

**MASTER**

**3D skin body reconstruction**

Zerva, C.

*Award date:*  
2015

[Link to publication](#)

**Disclaimer**

This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

**General rights**

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

# 3D Skin Body Reconstruction

*Master Thesis*

Christina Zerva

Supervisors:

Prof. Dr. Ir Gerard de Haan

Dr. Calina Ciuhu-Pijlman

Dr. Mounir Zeitouny

# Abstract

This report describes the work of a Master Graduation Project in the Embedded Systems Program of Technical University of Eindhoven, which is carried at Philips Research in the Personal Care and Wellness department.

The aim of this work is to show the feasibility of a 3D body scanner to reconstruct body skin surface from sequences of a 2D video. This will later serve for applications on consumer devices, mainly, body treatment devices. Users in the future will be able to have a body scan and get a 3D map of their body registered on a smart device or in the cloud. When personal care devices and personal maps are gained, localisation and navigation on the body is enabled they are offering an extraordinary experience to the users.

In the first part of this thesis we will address the prototype that we assembled together with the software algorithm that has been designed and implemented in order to have a functional body scanner.

The second part focuses on solutions for various implementations of body scanner targeting at reducing cost, processing spend and data footprint to make the proposed technology feasible for consumer devices.

# Acknowledgments

I wish to express my sincere thanks to all those people who supported me during this thesis work. First of all, I would like to thank Dr. Jim Coombs, for the opportunity of doing my internship at Philips Research in the department of Personal Care and Wellness. My supervisors, Dr. Calina Ciuhu-Pijlman, Dr. Mounir Zeitouny, Prof. Dr. ir. Gerard de Haan, thank you for your daily valuable advise, inspiration and guidance. I am also thankful to Jurgen Rusch, for introducing me to efficient programming to Python and Patrick Kechichian for his useful support with Android applications.

Thank you!

# Contents

Contents	iv
<b>1 Introduction</b>	<b>1</b>
1.1 Beauty in digital world . . . . .	1
1.2 Description of the problem . . . . .	3
1.3 Outline of the report . . . . .	4
<b>2 Human Skin</b>	<b>5</b>
2.1 The skin . . . . .	5
2.2 The hair . . . . .	6
2.3 Skin morphology . . . . .	6
2.3.1 Micro Scale . . . . .	6
2.3.2 Meso Scale . . . . .	7
2.3.3 Macro Scale . . . . .	8
2.4 Conclusion . . . . .	8
<b>3 System architecture overview</b>	<b>9</b>
3.1 Architecture of the system . . . . .	9
3.1.1 Prototyping the scanner device . . . . .	9
3.1.2 Lens system . . . . .	10
3.1.3 Gyroscope . . . . .	11
3.1.4 Data acquisition from the smartphone application . . . . .	12
3.1.5 DataLogger Output . . . . .	13
3.1.6 Synchronization between data and calculation of roll, pitch and yaw angles . . . . .	13
3.2 Conclusion . . . . .	15
<b>4 Image Stitching</b>	<b>16</b>
4.1 Map Construction . . . . .	16
4.2 Stitching algorithm . . . . .	18
4.3 Conclusion . . . . .	19
<b>5 3D visualization of skin body areas</b>	<b>20</b>
5.1 3D model reconstruction . . . . .	20
5.1.1 Rotation matrices . . . . .	20
5.2 Theoretical Background . . . . .	22
5.3 The Tool . . . . .	23
5.4 The Grid . . . . .	23
5.5 The Texture . . . . .	23
5.6 Algorithmic Implementation of 3D Reconstruction . . . . .	24
5.7 Conclusion . . . . .	25

---

<b>6</b>	<b>Map Navigation</b>	<b>26</b>
6.1	Map Navigation . . . . .	26
6.2	Theoretical Background . . . . .	26
6.2.1	Feature Detector . . . . .	27
6.2.2	Matching . . . . .	27
6.3	Implementation of Map Navigation . . . . .	28
6.3.1	Algorithmic Implementation of the map navigation. . . . .	29
6.4	Conclusion . . . . .	29
<b>7</b>	<b>System specifications identification</b>	<b>30</b>
7.1	Measured system and evaluation . . . . .	30
7.1.1	The DoE tool . . . . .	31
7.1.2	DoE plots . . . . .	31
7.2	Implementation of our DoE . . . . .	35
7.2.1	Parameters . . . . .	35
7.2.2	Results . . . . .	37
7.3	Conclusion . . . . .	41
<b>8</b>	<b>Conclusions</b>	<b>42</b>
	<b>Bibliography</b>	<b>43</b>
	<b>Appendix</b>	<b>44</b>
<b>A</b>	<b>3D Reconstruction Code</b>	<b>45</b>
<b>B</b>	<b>Map navigation Code in use for a DoE</b>	<b>50</b>

