBBLE_SIZE) {
            if (computeCircularity(&contour, c) < CIRCULARITY_THRESHOLD) {
                if (bubble1Index == -1) {
                    bubble1Index = c;
                    c++;
                } else {
                    if ((arcLength(contour.at(c), true)) > (arcLength(contour.at(
                        bubble1Index), true))) {
                        bubble1Index = c; c++;
                    } else { c++; }
                }
            } else { c++; }
        } else { c++; }
    }
    return bubble1Index;
}

int bubble2Isolate(Mat img, vector<vector<Point>> * cntr, int bubble1Index)
{
    // Isolate the 2nd largest contours (2 bubbles)
    if (bubble1Index != -1) {
        int bubble2Index = -1;
        vector<vector<Point>> contour = *(cntr);

        int d = BUBBLE_CNTR_IDX_START;

        while (d < (int)contour.size()) {
            if (d != bubble1Index) {
                if (arcLength(contour.at(d), true) <= MAX_BUBBLE_SIZE) {
                    if (computeCircularity(&contour, d) < CIRCULARITY_THRESHOLD) {
                        if (bubble2Index == -1) {
                            bubble2Index = d; d++;
                        } else {
                            if ((arcLength(contour.at(d), true)) > (arcLength(contour.at
                                (bubble2Index), true))) {
                                bubble2Index = d; d++;
                            } else { d++; }
                        }
                    } else { d++;}
                } else { d++; }
            } else { d++; }
        }
        return bubble2Index;
    }
    else return -1;
}
```

```

int bubble1Isolate_CenteredAt(Mat img, vector<vector<Point>> * cntr, int center1X, int
     center1Y)
{
    // Isolate the largest contour which follows these criteria:
    // 1) Is reasonably sized
    // 2) Is reasonably "circular"
    // 3) Is reasonably "near" to the given center
    int bubble1Index = -1;
    vector<vector<Point>> contour = *(cntr);

    int c = BUBBLE_CNTR_IDX_START;

    while (c < (int)contour.size()) {
        Rect bBound = boundingRect(contour.at(c));
        int cX = bBound.x + bBound.width/2;

        // Check Criterion 1
        if (arcLength(contour.at(c), true) <= MAX_BUBBLE_SIZE) {
            // Check Criterion 2
            if (computeCircularity(&contour, c) < CIRCULARITY_THRESHOLD) {
                // Check Criterion 3
                if ((abs(cX-center1X) < BUBBLE_SEARCH_RADIUS)) {
                    // No other bubbles have been found yet
                    if (bubble1Index == -1) {
                        bubble1Index = c;
                        c++;
                    } else {
                        // If there is a bubble already existing
                        if ((arcLength(contour.at(c), true)) > (arcLength(contour.at(
                            bubble1Index), true))) {
                            // This current one is bigger, move the index to here
                            bubble1Index = c;
                            c++;
                        } else {
                            // The previous index is still bigger
                            c++;
                        }
                    }
                } else {
                    // Criterion 3 failed, too far from the other bubbles!
                    c++;
                }
            } else {
                // Criterion 2 failed, bubble is not circular enough!
                c++;
            }
        } else {
            // Criterion 1 failed, bubble is too big!
            c++;
        }
    }
    return bubble1Index;
}

```

```
int bubble2Isolate_CenteredAt(Mat img, vector<vector<Point>> * cntr, int center1X, int
    center1Y, int bubble1Index)
{
    // Isolate the 2nd largest contours (2 bubbles)
    if (bubble1Index != -1) {
        int bubble2Index = -1;
        vector<vector<Point>> contour = *(cntr);

        int d = BUBBLE_CNTR_IDX_START;

        // Isolate the largest contour which follows these criteria:
        // 1) Is reasonably sized
        // 2) Is reasonably "circular"
        // 3) Is reasonably "near" to the given center
        // AND IS NOT chosen already as Bubble 1 index!
        while (d < (int)contour.size()) {
            Rect bBound = boundingRect(contour.at(d));
            int cX = bBound.x + bBound.width/2;

            if (d != bubble1Index) {
                if (arcLength(contour.at(d), true) <= MAX_BUBBLE_SIZE) {
                    if (computeCircularity(&contour, d) < CIRCULARITY_THRESHOLD) {
                        if ((abs(cX-center1X) < BUBBLE_SEARCH_RADIUS)) {
                            if (bubble2Index == -1) {
                                bubble2Index = d;
                                d++;
                            } else {
                                if ((arcLength(contour.at(d), true)) > (arcLength(
                                    contour.at(bubble2Index), true))) {
                                    bubble2Index = d;
                                    d++;
                                } else {
                                    d++;
                                }
                            }
                        } else {
                            d++;
                        }
                    } else {
                        d++;
                    }
                } else {
                    d++;
                }
            } else {
                d++;
            }
        }
        return bubble2Index;
    }
    else return -1;
}
```

B.4 bcomps Module

Listing B.6: AutoBubble/bcomps.h

```
#ifndef AUTOBUBBLE_BCOMPS_H_
#define AUTOBUBBLE_BCOMPS_H_

#include <opencv2/opencv.hpp>
#include <opencv/highgui.h>
#include "Common/defines.h"
#include "Common/includes.h"

using namespace cv;

struct BubblePair { int left; int right; };
struct BubbleStats {
    int fsize;
    vector<int> ftime;
    vector<double> ftime1_us;
    vector<double> bubble1Area;
    vector<double> bubble1Perimeter;
    vector<double> bubble1Radius;
    vector<double> bubble1Radius_mm;
    vector<double> bubble1Circularity;
    vector<double> bubble1PosX;
    vector<double> bubble1PosY;
    vector<double> bubble1DistanceFromWall;
    vector<double> bubble1DistanceFromWall_mm;
    vector<double> bubble1Reliability;
    vector<double> ftime2_us;
    vector<double> bubble2Area;
    vector<double> bubble2Perimeter;
    vector<double> bubble2Radius;
    vector<double> bubble2Radius_mm;
    vector<double> bubble2Circularity;
    vector<double> bubble2PosX;
    vector<double> bubble2PosY;
    vector<double> bubble2DistanceFromWall;
    vector<double> bubble2DistanceFromWall_mm;
    vector<double> bubble2Reliability;
};
extern BubbleStats bstats;

int computePositionX(contourField * cntr, int idx);
int computePositionY(contourField * cntr, int idx);
double computeArea(contourField * cntr, int idx);
double computePerimeter(contourField * cntr, int idx);
double computeRadius(contourField * cntr, int idx);
double computeCircularity(contourField * cntr, int idx);
double computeReliability(contourField * cntr, int idx);
double computeDistanceFromWall(contourField * cntr, int idx, int wall);

double computeCollapseTime(BubbleStats bstats, int bubbleID);
double computeEstimatedTimeStart(BubbleStats bstats, int bubbleID);
double computeEstimatedTimeMax(BubbleStats bstats, int bubbleID);
double computeEstimatedTimeCollapse1(BubbleStats bstats, int bubbleID);
double computeEstimatedTimeCollapse2(BubbleStats bstats, int bubbleID);
BubblePair set2BubbleIndices(contourField * cntr, int bubble1, int bubble2);

// Functions for bsave
int saveSheet(BubbleStats bstats, int answer);

#endif /* AUTOBUBBLE_BCOMPS_H_ */
```

Listing B.7: AutoBubble/bcomps.cpp

```
#include "AutoBubble/bcomps.h"

#include "Common/defines.h"

double computeArea(contourField * cntr, int idx)
{
    contourField contour_int = *(cntr);
    if (idx > -1) return contourArea(contour_int.at(idx), false);
    else return 0;
}

double computePerimeter(contourField * cntr, int idx)
{
    contourField contour_int = *(cntr);
    if (idx > -1) return arcLength(contour_int.at(idx), true);
    else return 0;
}

double computeRadius(contourField * cntr, int idx)
{
    return sqrt(computeArea(cntr, idx)/(M_PI));
}

double computeCircularity(contourField * cntr, int idx)
{
    return (pow(computePerimeter(cntr, idx),2))/(computeArea(cntr, idx)*4*M_PI);
}

double computeReliability(contourField * cntr, int idx)
{
    contourField contour_int = *(cntr);
    Rect bblBounds          = boundingRect(contour_int.at(idx));
    double bBoundWidth      = bblBounds.width;
    double bBoundHeight     = bblBounds.height;
    double bBoundArea       = bBoundWidth * bBoundHeight * BUBBLE_BOUNDREC_APPROX_FACTOR
        ;
    return (100 / (min((pow(arcLength(contour_int.at(idx), true),2)) /
            (contourArea(contour_int.at(idx), false)*4*M_PI),
            (pow((2*bBoundWidth + 2*bBoundHeight),2))/(bBoundArea*4*M_PI))));
}

BubblePair set2BubbleIndices(contourField * cntr, int bubble1, int bubble2)
{
    BubblePair bp;
    contourField contour_int = *(cntr);

    Rect bblBounds1     = boundingRect(contour_int.at(bubble1));
    double bCenterX1    = bblBounds1.x + bblBounds1.width/2;

    Rect bblBounds2     = boundingRect(contour_int.at(bubble2));
    double bCenterX2    = bblBounds2.x + bblBounds2.width/2;

    if (bCenterX1 < bCenterX2) {
        bp.left = bubble1; bp.right = bubble2;
    } else {
        bp.left = bubble2; bp.right = bubble1;
    }
    return bp;
}

int computePositionX(contourField * cntr, int idx)
{
    contourField contour_int = *(cntr);
    Rect bblBounds = boundingRect(contour_int.at(idx));
    return (bblBounds.x + bblBounds.width/2);
}

int computePositionY(contourField * cntr, int idx)
{
    contourField contour_int = *(cntr);
    Rect bblBounds = boundingRect(contour_int.at(idx));
    return (bblBounds.y + bblBounds.height/2);
}

double computeCollapseTime(BubbleStats bstats, int bubbleID)
{
    // Calculate collapse time
    int idx_collapse = 0;
    int idx_max_radius = 0;
    double curr_r = 0;
    double delta_r = 0;

    for (int i = 0; i < bstats.fsize; i++)
    {
        // If circularity is high, we are most likely past the first bubble collapse
        // Adding this line to weed out weird later radius values
        // if (stats_int.bubbleCircularity.at(i) > 2) { break; }

        // Find the maximum radius
        if (bubbleID == 1)
        {
            if (bstats.bubble1Radius.at(i) > curr_r)
            {
                idx_max_radius = i;
                curr_r = bstats.bubble1Radius.at(i);
            }
        }
        else if (bubbleID == 2)
        {
            if (bstats.bubble2Radius.at(i) > curr_r)
            {
                idx_max_radius = i;
                curr_r = bstats.bubble2Radius.at(i);
            }
        }
    }

    for (int i = idx_max_radius; i < bstats.fsize; i++)
    {
        // Start from the maximum radius and see where the collapse is
        if (i > idx_max_radius)
        {
            if (bubbleID == 1)
                delta_r = bstats.bubble1Radius.at(i) - bstats.bubble1Radius.at(i-1);
            else if (bubbleID == 2)
                delta_r = bstats.bubble2Radius.at(i) - bstats.bubble2Radius.at(i-1);
        }
        if (delta_r > 0) break;
        else idx_collapse = i;
    }
    double t_collapse = (double)((double)bstats.ftime.at(idx_collapse) / (double)
        FRAMERATE_CAMERA)
        + (double)(INITIAL_TIME_CAMERA) + (double)(DELAY_CAMERA_SECS);
    double t_max_radius = (double)((double)bstats.ftime.at(idx_max_radius) / (double)
        FRAMERATE_CAMERA)
            + (double)(INITIAL_TIME_CAMERA) + (double)(DELAY_CAMERA_SECS);
    double bubbleCollapseTime = t_collapse - t_max_radius;
    return bubbleCollapseTime;
}

double computeEstimatedTimeStart(BubbleStats bstats, int bubbleID)
{
    int idx_start = 0;

    for (int i = 0; i < bstats.fsize; i++)
    {
        // Find the maximum radius
        if (bubbleID == 1)
        {
            if (bstats.bubble1Radius.at(i) > -1)
            {
                idx_start = i;
                break;
            }
        }
        else if (bubbleID == 2)
        {
            if (bstats.bubble2Radius.at(i) > -1)
            {
                idx_start = i;
                break;
            }
        }
```

```cpp
    }
    double t_start = (double)((double)bstats.ftime.at(idx_start) / (double)
        FRAMERATE_CAMERA)
            + (double)(INITIAL_TIME_CAMERA) + (double)(DELAY_CAMERA_SECS);
    return t_start;
}

double computeEstimatedTimeMax(BubbleStats bstats, int bubbleID)
{
    // Calculate collapse time
    int idx_max_radius = 0;
    double curr_r = 0;

    for (int i = 0; i < bstats.fsize; i++)
    {
        // If circularity is high, we are most likely past the first bubble collapse
        // Adding this line to weed out weird later radius values
        // if (stats_int.bubbleCircularity.at(i) > 2) { break; }

        // Find the maximum radius
        if (bubbleID == 1)
        {
            if (bstats.bubble1Radius.at(i) > curr_r)
            {
                idx_max_radius = i;
                curr_r = bstats.bubble1Radius.at(i);
            }
        }
        else if (bubbleID == 2)
        {
            if (bstats.bubble2Radius.at(i) > curr_r)
            {
                idx_max_radius = i;
                curr_r = bstats.bubble2Radius.at(i);
            }
        }
    }
    double t_max_radius = (double)((double)bstats.ftime.at(idx_max_radius) / (double)
        FRAMERATE_CAMERA)
                + (double)(INITIAL_TIME_CAMERA) + (double)(DELAY_CAMERA_SECS);
    return t_max_radius;
}

double computeEstimatedTimeCollapse1(BubbleStats bstats, int bubbleID)
{
    // Calculate collapse time
    int idx_collapse = 0;
    int idx_max_radius = 0;
    double curr_r = 0;
    double delta_r = 0;

    for (int i = 0; i < bstats.fsize; i++)
    {
        // If circularity is high, we are most likely past the first bubble collapse
        // Adding this line to weed out weird later radius values
        // if (stats_int.bubbleCircularity.at(i) > 2) { break; }

        // Find the maximum radius
        if (bubbleID == 1)
        {
            if (bstats.bubble1Radius.at(i) > curr_r)
            {
                idx_max_radius = i;
                curr_r = bstats.bubble1Radius.at(i);
            }
        }
        else if (bubbleID == 2)
        {
            if (bstats.bubble2Radius.at(i) > curr_r)
            {
                idx_max_radius = i;
                curr_r = bstats.bubble2Radius.at(i);
            }
        }
    }

    for (int i = idx_max_radius; i < bstats.fsize; i++)
    {
        // Start from the maximum radius and see where the collapse is
        if (i > idx_max_radius)
        {
            if (bubbleID == 1)
                delta_r = bstats.bubble1Radius.at(i) - bstats.bubble1Radius.at(i-1);
            else if (bubbleID == 2)
                delta_r = bstats.bubble2Radius.at(i) - bstats.bubble2Radius.at(i-1);
        }
        if (delta_r > 0) break;
        else idx_collapse = i;
    }
    double t_collapse = (double)((double)bstats.ftime.at(idx_collapse) / (double)
        FRAMERATE_CAMERA)
        + (double)(INITIAL_TIME_CAMERA) + (double)(DELAY_CAMERA_SECS);
    return t_collapse;
}

double computeEstimatedTimeCollapse2(BubbleStats bstats, int bubbleID)
{
    // Calculate start time
    int idx_start = 0;
    for (int i = 0; i < bstats.fsize; i++)
    {
        // Find the maximum radius
        if (bubbleID == 1)
        {
            if (bstats.bubble1Radius.at(i) > -1)
            {
                idx_start = i;
                break;
            }
        }
        else if (bubbleID == 2)
        {
            if (bstats.bubble2Radius.at(i) > -1)
            {
                idx_start = i;
                break;
            }
        }
    }