# Chapter 1

## Introduction

Cosmetic and toiletries have a significant and interesting part in human history and evolution and they are used in order to improve the appearance, health and even the scent. In particular, hair removal plays an important role in the improvement of appearance where various rituals are often adopted to the society's aesthetic preferences and needs. Many hair removal practices have been constantly renovated in order to suit every time to the latest fashion trends. According to some historical findings, these practices begun with sharp rock blades made of flint which were used to scrape hair from the body (something like the first razors) and nowadays there are electrical devices for shaving and epilation or even devices that use Intense Pulsed Light (IPL) technology for home-use, like in Figure 1.1.

The personal care industry worth over USD 300 bn globally and a very big part of it, is home-use beauty devices [1]. This indicates that there is a significant opportunity for cosmetic companies to focus on this sector in order to capture and increase the market share. Hair removal is one of the most promising areas in this market and this leads to a big competition among the companies. The hair removal devices provide to the consumers levels of satisfaction comparable with professional beauty treatments and at the same time they offer greater privacy and personal convenience. Also, the advantage of personal hair removal devices is the reduction in cost of maintaining hair-free skin for extended periods.



Figure 1.1: Lumea IPL Precision Plus Hair Removal System from Philips

### 1.1 Beauty in digital world

In the recent years we see a steady growth of digital technology, alongside beauty and cosmetics. Facial recognition, predictive analytics technology and other technological advancements are starting to be used into the beauty industry in the digital world. In addition, there is a serious industry turn-on for beauty lovers with the use of mobile devices smartphone, tablet and hybrid

phablet. With more consumers using their mobile devices to browse and buy on the go, the beauty industry is trying to get to grips with the trend.

Smartphones nowadays with the availability of various gadgets can have multiple uses. Apart from being a normal communication device, they can be used like photo cameras, game consoles and with the use of some additional kits they can turn into a tool for personal care (e.g. skin diagnostic with the use of dermoscopy device). With the current computation power and sensing capabilities of mobile phones, we expect to see innovative functionalities in the hand of consumer, defining a new way of interaction between human and devices/computers.

The use of smartphones has just started to expand in the cosmetics industry too. A lot of applications have been created from cosmetic companies which are using facial detection features and face analysis. An advanced one is L’Oreal makeup genius app, Figure 1.2 [2]. This application works like a mirror on the smartphone and allows the user to see the result of different makeup styles directly to her physiognomy and share this experience with friends. Firstly, the application maps the user’s face and captures 64 facial data points and up to 100 facial expressions. After that procedure there is the option of adding several filters on the image. These filters are the different makeup styles that can be virtually applied and they stay in place while the user moves, smiles, gesticulates and takes selfies.



Figure 1.2: L’Oreal makeup genius app

A similar application that uses face features is Philips Grooming: Shave & Style app [3], Figure 1.3, which deals with men shaving styles. This app uses the device’s front camera to take a selfie. Then it uses an algorithm that is able to recognize the position of the face (the distance from eyes, ears nose and mouth). After the recognition of the face’s features it applies with accuracy several virtual beard styles, creating a virtual look.

Although various applications have been created for cosmetic purposes none of them benefits the use of sensors in smartphones. In this project we are going to benefit from these sensors. We are going to make an application that will benefit from the camera and the sensors in order to monitor the angular position of the device during its use. This application will be used for 3D reconstruction of body parts and body tracking and localization of the recorded body areas.



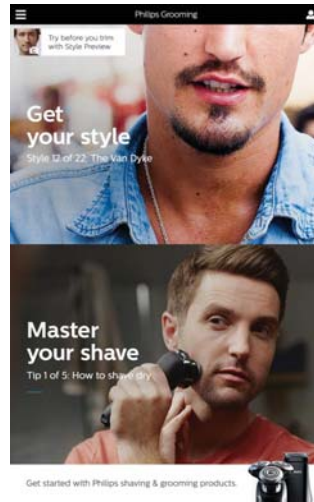


Figure 1.3: Philips Grooming: Shave &amp; Style app

## 1.2 Description of the problem

The treatment with an IPL device should be done on hair-free areas. Therefore, while treating, it is not possible to notice the progress of the treatment. Because of this, the users are struggling to find missed spots, especially in body parts that they cannot see (like the back of the leg during hair removal). This leads into unsatisfactory results in the end of the treatment. The aim of this project is to provide a solution for treatment-based personal beauty devices like IPL, when the user can obtain guidance, according to the usage.