    // Calculate collapse time
    int idx_collapse = 0;
    int idx_peak2 = 0;
    int idx_collapse2 = 0;
    int idx_max_radius = 0;
    double curr_r = 0;
    double delta_r = 0;
    double delta_r2 = 0;
    double delta_r3 = 0;

    for (int i = 0; i < bstats.fsize; i++)
    {
        // If circularity is high, we are most likely past the first bubble collapse
        // Adding this line to weed out weird later radius values
        // if (stats_int.bubbleCircularity.at(i) > 2) { break; }

        // Find the maximum radius
        if (bubbleID == 1)
        {
            if (bstats.bubble1Radius.at(i) > curr_r)
            {
                idx_max_radius = i;
                curr_r = bstats.bubble1Radius.at(i);
            }
        }
        else if (bubbleID == 2)
        {
            if (bstats.bubble2Radius.at(i) > curr_r)
            {
                idx_max_radius = i;
                curr_r = bstats.bubble2Radius.at(i);
            }
        }
    }

    double uncertainty = 0;
    for (int i = idx_max_radius; i < bstats.fsize; i++)
    {
        // Start from the maximum radius and see where the 1st collapse time is
        if (i > idx_max_radius)
        {
            if (bubbleID == 1)
                delta_r = bstats.bubble1Radius.at(i) - bstats.bubble1Radius.at(i-1);
            else if (bubbleID == 2)
                delta_r = bstats.bubble2Radius.at(i) - bstats.bubble2Radius.at(i-1);
        }
        if (delta_r > 0) break;
        else idx_collapse = i;
    }

    for (int i = idx_collapse; i < bstats.fsize; i++)
    {
        // Start from the 1st collapse time and see where the 2nd peak is
        if (i > idx_collapse)
        {
            if (bubbleID == 1)
                delta_r2 = bstats.bubble1Radius.at(i) - bstats.bubble1Radius.at(i-1);
            else if (bubbleID == 2)
                delta_r2 = bstats.bubble2Radius.at(i) - bstats.bubble2Radius.at(i-1);
        }

        if (delta_r2 < 0) break;
        else idx_peak2 = i;
    }
    for (int i = idx_peak2; i < bstats.fsize; i++)
    {
        // Start from the 2nd peak time and see where the 2nd collapse time is
        if (i > idx_peak2)
        {
            if (bubbleID == 1)
                delta_r3 = bstats.bubble1Radius.at(i) - bstats.bubble1Radius.at(i-1);
            else if (bubbleID == 2)
                delta_r3 = bstats.bubble2Radius.at(i) - bstats.bubble2Radius.at(i-1);
        }
        if (delta_r3 > 0) break;
        else
        {
            // Set a limit to the estimated frame at half of the length of BP1
            // If this limit is exceeded without finding collapse, go back by half a
                camera frame period
            // (This "uncertainty" is about 0.00005 s)
            if (i < (idx_peak2 + (idx_collapse - idx_start)/2))
                idx_collapse2 = i;
            else {
                uncertainty = DELTA_T_CAMERA * -0.5;
                break;
            }
        }
    }

    double t_collapse2 = (double)((double)bstats.ftime.at(idx_collapse2) / (double)
        FRAMERATE_CAMERA)
        + (double)(INITIAL_TIME_CAMERA) + (double)(DELAY_CAMERA_SECS) + uncertainty;
    return t_collapse2;
}
```

B.5 bsave Module

Listing B.8: AutoBubble/bsave.h

```cpp
#ifndef AUTOBUBBLE_BSAVE_H_
#define AUTOBUBBLE_BSAVE_H_

#include "AutoBubble/bcomps.h"

#include <opencv2/opencv.hpp>
#include <opencv/highgui.h>

#include <iostream>
#include <fstream>

#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <3rdParty/SimpleXlsx/Xlsx/Chart.h>
#include <3rdParty/SimpleXlsx/Xlsx/Chartsheet.h>
#include <3rdParty/SimpleXlsx/Xlsx/Workbook.h>
#include <3rdParty/SimpleXlsx/XLSXColors/XLSXColorLib.h>

#include "Common/defines.h"

using namespace cv;
using namespace SimpleXlsx;

#endif /* AUTOBUBBLE_BSAVE_H_ */
```

Listing B.9: AutoBubble/bsave.cpp

```cpp
#include "AutoBubble/bsave.h"

int saveSheet(BubbleStats bstats, int answer)
{
    // Implementing SimpleXlsx
    CWorkbook book( "CavTools" );

    Style style_frm_title;
    style_frm_title.horizAlign = ALIGN_H_CENTER;
    style_frm_title.font.attributes = FONT_BOLD;
    size_t StyleFrmTitle = book.AddStyle( style_frm_title );

    CellDataDbl cellDbl;
    cellDbl.style_id = 0;

    // Construct Statistics Sheet
    cout << "[INFO] Collecting bubble statistics from camera data... " << endl;

    CWorksheet & statsheet = book.AddSheet( "Statistics" );
    int stat_row = 0;
    statsheet.BeginRow(); stat_row++;
        statsheet.AddCell("Bubble 1 (from Camera Data)", StyleFrmTitle);
        statsheet.MergeCells( CellCoord( stat_row, 0 ), CellCoord( stat_row, 3 ) );
        statsheet.EndRow();
    statsheet.BeginRow(); stat_row++;
        statsheet.AddCell("Gamma");                            // Gamma = max radius / dist from wall
        statsheet.AddEmptyCells(2);
        statsheet.MergeCells( CellCoord( stat_row, 0 ), CellCoord( stat_row, 1 ) );
        stringstream formula;
        formula << "=1/(MAX('Camera'!H2:'Camera'!H27)"; // reciprocal of maximum radius divided by...
        formula << "/INDIRECT(ADDRESS(CELL(\"row\", ";          // access the equivalent row of max radius
        formula << "INDEX('Camera'!H2:H27, ";               // find the cell where that maximum is
        formula << "MATCH(MAX('Camera'!H2:H27),";           // get the maximum in Radius (mm)
        formula << "'Camera'!H2:H27,0))),";             // from this range in Camera Data sheet
        formula << "10,1,1,\"Camera\")))";                  // select it from column Dist from Wall (col 10)
        statsheet.AddCell(formula.str());
        statsheet.EndRow();
    statsheet.BeginRow(); stat_row++;
        statsheet.AddCell("Max Radius (mm)");
        statsheet.AddEmptyCells(2);
        statsheet.MergeCells( CellCoord( stat_row, 0 ), CellCoord( stat_row, 1 ) );
        statsheet.AddCell("=MAX('Camera'!H2:'Camera'!H27)");    // get the maximum in the column Radius (mm)
        statsheet.EndRow();
    statsheet.BeginRow(); stat_row++;
        statsheet.AddCell("Collapse Time 1 (s)");                      // start of the bubble to collapse of the bubble
        statsheet.AddEmptyCells(2);
        statsheet.MergeCells( CellCoord( stat_row, 0 ), CellCoord( stat_row, 1 ) );
        double ct = computeCollapseTime(bstats, 1);
        statsheet.AddCell(ct);
        statsheet.EndRow();
    statsheet.BeginRow(); stat_row++;
        statsheet.AddCell("Conversion Factor");
        statsheet.AddEmptyCells(2);
        statsheet.MergeCells( CellCoord( stat_row, 0 ), CellCoord( stat_row, 1 ) );
        statsheet.AddCell((double)MM_PX_SCALE);
        statsheet.EndRow();
    statsheet.BeginRow(); stat_row++;
        statsheet.EndRow();
    statsheet.BeginRow(); stat_row++;
        statsheet.AddCell("Est. Time of Start (s)");        // when a bubble starts to develop
        statsheet.AddEmptyCells(2);
        statsheet.MergeCells( CellCoord( stat_row, 0 ), CellCoord( stat_row, 1 ) );
        double et0 = computeEstimatedTimeStart(bstats, 1);
        statsheet.AddCell(et0);
        statsheet.EndRow();
    statsheet.BeginRow(); stat_row++;
        statsheet.AddCell("Est. Time of Max. Radius (s)");  // when a bubble obtains max
```

```
 radius
        statsheet.AddEmptyCells(2);
        statsheet.MergeCells( CellCoord( stat_row, 0 ), CellCoord( stat_row, 1 ) );
        double et05 = computeEstimatedTimeMax(bstats, 1);
        statsheet.AddCell(et05);
        statsheet.EndRow();
    statsheet.BeginRow(); stat_row++;
        statsheet.AddCell("Est. Time of Collapse 1 (s)");   // when the first collapse of bubble is
        statsheet.AddEmptyCells(2);
        statsheet.MergeCells( CellCoord( stat_row, 0 ), CellCoord( stat_row, 1 ) );
        double et1 = computeEstimatedTimeCollapse1(bstats, 1);
        statsheet.AddCell(et1);
        statsheet.EndRow();
    statsheet.BeginRow(); stat_row++;
        statsheet.AddCell("Est. Time of Collapse 2 (s)");   // when the second collapse of bubble is
        statsheet.AddEmptyCells(2);
        statsheet.MergeCells( CellCoord( stat_row, 0 ), CellCoord( stat_row, 1 ) );
        double et2 = computeEstimatedTimeCollapse2(bstats, 1);
        statsheet.AddCell(et2);
        statsheet.EndRow();
    statsheet.BeginRow(); stat_row++;
            statsheet.EndRow();
    if(answer == 2) {
        // Camera Data
        statsheet.BeginRow(); stat_row++;
            statsheet.AddCell("Bubble 2 (from Camera Data)", StyleFrmTitle);
            statsheet.MergeCells( CellCoord( stat_row, 0 ), CellCoord( stat_row, 3 ) );
            statsheet.EndRow();
        statsheet.BeginRow(); stat_row++;
            statsheet.AddCell("Gamma");                             // Gamma = max radius / dist from wall
            statsheet.AddEmptyCells(2);
            statsheet.MergeCells( CellCoord( stat_row, 0 ), CellCoord( stat_row, 1 ) );
            stringstream formula;
            formula << "=1/(MAX('Camera'!Q2:'Camera'!Q27)"; // reciprocal of maximum radius divided by...
            formula << "/INDIRECT(ADDRESS(CELL(\"row\", ";          // access the equivalent row of max radius
            formula << "INDEX('Camera'!Q2:Q27, ";               // find the cell where that maximum is
            formula << "MATCH(MAX('Camera'!Q2:Q27),";           // get the maximum in Radius (mm)
            formula << "'Camera'!Q2:Q27,0))),";             // from this range in Camera Data sheet
            formula << "19,1,1,\"Camera\")))";                 // select it from column Dist from Wall (col 10)
            statsheet.AddCell(formula.str());
            statsheet.EndRow();
        statsheet.BeginRow(); stat_row++;
            statsheet.AddCell("Max Radius (mm)");
            statsheet.AddEmptyCells(2);
            statsheet.MergeCells( CellCoord( stat_row, 0 ), CellCoord( stat_row, 1 ) );
            statsheet.AddCell("=MAX('Camera'!Q2:'Camera'!Q27)");    // get the maximum
```

```
                    in the column Radius (mm)
            statsheet.EndRow();
        statsheet.BeginRow(); stat_row++;
            statsheet.AddCell("Collapse Time (s)");                     // start of the
                bubble to collapse of the bubble
            statsheet.AddEmptyCells(2);
            statsheet.MergeCells( CellCoord( stat_row, 0 ), CellCoord( stat_row, 1 ) );
            double ct = computeCollapseTime(bstats, 2);
            statsheet.AddCell(ct);
            statsheet.EndRow();
        statsheet.BeginRow(); stat_row++;
            statsheet.AddCell("Conversion Factor");
            statsheet.AddEmptyCells(2);
            statsheet.MergeCells( CellCoord( stat_row, 0 ), CellCoord( stat_row, 1 ) );
            statsheet.AddCell((double)MM_PX_SCALE);
            statsheet.EndRow();
        statsheet.BeginRow(); stat_row++;
            statsheet.EndRow();
        statsheet.BeginRow(); stat_row++;
            statsheet.AddCell("Est. Time of Start (s)");        // when a bubble starts
                to develop
            statsheet.AddEmptyCells(2);
            statsheet.MergeCells( CellCoord( stat_row, 0 ), CellCoord( stat_row, 1 ) );
            double et0_2 = computeEstimatedTimeStart(bstats, 2);
            statsheet.AddCell(et0_2);
            statsheet.EndRow();
        statsheet.BeginRow(); stat_row++;
            statsheet.AddCell("Est. Time of Max. Radius (s)");  // when a bubble obtains
                 max radius
            statsheet.AddEmptyCells(2);
            statsheet.MergeCells( CellCoord( stat_row, 0 ), CellCoord( stat_row, 1 ) );
            double et05_2 = computeEstimatedTimeMax(bstats, 2);
            statsheet.AddCell(et05_2);
            statsheet.EndRow();
        statsheet.BeginRow(); stat_row++;
            statsheet.AddCell("Est. Time of Collapse 1 (s)");   // when the first
                collapse of bubble is
            statsheet.AddEmptyCells(2);
            statsheet.MergeCells( CellCoord( stat_row, 0 ), CellCoord( stat_row, 1 ) );
            double et1_2 = computeEstimatedTimeCollapse1(bstats, 2);
            statsheet.AddCell(et1_2);
            statsheet.EndRow();
        statsheet.BeginRow(); stat_row++;
            statsheet.AddCell("Est. Time of Collapse 2 (s)");   // when the second
                collapse of bubble is
            statsheet.AddEmptyCells(2);
            statsheet.MergeCells( CellCoord( stat_row, 0 ), CellCoord( stat_row, 1 ) );
            double et2_2 = computeEstimatedTimeCollapse2(bstats, 2);
            statsheet.AddCell(et2_2);
            statsheet.EndRow();
        statsheet.BeginRow(); stat_row++; statsheet.EndRow();
    }

    // Construct Calculations Sheet for PVDF Data
    CWorksheet & calcsheet = book.AddSheet( "Calculations" );
    int calc_row = 0;
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("Preset Constants", StyleFrmTitle);
        calcsheet.MergeCells( CellCoord( calc_row, 0 ), CellCoord( calc_row, 3 ) );
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("N_PVDF");                    // Number of datapoints for PVDF
        calcsheet.AddCell(NUM_OF_PVDF_POINTS);
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("N_CAM");                     // Number of datapoints for
            camera
        calcsheet.AddCell(NUM_OF_CAMERA_POINTS);
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("f_PVDF");                    // Period of PVDF
        calcsheet.AddCell(DELTA_T_PVDF);
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("f_CAM");                     // Period of camera
        calcsheet.AddCell(DELTA_T_CAMERA);
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++; calcsheet.EndRow();

    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("Estimates from Camera Data", StyleFrmTitle);
        calcsheet.MergeCells( CellCoord( calc_row, 0 ), CellCoord( calc_row, 3 ) );
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("ET0");                       // Estimated Time of Start
        calcsheet.AddCell(computeEstimatedTimeStart(bstats, 1));
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("ET0.5");                     // Estimated Time of Max Radius
        calcsheet.AddCell(computeEstimatedTimeMax(bstats, 1));
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("ET1");                       // Estimated Time of Collapse 1
        calcsheet.AddCell(computeEstimatedTimeCollapse1(bstats, 1));
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("ET2");                       // Estimated Time of Collapse 2
        calcsheet.AddCell(computeEstimatedTimeCollapse2(bstats, 1));
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++; calcsheet.EndRow();