In order to avoid missed spots or under treated areas the users are advised to over treat the target area by making sure that the device footprints are partially overlapping. This in turn is providing a hassle for the consumer since longer time is needed to finalize the treatment of the target area. Especially, when the treatment is used on large body areas like legs, chest and back the problem is bigger since the users have not visual contact with the treated area.

The aim of this project is to address the above problems by embedding a camera on a consumer device in combination with position/motion sensors. Then, algorithms will be designed and built enabling analysing, searching and visualization.

Apart from body tracking, this additional functionality on the personal care devices will bring significant benefits to the users. In order to achieve a more appealing representation of body tracking to the user, this project also aims to succeed the 3D reconstruction of the body using the sensors and the camera of a smartphone. In this way the credibility of the device will be increased since it will get more in connection with the user. Moreover, important information about treatment results and knowledge on how people are using the devices can be collected. This kind of information is very valuable since it is useful for other research purposes to improve users satisfaction.

Providing a solution for 3D body reconstruction and skin navigation is not straight forward. There are some challenges that we need to address:

- constructing a map is not new. Examples are map construction from automotive industry, 3D panoramic images etc. In our case the challenge is to build the map for skin. We will need to find out the right features and therefore understand the skin morphology and how to image it.
- second challenge regards building up our solution which is suitable for the requirements of a consumer device. For this we need to address the system architecture and make the right choices.

- another difficulty is related with the stitching and creating a map from stitching. The question here that we have to provide a solution is how to adjust the parameters of a stitching algorithm to a system with low specifications that we could use in a consumer device.
- the puzzle challenge is to be able to find the registration parameters in order to calculate the 3D coordinates of the scanned item and achieve the right folding and represent in a 3D digital way the exact shape of the item
- another question addresses the finding of a method that is fast and that complies with the memory usage.

### 1.3 Outline of the report

The outline of this report is the following. Chapter 2 elaborates on skin, hair and skin morphology. Chapter 3 presents the System Overview, it describes the architecture of the system that is constructed during that project and explains in details the software modifications that are made in order the created system to have the needed functionality. Chapter 4 presents details of the stitching algorithm that was used in order to be able to continue with the implementation of the 3D reconstruction and the map navigation. Chapter 5 describes the procedure that we followed for achieving the 3D body reconstruction. Chapter 6 explains the methods that we used for the map navigation. The next chapter contains the Design of Experiment which selects the system specifications. Finally, conclusions for the whole project are made in Chapter 8 and some recommendations for future work are also included.

## Chapter 2

# Human Skin

In this chapter we provide the reader with an overview regarding the properties of skin, hair and skin's morphology. This background is necessary as the project targets the development of devices that are applied on the skin area. This brief research on the skin characteristics will help us to understand which are the strongest and most stable features that we can use in order to succeed an accurate body tracking.

### 2.1 The skin

The skin is the largest organ of the human body. It forms an effective barrier between the organism and the environment. The purpose of this barrier is to protect the body from pathogens, to fend off the chemical and physical invasions, along with the avoidance of unregulated loss of water and solutes. Also, it is considered to be a sensory organ because it allows us to feel stimuli from the environment like temperature, texture, pain, pressure and others, it produces vitamins and prevents loss of moisture from the body. The skin consists three layers the epidermis (outer layer), the dermis and the subcutaneous layer.

The epidermis is the outer layer of the skin and it consist 5-6 layers. The layer that we can see is called stratum corneum. It is the part which interfaces with the environment and consist dead, keratinized cells. Epidermis is composed up to 30% of water and it has ph 4.5-6 depending on the part of the body. The amount and distribution of melanin pigment is responsible for the variations of the skin color. Melanin is found in the small melanosomes, particles formed in melanocytes from where they are transferred to the surrounding keratinocytes. The exposure on UV radiation increases the number of melanosomes in the keratinocytes and this is the reason of skin tanning.