    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("Actual Times of Collapse", StyleFrmTitle);
        calcsheet.MergeCells( CellCoord( calc_row, 0 ), CellCoord( calc_row, 3 ) );
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("T0");                        // Actual Time of Start
        calcsheet.AddCell("=D45");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
```

```
        calcsheet.AddCell("T0.5");                        // Actual Time of Max Radius
        calcsheet.AddCell("=D46");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("T1");                          // Actual Time of Collapse 1
        calcsheet.AddCell("=D47");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("T2");                          // Actual Time of Collapse 2
        calcsheet.AddCell("=D48");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("errT0");                       // Error between ET0 and T0
        calcsheet.AddCell("=ABS(B14-B8)");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("errT0.5");                     // Error between ET0.5 and T0.5
        calcsheet.AddCell("=ABS(B15-B9)");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("errT1");                       // Error between ET1 and T1
        calcsheet.AddCell("=ABS(B16-B10)");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("errT2");                       // Error between ET2 and T2
        calcsheet.AddCell("=ABS(B17-B11)");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++; calcsheet.EndRow();

    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("Bubble Periods", StyleFrmTitle);
        calcsheet.MergeCells( CellCoord( calc_row, 0 ), CellCoord( calc_row, 3 ) );
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("ERCT");                        // Estimated Rayleigh Collapse
            Time
        calcsheet.AddCell("=B10-B9");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("DRCT");                        // Detected Rayleigh Collapse
            Time
        calcsheet.AddCell("=B16-B15");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("BP1");                         // Bubble Period 1
        calcsheet.AddCell("=B16-B14");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("BP2");                         // Bubble Period 2
        calcsheet.AddCell("=B17-B16");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++; calcsheet.EndRow();

    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("Collapse Voltages", StyleFrmTitle);
        calcsheet.MergeCells( CellCoord( calc_row, 0 ), CellCoord( calc_row, 3 ) );
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("CV1");                   // Collapse Voltage 1
        calcsheet.AddCell("=C47");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("CV2");                   // Collapse Voltage 2
        calcsheet.AddCell("=C48");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++; calcsheet.EndRow();

    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("Collapse Widths", StyleFrmTitle);
        calcsheet.MergeCells( CellCoord( calc_row, 0 ), CellCoord( calc_row, 3 ) );
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("CW1");                   // Collapse Width 1
        calcsheet.AddCell("=(I52-H52)*$B$4");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("CW2");                   // Collapse Width 2
        calcsheet.AddCell("=(I53-H53)*$B$4");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++; calcsheet.EndRow();

    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("Collapse Areas and Impulses", StyleFrmTitle);
        calcsheet.MergeCells( CellCoord( calc_row, 0 ), CellCoord( calc_row, 3 ) );
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("CA1");                   // Collapse Area 1
        calcsheet.AddCell("=(SUM(INDIRECT(\"PVDF!\"&ADDRESS(L52,5,1,1)&\":\"&ADDRESS(M52
            ,5,1,1)))*$B$4)");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("CA2");                   // Collapse Area 2
        calcsheet.AddCell("=(SUM(INDIRECT(\"PVDF!\"&ADDRESS(L53,5,1,1)&\":\"&ADDRESS(M53
            ,5,1,1)))*$B$4)");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("CJ1");                   // Collapse Impulse 1
        calcsheet.AddCell("=(SUM(INDIRECT(\"PVDF!\"&ADDRESS(L52,6,1,1)&\":\"&ADDRESS(M52
            ,6,1,1)))*$B$4)");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("CJ2");                   // Collapse Impulse 2
        calcsheet.AddCell("=(SUM(INDIRECT(\"PVDF!\"&ADDRESS(L53,6,1,1)&\":\"&ADDRESS(M53
            ,6,1,1)))*$B$4)");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++; calcsheet.EndRow();

    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("Searching for Actual Times of Collapse", StyleFrmTitle);
        calcsheet.MergeCells( CellCoord( calc_row, 0 ), CellCoord( calc_row, 3 ) );
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("Loc Start");              // Start Location of Search
        calcsheet.AddCell("Loc End");                // End Location of Search
        calcsheet.AddCell("Max Btwn");               // Maximum between Start and End
        calcsheet.AddCell("Loc Max Btwn");           // Location of this Maximum
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell(1);
        calcsheet.AddCell("=MATCH($B$8,INDIRECT(\"PVDF!\"&ADDRESS(2,1,1,1)&\":\"&ADDRESS($B$2+1,1,1,1)),0)");
        calcsheet.AddCell("=MAX(INDIRECT(\"PVDF!\"&ADDRESS(A45,2,1,1)&\":\"&ADDRESS(B45,2,1,1)))");
        calcsheet.AddCell("=INDIRECT(\"PVDF!\"&ADDRESS(MATCH(C45,INDIRECT(\"PVDF!\"&ADDRESS(2,2,1,1)&\":\"&ADDRESS($B$2+1,2,1,1)),0),1,1,1))");
        calcsheet.AddEmptyCells(1);
        calcsheet.AddCell("Find first peak (T0)");
        calcsheet.AddEmptyCells(2);
        calcsheet.AddCell("Search from 1 to ET0");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("=MATCH($B$8,INDIRECT(\"PVDF!\"&ADDRESS(2,1,1,1)&\":\"&ADDRESS($B$2+1,1,1,1)),0)");
        calcsheet.AddCell("=MATCH($B$9+$B$5,INDIRECT(\"PVDF!\"&ADDRESS(2,1,1,1)&\":\"&ADDRESS($B$2+1,1,1,1)))");
        calcsheet.AddCell("=MAX(INDIRECT(\"PVDF!\"&ADDRESS(A46,2,1,1)&\":\"&ADDRESS(B46,2,1,1)))");
        calcsheet.AddCell("=INDIRECT(\"PVDF!\"&ADDRESS(MATCH(C46,INDIRECT(\"PVDF!\"&ADDRESS(A46,2,1,1)&\":\"&ADDRESS($B$2+1,2,1,1)),0)+A46,1,1,1))");
        calcsheet.AddEmptyCells(1);
        calcsheet.AddCell("Find second peak (T0.5)");
        calcsheet.AddEmptyCells(2);
        calcsheet.AddCell("Search from ET0 to ET0.5+dET");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("=MATCH($B$10,INDIRECT(\"PVDF!\"&ADDRESS(2,1,1,1)&\":\"&ADDRESS($B$2+1,1,1,1)),0)");
        calcsheet.AddCell("=MATCH($B$11,INDIRECT(\"PVDF!\"&ADDRESS(2,1,1,1)&\":\"&ADDRESS($B$2+1,1,1,1)))");
        calcsheet.AddCell("=MAX(INDIRECT(\"PVDF!\"&ADDRESS(A47,2,1,1)&\":\"&ADDRESS(B47,2,1,1)))");
        calcsheet.AddCell("=INDIRECT(\"PVDF!\"&ADDRESS(MATCH(C47,INDIRECT(\"PVDF!\"&ADDRESS(A47,2,1,1)&\":\"&ADDRESS($B$2+1,2,1,1)),0)+A47,1,1,1))");
        calcsheet.AddEmptyCells(1);
        calcsheet.AddCell("Find third peak (T1)");
        calcsheet.AddEmptyCells(2);
        calcsheet.AddCell("Search from ET1 to ET2");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("=MATCH($B$11,INDIRECT(\"PVDF!\"&ADDRESS(2,1,1,1)&\":\"&ADDRESS($B$2+1,1,1,1)))");
        calcsheet.AddCell("=MATCH($B$11+(3*$B$5),INDIRECT(\"PVDF!\"&ADDRESS(2,1,1,1)&\":\"&ADDRESS($B$2+1,1,1,1)))");
        calcsheet.AddCell("=MAX(INDIRECT(\"PVDF!\"&ADDRESS(A48,2,1,1)&\":\"&ADDRESS(B48,2,1,1)))");
        calcsheet.AddCell("=INDIRECT(\"PVDF!\"&ADDRESS(MATCH(C48,INDIRECT(\"PVDF!\"&
            ADDRESS(A48,2,1,1)&\":\"&ADDRESS($B$2+1,2,1,1)),0)+A48,1,1,1))");
        calcsheet.AddEmptyCells(1);
        calcsheet.AddCell("Find fourth peak (T0)");
        calcsheet.AddEmptyCells(2);
        calcsheet.AddCell("Search from ET2 to ET2+3dET2");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++; calcsheet.EndRow();

    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("Searching for Collapse Widths and Areas", StyleFrmTitle);
        calcsheet.MergeCells( CellCoord( calc_row, 0 ), CellCoord( calc_row, 3 ) );
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("T_pk");              // Time of collapse
        calcsheet.AddCell("1/2 CV");            // Half of collapse voltage
        calcsheet.AddCell("T_pk -0.5");         // Halfway between previous and this
            peak
        calcsheet.AddCell("T_pk +0.5");         // Halfway between this and next peak
        calcsheet.AddCell("Loc T_pk -0.5");     // Location of T_pk -0.5
        calcsheet.AddCell("Loc T_pk +0.5");     // Location of T_pk +0.5
        calcsheet.AddCell("Loc T_pk");          // Location of T_pk
        calcsheet.AddCell("Loc 1/2 CV -");      // Location of 1/2 CV -
        calcsheet.AddCell("Loc 1/2 CV +");      // Location of 1/2 CV +
        calcsheet.AddCell("T_pk -0");           // Bottom of the peak, left
        calcsheet.AddCell("T_pk +0");           // Bottom of the peak, right
        calcsheet.AddCell("Loc T_pk -0");       // Where T_pk -0 is found
        calcsheet.AddCell("Loc T_pk +0");       // Where T_pk +0 is found
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
        calcsheet.AddCell("=B16");
        calcsheet.AddCell("=B30/2");
        calcsheet.AddCell("=AVERAGE(B15:B16)");
        calcsheet.AddCell("=AVERAGE(B16:B17)");
        calcsheet.AddCell("=MATCH(C52,INDIRECT(\"PVDF!\"&ADDRESS(2,1,1,1)&\":\"&ADDRESS(
            $B$2+1,1,1,1)))");
        calcsheet.AddCell("=MATCH(D52,INDIRECT(\"PVDF!\"&ADDRESS(E52,1,1,1)&\":\"&
            ADDRESS($B$2+1,1,1,1)))+E52");
        calcsheet.AddCell("=MATCH(B30,INDIRECT(\"PVDF!\"&ADDRESS(A47,2,1,1)&\":\"&
            ADDRESS($B$2+1,2,1,1)),0)+A47");
        calcsheet.AddCell("=MATCH(B52,INDIRECT(\"PVDF!\"&ADDRESS(E52,2,1,1)&\":\"&
            ADDRESS(G52,2,1,1)))+E52");
        calcsheet.AddCell("=MATCH(B52,INDIRECT(\"PVDF!\"&ADDRESS(G52,2,1,1)&\":\"&
            ADDRESS(F52,2,1,1)),-1)+G52");
        calcsheet.AddCell("=MIN(INDIRECT(\"PVDF!\"&ADDRESS(E52,2,1,1)&\":\"&ADDRESS(G52
            ,2,1,1)))");
        calcsheet.AddCell("=MIN(INDIRECT(\"PVDF!\"&ADDRESS(G52,2,1,1)&\":\"&ADDRESS(F52
            ,2,1,1)))");
        calcsheet.AddCell("=MATCH(J52,INDIRECT(\"PVDF!\"&ADDRESS(E52,2,1,1)&\":\"&
            ADDRESS(G52,2,1,1)),-1)+E52");
        calcsheet.AddCell("=MATCH(K52,INDIRECT(\"PVDF!\"&ADDRESS(G52,2,1,1)&\":\"&
            ADDRESS(F52,2,1,1)),-1)+G52");
        calcsheet.EndRow();
    calcsheet.BeginRow(); calc_row++;
    calcsheet.AddCell("=B17");
```

```
        calcsheet.AddCell("=B31/2");
        calcsheet.AddCell("=AVERAGE(B16:B17)");
        calcsheet.AddCell("=AVERAGE(B17,B17+(2*B5))");
        calcsheet.AddCell("=MATCH(C53,INDIRECT(\"PVDF!\"&ADDRESS(2,1,1,1)&\":\"&ADDRESS(
            $B$2+1,1,1,1)))");
        calcsheet.AddCell("=MATCH(D53,INDIRECT(\"PVDF!\"&ADDRESS(E53,1,1,1)&\":\"&
            ADDRESS($B$2+1,1,1,1)))+E53");
        calcsheet.AddCell("=MATCH(B31,INDIRECT(\"PVDF!\"&ADDRESS(A48,2,1,1)&\":\"&
            ADDRESS($B$2+1,2,1,1)),0)+A48");
        calcsheet.AddCell("=MATCH(B53,INDIRECT(\"PVDF!\"&ADDRESS(E53,2,1,1)&\":\"&
            ADDRESS(G53,2,1,1)))+E53");
        calcsheet.AddCell("=MATCH(B53,INDIRECT(\"PVDF!\"&ADDRESS(G53,2,1,1)&\":\"&
            ADDRESS(F53,2,1,1)),-1)+G53");
        calcsheet.AddCell("=MIN(INDIRECT(\"PVDF!\"&ADDRESS(E53,2,1,1)&\":\"&ADDRESS(G53
            ,2,1,1)))");
        calcsheet.AddCell("=MIN(INDIRECT(\"PVDF!\"&ADDRESS(G53,2,1,1)&\":\"&ADDRESS(F53
            ,2,1,1)))");
        calcsheet.AddCell("=MATCH(J53,INDIRECT(\"PVDF!\"&ADDRESS(E53,2,1,1)&\":\"&
            ADDRESS(G53,2,1,1)),-1)+E53");
        calcsheet.AddCell("=MATCH(K53,INDIRECT(\"PVDF!\"&ADDRESS(G53,2,1,1)&\":\"&
            ADDRESS(F53,2,1,1)),-1)+G53");
    calcsheet.EndRow();

    // Construct Camera Data Sheet
    CWorksheet & sheet_c = book.AddSheet( "Camera" );
    cout << "[INFO] Writing XLSX data..." << endl;
    sheet_c.BeginRow();
    sheet_c.AddCell("Frame");
    sheet_c.AddCell("Time (s)");
    sheet_c.AddCell("B1 PosX");
    sheet_c.AddCell("B1 PosY");
    sheet_c.AddCell("B1 Area");
    sheet_c.AddCell("B1 Perimeter");
    sheet_c.AddCell("B1 Radius");
    sheet_c.AddCell("B1 Radius (mm)");
    sheet_c.AddCell("B1 Dist From Wall");
    sheet_c.AddCell("B1 Dist From Wall (mm)");
    sheet_c.AddCell("B1 Circularity");
    if (answer == 2)
    {
        sheet_c.AddCell("B2 PosX");
        sheet_c.AddCell("B2 PosY");
        sheet_c.AddCell("B2 Area");
        sheet_c.AddCell("B2 Perimeter");
        sheet_c.AddCell("B2 Radius");
        sheet_c.AddCell("B2 Radius (mm)");
        sheet_c.AddCell("B2 Dist From Wall");
        sheet_c.AddCell("B2 Dist From Wall (mm)");
        sheet_c.AddCell("B2 Circularity");
    }
    sheet_c.EndRow();
    for (int i = 0; i < NUM_OF_CAMERA_POINTS; i++)
    {
        sheet_c.BeginRow();
        cellDbl.value = bstats.ftime.at(i);
        sheet_c.AddCell( cellDbl );
        double ftime_s = (double)((double)bstats.ftime.at(i) / (double)FRAMERATE_CAMERA)
            + (double)(INITIAL_TIME_CAMERA) + (double)(DELAY_CAMERA_SECS);
        cellDbl.value = ftime_s;
        sheet_c.AddCell( cellDbl );
        cellDbl.value = bstats.bubble1PosX.at(i);
        sheet_c.AddCell( cellDbl );
        cellDbl.value = bstats.bubble1PosY.at(i);
        sheet_c.AddCell( cellDbl );
        cellDbl.value = bstats.bubble1Area.at(i);
        sheet_c.AddCell( cellDbl );
        cellDbl.value = bstats.bubble1Perimeter.at(i);
        sheet_c.AddCell( cellDbl );
        cellDbl.value = bstats.bubble1Radius.at(i);
        sheet_c.AddCell( cellDbl );
        cellDbl.value = bstats.bubble1Radius.at(i) * MM_PX_SCALE;
        sheet_c.AddCell( cellDbl );
        cellDbl.value = bstats.bubble1DistanceFromWall.at(i);
        sheet_c.AddCell( cellDbl );
        cellDbl.value = bstats.bubble1DistanceFromWall.at(i) * MM_PX_SCALE;
        sheet_c.AddCell( cellDbl );
        cellDbl.value = bstats.bubble1Circularity.at(i);
        sheet_c.AddCell( cellDbl );