The dermis, also called corium, is the thicker, deeper layer of skin beneath the epidermis. It is made up of connective tissue and cushions the body from stress and strain. It is about 80% of water and provides the skin with tensile, strength and elasticity through an extracellular matrix composed of collagen fibrils, microfibrils and elastic fibers embedded in proteoglycans. Dermis is also responsible for the sense of touch and heat, hair follicles, sweat glands, sebaceous glands, apocrine glands, lymphatic vessels and blood vessels. It is divided into two layers the superficial area adjacent to the epidermis called the papillary region and a deep thicker area known as the reticular dermis. In humans the dermis projects into the overlying epidermis in ridges called papillae. Through the dermis and in the end of the papillae there are the nerves, which are sensitive to heat, cold, pain, and pressure. In the deeper stratum reticulare there are the sweat and oil glands, the bases of hair follicles, the nail beds, and blood and lymph vessels.

The next layer is the subcutaneous tissue or hypodermis. It is not part of the skin, and it lies below the dermis. The aim of this layer is to connect the skin with the underlying bone and muscle and also to provide blood vessels and nerves to it. It consists of loose connective tissue and elastin. Finally, the hypodermis contains 50% of body fat which is used as padding and insulation for the body.

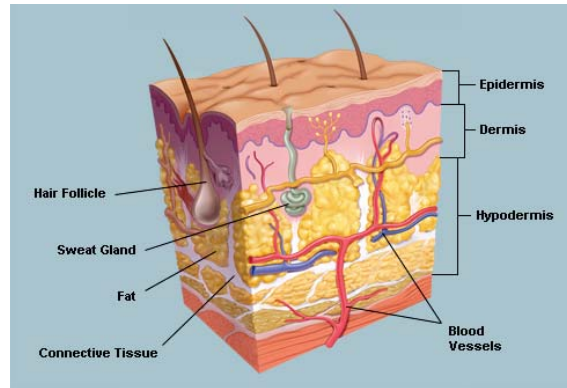


Figure 2.1: Basic Human Skin Anatomy [9]

Apart from its undeniably significant contribution on body function, skin is also an important way of transmitting information. Skin is one of the first things that we notice when we meet a person and it informs us about the person's age, nutritional and psychological status even for the person's health and personal hygiene. This is the reason why people pay much attention to skin care [5], [6], [7], [8].

## 2.2 The hair

Human beings like all the mammals have hair. Hair is the characteristic capilliform that grows in the epidermis. Mammalian hair consists two parts. One part is a shaft that protrudes above the skin and the other part is the root, which is positioned in a follicle beneath the skin surface. Hair is mostly a dead tissue, composed of keratin and related proteins. The human hair is formed by divisions of cells at the base of the follicle. After the upward thrust of the cells from the follicle base, they become keratinized and undergo pigmentation. Hair is following alternating cycles of growth, rest, fallout, and renewed growth. The average life of different varieties of hair varies from about 4 months to 3 or even 5 years. Human hair grows at the rate of about one-third of a millimeter per day. This rate creates a need of hair removal methods that they can be effective, quick and to endure as much as it is possible.

## 2.3 Skin morphology

In order to examine the morphology of the skin, we need to focus on the optical/visual feature keys of the constituent components of skin. A comprehensible way of a hierarchical representation of these components is based on the scale of the optical processes included by them. In this report, the smallest components are referred as micro scale, bigger components as meso scale and the biggest components as macro scale. Each scale is divided into other levels based on physiology and anatomy.

### 2.3.1 Micro Scale

Skin layers and cellular level elements compose the scale of the physio-anatomical skin structure. These components are hardly visible with bare eye and their visual properties are result of their optical interaction with incident light. The optical properties in this level are scattering and absorption and the only way to be studies is by using wave optics. The skin layers have different physiological features from each other, since they have very different structure. These differences





Scale	Level	Physiological/Anatomical Components
Meso	3	<p><b>Skin</b></p> <ul style="list-style-type: none"> <li>• skin surface lipid</li> <li>• hair</li> <li>• skin layers</li> <li>• fine wrinkle</li> </ul>  <p>skin layers</p>  <p>fine wrinkle and hair</p>
	4	<p><b>Skin Features</b></p> <ul style="list-style-type: none"> <li>• wrinkle</li> <li>• pore</li> <li>• mole</li> <li>• freckle</li> </ul>  <p>wrinkles</p>  <p>freckles</p>

Figure 2.2: Meso Scale [7]

affect the light propagation and lead to very different optical effects, like the epidermis is a transparent optical medium and the dermis is a turbid medium. These optical differences are very useful to describe the optical properties of higher scale components. In this project we will not deal with this part of the skin.

### 2.3.2 Meso Scale

The meso scale components are skin and skin features and they can be seen with bare eye. Skin consists of skin layers, skin surface lipid, hairs, fine wrinkles, etc. and its appearance can be seen as the combination of the optical phenomena induced by these substructures. On the surface of the skin can be found skin surface lipids, hairs and fine wrinkles and all these components give interesting optical effects. Except the above features, meso scale includes also higher level components like freckles, moles, wrinkles and pores. These characteristics can be viewed as morphological variations of skin and they contribute in the visual properties of skin features.

Scale	Level	Physiological/Anatomical Components
Macro	5	<p><b>Body Regions</b></p> <ul style="list-style-type: none"> <li>• nose</li> <li>• finger</li> <li>• elbow</li> <li>• knee</li> </ul>  <p>nose</p>  <p>elbow</p>
	6	<p><b>Body Parts</b></p> <ul style="list-style-type: none"> <li>• face</li> <li>• arm</li> <li>• leg</li> <li>• torso</li> </ul>  <p>face</p>  <p>arm</p>

Figure 2.3: Macro Scale [7]

### 2.3.3 Macro Scale

The macro scale components are body regions and body parts. The appearance of skin varies across different regions of the body because the physio-anatomical characteristics of the lower-level components can differ significantly from one region of the body to another. Body regions are divided into body parts like face, arm, leg and torso.

The appearance of each body part includes the appearances of the body regions that constitute it. Until now, there is not a unified framework which can describe the appearance variations over the body in a form of a physical model. The aim of that project is the creation of models in a 2D and a 3D way. These models will illustrate the shape of the human body and they will describe the appearance variations of the different body regions. In that way it will be easier to identify the part of the body during a body scanning.

## 2.4 Conclusion

In this chapter we briefly addressed some of the fundamental knowledge on human skin. This information will help us to set up our prototype e.g. in term of lighting conditions as well as recognize various detected skin features e.g. skin pores and understand their stability. Therefore, for the body tracking application we are going to use the meso scale components which are more stable, like pores, moles and freckles. In the next chapter we will give an overview on the prototype we assembled for this project.

## Chapter 3

# System architecture overview

In this chapter we introduce the reader to the architecture of our prototype scanning device. The main goal of this project is to construct a 3D body model and be able to navigate it. This means recognizing in a consumer-like device in real time the part of the anatomy where the device is located. Hence, the first is to construct a scanning system that it will be able to give the right skin image as an input in combination with the angles of the body area that was scanned. We will focus on the hardware component of the system together with the required software methods and their functionalities. This system will be able to full-fill two main purposes. The first purpose is the angle acquisition and reconstruction of a 3D body image. The second purpose is the achievement of the skin navigation for the body tracking application.

### 3.1 Architecture of the system

We aim at developing a system enabling the representation of the body skin surface in 3D. The 3D scanning over the body requires images that are later processed to formulate a 3D shape as illustrated in Figure 3.1. To achieve this goal we first develop a scanner that acquires 2D video. Moreover, as discussed in Section 2.3, we need to register skin morphology, that will later be used to distinguish between skin patches.

As the scanner will be moving over the skin we need to use sensors in order to enable 3D body reconstruction. For that, we chose to use a gyroscope synchronized to the image frame rate having the image information and the motion sensors one can rely on image processing software in order to achieve acceptable 3D representation of the body. The software routines that are needed to be developed will be discussed in the next sections.

First, we start introducing the individual hardware components with their functionality and a brief explanation of image registration that will lead us to a full 3D body reconstruction.

#### 3.1.1 Prototyping the scanner device

The scanning device that we are going to build should be constructed with simple hardware that everybody could use. For that reason we decided to use a smartphone Samsung galaxy s5. The phone features a 16MP camera with resolution of 5312 x 2988 pixels and embedded accelerometer and gyroscope sensors [10]. As it outlined in the previous section, in order to be able to navigate on the created body map we need to use the data from those sensors and for that reason we will use an already existing application.