        if (answer == 2) {
            cellDbl.value = bstats.bubble2PosX.at(i);
            sheet_c.AddCell( cellDbl );
            cellDbl.value = bstats.bubble2PosY.at(i);
            sheet_c.AddCell( cellDbl );
            cellDbl.value = bstats.bubble2Area.at(i);
            sheet_c.AddCell( cellDbl );
            cellDbl.value = bstats.bubble2Perimeter.at(i);
            sheet_c.AddCell( cellDbl );
            cellDbl.value = bstats.bubble2Radius.at(i);
            sheet_c.AddCell( cellDbl );
            cellDbl.value = bstats.bubble2Radius.at(i) * MM_PX_SCALE;
            sheet_c.AddCell( cellDbl );
            cellDbl.value = bstats.bubble2DistanceFromWall.at(i);
            sheet_c.AddCell( cellDbl );
            cellDbl.value = bstats.bubble2DistanceFromWall.at(i) * MM_PX_SCALE;
            sheet_c.AddCell( cellDbl );
            cellDbl.value = bstats.bubble2Circularity.at(i);
            sheet_c.AddCell( cellDbl );
        }
        sheet_c.EndRow();
    }

    // Construct PVDF Data Sheet from CSV
    CWorksheet & sheet_p = book.AddSheet( "PVDF" );
    sheet_p.BeginRow();
    sheet_p.AddCell("Time (s)");
    sheet_p.AddCell("Voltage (V)");
    sheet_p.AddCell("Voltage (V) x10");
    sheet_p.AddCell("Force (N)");
    sheet_p.AddCell("Voltage x Time (V.s)");
```

```cpp
    sheet_p.AddCell("Force x Time (N.s)");
    sheet_p.EndRow();
    fstream pvdf_data;
    try {
        pvdf_data.open("pvdf_data.csv", ios::in);
        cout << "[INFO] PVDF CSV datafile found." << endl;
    }
    catch (int e) { cout << "[ERROR] PVDF data not found in working directory!" << endl;
         }
    vector<string> row;
    string line, word;
    if (pvdf_data.is_open())
    {
        // Remove the header
        for (int head_cnt = 0; head_cnt < 8; head_cnt++)
            getline(pvdf_data, line);

        // Search for the start of the data
        double x_val;
        do
        {
            row.clear();
            getline(pvdf_data, line);
            stringstream s(line);
            while (getline(s, word, ',')) {
                row.push_back(word);
            }
            x_val = stod(row[0]);
        } while (x_val < (double)(INITIAL_TIME_CAMERA - (DELAY_PVDF_SECS -
            DELAY_CAMERA_SECS)));

        cout << "[INFO] Opening included PVDF data..." << endl;

        // Parse the next set of incoming data (120000 datapoints)
        double temp_t, temp_v, temp_v10, temp_n, temp_vs, temp_ns;
        for (int data_cnt = 0; data_cnt < NUM_OF_PVDF_POINTS; data_cnt++)
        {
            sheet_p.BeginRow();
            row.clear();
            getline(pvdf_data, line);
            stringstream s(line);
            while (getline(s, word, ',')) {
                row.push_back(word);
            }
            temp_t = stod(row[0]) + DELAY_PVDF_SECS;
            sheet_p.AddCell( temp_t );
            temp_v = stod(row[1]);
            sheet_p.AddCell( temp_v );
            temp_v10 = temp_v * 10;
            sheet_p.AddCell( temp_v10 );
            temp_n = temp_v / CALIB_CONSTANT;
            sheet_p.AddCell( temp_n );
            temp_vs = temp_v * DELTA_T_PVDF;
            sheet_p.AddCell( temp_vs );
            temp_ns = temp_n * DELTA_T_PVDF;
            sheet_p.AddCell( temp_ns );
            sheet_p.EndRow();
        }
    }

    // Build the graph
    cout << "[INFO] Building bubble evolution graph..." << endl;
    XLSXColorLib cl_lib, gs10_lib;
    make_excell_like_named_colors( cl_lib );
    make_grayscale10( gs10_lib );
    CChart::Series ser;
    CChartsheet & bubble_evol = book.AddChartSheet( "Bubble Evolution", CHART_SCATTER );

    // Add PVDF Data
    ser.title = "Voltage (V)";
    ser.catSheet =  &sheet_p;
    ser.catAxisFrom = CellCoord( 2, 0 );
    ser.catAxisTo = CellCoord( (NUM_OF_PVDF_POINTS + 1), 0 );
    ser.valSheet =  &sheet_p;
    ser.valAxisFrom = CellCoord( 2, 1 );
    ser.valAxisTo = CellCoord( (NUM_OF_PVDF_POINTS + 1), 1 );
    ser.JoinType = CChart::Series::joinSmooth;
    ser.Marker.LineColor = cl_lib.GetColor( "Black" );
    ser.Marker.Type = CChart::Series::symNone;
    bubble_evol.Chart().AddSeries( ser );

    // Add Camera Data
    ser.title = "Bubble 1 Radius (mm)";
    ser.catSheet =  &sheet_c;
    ser.catAxisFrom = CellCoord( 2, 1 );
    ser.catAxisTo = CellCoord( (NUM_OF_CAMERA_POINTS + 1), 1 );
    ser.valSheet =  &sheet_c;
    ser.valAxisFrom = CellCoord( 2, 7 );
    ser.valAxisTo = CellCoord( (NUM_OF_CAMERA_POINTS + 1), 7 );
    ser.JoinType = CChart::Series::joinNone;
    ser.Marker.LineColor = cl_lib.GetColor( "Blue" );
    ser.Marker.Type = CChart::Series::symCircle;
    bubble_evol.Chart().AddSeries( ser );
    if (answer == 2)
    {
        ser.title = "Bubble 2 Radius (mm)";
        ser.catSheet =  &sheet_c;
        ser.catAxisFrom = CellCoord( 2, 1 );
        ser.catAxisTo = CellCoord( (NUM_OF_CAMERA_POINTS + 1), 1 );
        ser.valSheet =  &sheet_c;
        ser.valAxisFrom = CellCoord( 2, 16 );
        ser.valAxisTo = CellCoord( (NUM_OF_CAMERA_POINTS + 1), 16 );
        ser.JoinType = CChart::Series::joinNone;
        ser.Marker.LineColor = cl_lib.GetColor( "Red" );
        ser.Marker.Type = CChart::Series::symDiamond;
        bubble_evol.Chart().AddSeries( ser );
    }

    bubble_evol.Chart().SetXAxisName( "Time (sec)" );
    bubble_evol.Chart().SetXAxisMin("0.0055");
    bubble_evol.Chart().SetXAxisMax("0.0085");
    bubble_evol.Chart().SetYAxisMin("-2");
    bubble_evol.Chart().SetYAxisMax("6");
    bubble_evol.Chart().SetDiagrammName( "Bubble Evolution" );