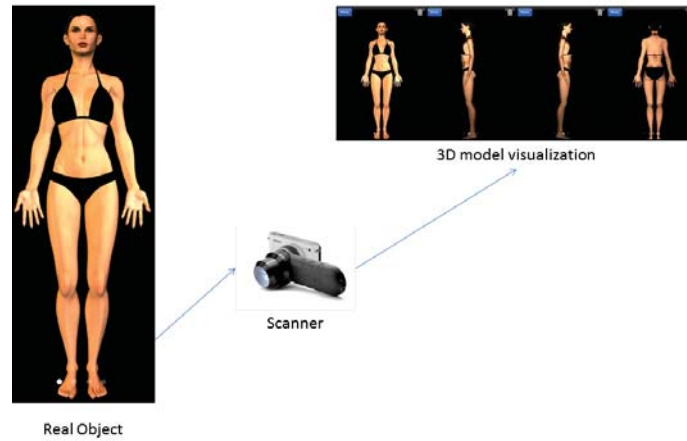


Figure 3.1: 3D scanning procedure



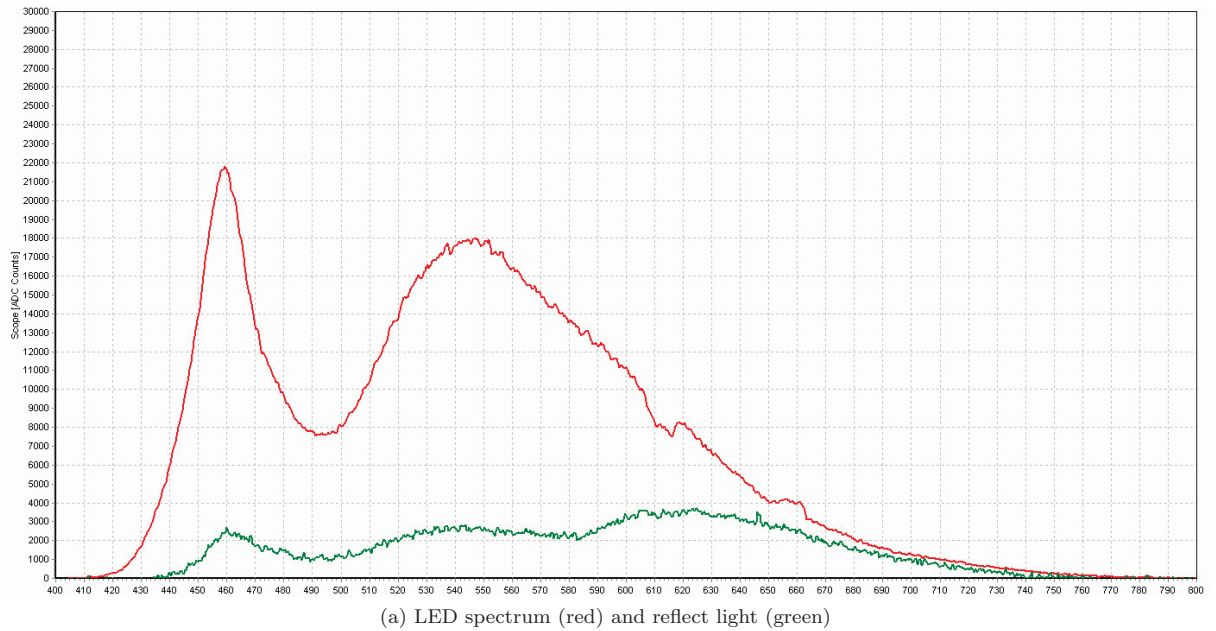
Figure 3.2: Scanner Device

### 3.1.2 Lens system

As we require to image skin morphological structures, using a phone camera alone will not allow us to obtain images of the skin with good resolution, as shown in Figure 3.3(b). Therefore, we acquired a lens system designed initially to be used in clinics for skin diagnosis and can be used with a smartphone device.

The device that we chose to add in our system is DermLite DL3, as it shown in Figure 3.2. It is a dermoscopy epiluminescence device, with a retractable faceplate spacer design and a focusing mechanism. The faceplate spacer has a twisting large precision dial and it can focus on the image with a focal range in excess of  $\pm 4\text{mm}$ . It comprises a LED light that illuminates the skin by providing cross-polarized and parallel-polarized light, Figure 3.3(a) shows the LED spectrum (red) and reflect light from the skin (green). This illumination contains blue colour which means it can show clearly the skin surface and also the superficial internal structures of the skin, which can be very helpful for finding the skin key descriptors for the map navigation. The result of the dermoscope's use in combination with the smartphone and the android application can be shown in Figure 3.3(c).





(b) Image without Dermoscopy Device.



(c) Image with Dermoscopy Device.

Figure 3.3: Light spectrum and Images with and without the lens system

### 3.1.3 Gyroscope

In addition to images of the skin, we are going to need the angular position of the pixels in order to fold a 2D image into 3D. For this reason we are going to use a gyroscope which is a sensor for measuring the angular rate orientation. It can maintain its level of effectiveness by being able to measure the rate of rotation around a particular axis. When the device is not rotating, the values that it gives are zeroes. In order to estimate the angle from gyroscope's values, the angular rate signal must be integrated with respect to time. It gives as an output 3 values:

- Roll: movement of the device around x-axis
- Pitch: movement of the device around y-axis

- Yaw: movement of the device around z-axis

Note that the image plane is in the  $xy$ -plane, for comparison see Figure 3.6.

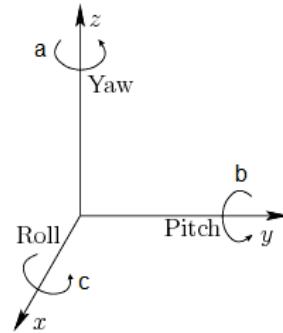


Figure 3.4: Three-dimensional rotation described as a sequence of yaw, pitch, and roll rotations [29].

### 3.1.4 Data acquisition from the smartphone application

The role of the application is to collect data from the phone's sensors. One of the main goals of this project is to construct a 3D body model. In order to succeed that goal, we will use an android application. This application is named Sensor Data Logger [11] and it is a sensor data logging Android application that gathers measurements from all the available phones sensors while recording a video, same as in Figure 3.5.



Figure 3.5: Interface of the SensorDataLogger application [11].

The coordinate system in the Android phone's is defined relative to the screen of the phone in its default orientation. This system is shown in Figure 3.6 and the axes are not swapped when the device's screen orientation changes. The X axis is horizontal and points to the right, the Y axis is vertical and points up and the Z axis points towards the outside of the front face of the screen. In this system, coordinates behind the screen have negative Z values [16].

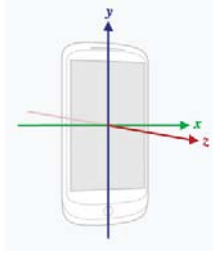


Figure 3.6: Coordinate system [16]

We first focus onto understanding the data that obtained from the android application. Our focus was mainly to determine the phase delay between the various sensors because the sensors and the camera do not start at the same time. This is because the device do not record data at equal timestamps. We explain the necessary steps to be taken in order to correct for data synchronizations in the subsequent chapters. Then, we analyse this data to find out which was the corresponding angle for every video frame. Those steps are being explained in the following sections. After that we are ready to continue with the 3D processing and the map navigation.

### 3.1.5 DataLogger Output

The SensorDataLogger application creates a folder which contains 26 files (in .txt format). From these files we are only interested in the file obtained from the gyroscope sensor. As it is shown in Table 3.1 this file contains some data values. These values represent the following [15], [16]:

**Timestamp:** in nanoseconds

**Sensor:** The number of sensor that generated this event (gyroscope=3).

**Roll:** angular speed around x-axis (rads/sec)

**Pitch:** angular speed around y-axis (rads/sec)

**Yaw:** angular speed around z-axis (rads/sec)

Table 3.1: Data from gyroscope file

Timestamp	Sensor	Roll	Pitch	Yaw
1418291826461411652	3	-0.005060006	-0.0013315806	-0.009321064
1418291826481580766	3	-5.326322E-4	0.007190535	-0.011984225
1418291826503461287	3	-0.003728456	-0.01438107	-0.0055926386

### 3.1.6 Synchronization between data and calculation of roll, pitch and yaw angles

After recording a video with the Sensor Data Logger application, we noticed that there was a big difference between the number of data in the gyroscope file and the number of video frames. For example, a video with length 15 seconds, since the camera that we are using has a frame rate 30 fps (frames per second), will contain 453 video frames. Although, the gyroscope will give as an output 865 different data. This means that the sampling time of the gyroscope is 1.9 times faster than the frame rate of the camera. This inability in synchronization prevented us from finding the corresponding data for every video frame. In order to overcome this problem we did a slight modification in the android application.

Since we were not able to change the sampling time of any of the components (hardware decisions) we decided to take advantage of the timestamps. In the beginning we made the application to print in the name of the video file the timestamp of the beginning time of the video. In this way, considering that we know the frame rate we can calculate the timestamps for all the video frames. Thereafter, we used the closest distance method to synchronize them with all the data from the sensors.

Another step that we had to implement before proceeding into the synchronization of the data, is the integration of the values that the gyroscope gives as an output. The gyroscope gives us the rate of change of the angular position over time (angular velocity) with a unit of [rad/s]. This means that we get the derivative of the angular position over time.

$$\dot{\theta} = \frac{\partial \theta}{\partial t}. \quad (3.1)$$

where  $\theta$  can be roll, pitch, yaw.