    // Add bubble images
    cout << "[INFO] Adding bubble images from camera..." << endl;
    CWorksheet & bubble_imgs = book.AddSheet( "Bubble Dynamics Images" );
    for (int row = 1; row < 6; row++) { bubble_imgs.BeginRow(); bubble_imgs.EndRow(); }
    for(int frm = 0; frm < NUM_OF_CAMERA_POINTS; frm++)
    {
        string filename = buildBubbleFileNames("ImgA", (frm + INITIAL_FRAME_CAMERA));
        book.AddImage( bubble_imgs, filename, DrawingPoint( (frm*2), 1 ), DrawingPoint(
            ((frm*2)+2), 5 ) );
    }

    bubble_imgs.BeginRow();     // Row 6
    for(int frm_idx = 0; frm_idx < NUM_OF_CAMERA_POINTS; frm_idx++)
    {
        stringstream ss;
        ss << frm_idx;
        string s = ss.str();
        bubble_imgs.AddCell( "Frame " + s, StyleFrmTitle );
        bubble_imgs.AddEmptyCells( 1 );
        bubble_imgs.MergeCells( CellCoord( 6, (frm_idx*2) ), CellCoord( 6, (frm_idx*2)+1
             ) );
    }
    bubble_imgs.EndRow();
    Style style_frm_time;
    style_frm_time.horizAlign = ALIGN_H_CENTER;
    size_t StyleFrmTime = book.AddStyle( style_frm_time );
    bubble_imgs.BeginRow();     // Row 7
    for(int frm_idx2 = 0; frm_idx2 < NUM_OF_CAMERA_POINTS; frm_idx2++)
    {
        stringstream ss;
        double ftime_sec = (double)((double)bstats.ftime.at(frm_idx2) / (double)
            FRAMERATE_CAMERA)
            + (double)(INITIAL_TIME_CAMERA) + (double)(DELAY_CAMERA_SECS);
        ss << (ftime_sec * 1000);
        string s = ss.str();
        bubble_imgs.AddCell( s + " ms", StyleFrmTime );
        bubble_imgs.AddEmptyCells( 1 );
        bubble_imgs.MergeCells( CellCoord( 7, (frm_idx2*2) ), CellCoord( 7, (frm_idx2*2)
            +1 ) );
    }
    bubble_imgs.EndRow();

    // Try saving the XLSX with the data
    if( book.Save( "data_output_xlsx_graph.xlsx" ) )
        cout << "[INFO] The output XLSX has been saved successfully" << endl;
    else cout << "[ERROR] Output XLSX saving has failed" << endl;

    return (int)(bstats.bubble1Radius.at(10) * MM_PX_SCALE);
}
```

B.6 bwin Module

Listing B.10: AutoBubble/bwin.h

```
#ifndef AUTOBUBBLE_BWIN_H_
#define AUTOBUBBLE_BWIN_H_

#include <iostream>
#include <windows.h>
#include <Commdlg.h>

int AskSingleBubble();

#endif /* AUTOBUBBLE_BWIN_H_ */
```

Listing B.11: AutoBubble/bwin.cpp

```
int AskSingleBubble()
{
    int result;
    int msgboxID = MessageBoxA(
        NULL,
        (const char*)"Is this setup for a double bubble configuration? \n(If No, this is
             a single bubble configuration)",
        (const char*)"Importing Camera data...",
        MB_ICONINFORMATION | MB_YESNO | MB_DEFBUTTON1
    );
    switch (msgboxID) {
        case IDYES: result = 2; break;
        case IDNO: result = 1;break;
    }
    return result;
}
```

B.7 Miscellaneous Includes

Listing B.12: Common/includes.h

```
#ifndef COMMON_INCLUDES_H_
#define COMMON_INCLUDES_H_

//#include "AutoCloud/autocloud.h"
#include "AutoBubble/bcomps.h"
#include "AutoBubble/bload.h"
#include "AutoBubble/bread.h"
#include "AutoBubble/bsave.h"
#include "AutoBubble/bwin.h"

#include <opencv2/opencv.hpp>
#include <opencv/highgui.h>

#include <windows.h>
#include <Commdlg.h>

#include <iostream>
#include <fstream>

#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <3rdParty/SimpleXlsx/Xlsx/Chart.h>
#include <3rdParty/SimpleXlsx/Xlsx/Chartsheet.h>
#include <3rdParty/SimpleXlsx/Xlsx/Workbook.h>
#include <3rdParty/SimpleXlsx/XLSXColors/XLSXColorLib.h>

// Namespaces
using namespace cv;
using namespace std;
using namespace SimpleXlsx;

#endif /* COMMON_INCLUDES_H_ */
```

Listing B.13: Common/defines.h

```
#ifndef COMMON_DEFINES_H_
#define COMMON_DEFINES_H_

// Constants
#define MIN_BUBBLE_SIZE                 2
#define MIN_WALL_LENGTH                 100
#define MAX_BUBBLE_SIZE                 300         // bubble can only be as reasonably
    large as this value
#define SAFE_BUBBLE_IDX                 13          // bubble index which can be used as
     the "safe bubble"
#define BUBBLE_CNTR_IDX_START           0           // The index of the contours will
    start at this number
#define PCT_BUBBLE_2_BOUNDS_TOLERANCE   0.4         // bubble can be as small as this
    percentage of bounding box
```

```
#define BUBBLE_BOUNDREC_APPROX_FACTOR   (M_PI/4)    // pi/4 = 0.875, ratio of areas of ellipse vs circumscribed rectangle
#define CIRCULARITY_THRESHOLD           4.5         // bubbles are not possible to be above this circularity measure
#define RELIABILITY_THRESHOLD           50          // contour perimeter ^2 / (contour area * 4 * pi)
#define DIFFERENCE_THRESHOLD            2           // threshold of bubble size delta difference
#define BUBBLE_SEARCH_RADIUS            25          // radius threshold that dictates reasonable bubble movement
#define RHO_H2O                         997         // fluid (water) density in kg m-^3
#define P_ATM                           101325      // atmospheric pressure in Pa (kg m^-1 s^-2)
#define CALIB_CONSTANT                  0.00771     // conversion factor in V/N for PVDF, from calibration

// Datatypes
#define contourField vector<vector<Point>>
#define hierField vector<Vec4i>

// Camera + PVDF Settings
#define FRAMERATE_PVDF                  60000000        // 60 MHz film data read rate
#define FRAMERATE_CAMERA                10000           // 10000 frames per second camera setting
#define DELTA_T_PVDF                    0.0000000167    // 1.67e-8 s
#define DELTA_T_CAMERA                  0.0001          // 1.00e-4 s
#define INITIAL_TIME_CAMERA             0.0025          // Starting point for bubble in camera (s)
#define INITIAL_FRAME_CAMERA            INITIAL_TIME_CAMERA * FRAMERATE_CAMERA  // Starting frame for bubble in camera (f)
#define NUM_OF_CAMERA_FRAMES            140             // set in the camera
#define NUM_OF_CAMERA_POINTS            31              // number of data points for camera
#define NUM_OF_PVDF_POINTS              (NUM_OF_CAMERA_POINTS-1) * (FRAMERATE_PVDF / FRAMERATE_CAMERA) // number of data points for pvdf

#define DELAY_CAMERA_SECS               0.003           // Delay in camera (s)
#define DELAY_PVDF_SECS                 0.005           // Delay in pvdf (s)

#define MM_PX_SCALE                     0.2174          // millimeters / pixel (5/23)

#endif /* COMMON_DEFINES_H_ */
```