To obtain the angular position, we have to integrate the angular velocity. So, assuming that at  $t=0$   $\theta=0$ , we can find the angular position at any given moment  $t$  with the following equation:

$$\theta(t) = \int_0^t \dot{\theta}(t) dt \approx \sum_{i=0}^T \dot{\theta}(t)(t(i) - t(i-1)). \quad (3.2)$$

In digital systems we have to do an approximation of integration, which is shown in the third part of Equation 3.2. We are taking the sum of a finite number of samples taken at the according time interval of each time.

After all the steps above, we can continue with the synchronization of the data. This method is comparing the timestamp of each video frame with the timestamps from the gyroscope files and finds with which one has the smallest difference. After that, we create a new file which contain only the data that we need for the synchronization. The algorithm that we made for calculating the corresponding yaw, pitch and roll for all the video frames, is shown in the Figure 3.7.

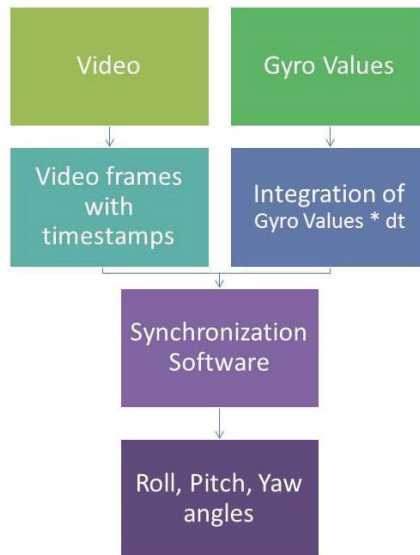


Figure 3.7: Flow chart for calculating the right angles

## **3.2 Conclusion**

In this chapter we described our scanning device from the system architecture point of view. We explained the hardware components that we used and the software modifications that we made in order to have the right functionality. After this step we are able to use the device in contact to the skin for recording videos and store the angle information of the device's motion during the video. In the next chapter we are going to describe the theoretical details of the software that we will use in order to convert the video into a single 2D image. This image is going to be used for the 3D reconstruction and the map navigation purposes.

## Chapter 4

# Image Stitching

After explaining the hardware components that we are going to use in the project, we will continue with the description of the theoretical details and the implementation of the software that we will use for the 3D reconstruction and the map navigation. In order to be able to implement them we need as an output a stitched image. This chapter explains the theoretical background of stitching procedure and present the already built in-house stitching algorithm that we are going to use for this method.

### 4.1 Map Construction

The skin map reconstruction refers to any method that is used and applied to video frames in order to convert them to a single image without any overlap between originally acquired adjacent frames. The body map is constructed by stitching the sequential frames from a video, using the technique of panorama. In this technique a big image is constructed by a video from a rotating camera. Between every two sequent frames there is an overlapping area. An image with bigger view can be created by aligning the overlapping area of the two frames. An automatic panorama algorithm can be used for this purpose which uses the scheme of feature match. This algorithm uses the panorama workflow shown in Figure 4.1.



Figure 4.1: The workflow of panorama [4].

The methods that will be used are the following:

- **Tracking:** in this step feature points are selected from the images, which are pixels with large intensity gradients. There are many algorithms that have been developed for tracking and they are referred to corner detectors. The stitching algorithm [4] that we will use in our project uses the corner detector named FAST [12], which is a fast and robust tracker and it can provide us a lot of candidates features for the subsequent filtering. In tracking the algorithm uses only the top part of every frame with size  $720 \times 109$  pixels (*width*  $\times$  *height*). This happens because since we are using a video that means that the frames have a big overlapping.

As it is shown in Figure 4.2, for each location on the circle  $x \in \{1...16\}$  around the pixel  $p$ , a state function is given as:

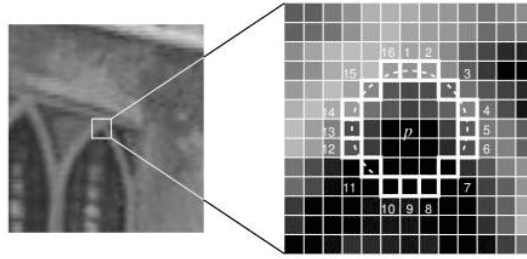


Figure 4.2: The FAST algorithm examines a circle of 16 pixels around the pixel under test. [4].

$$S_x = \begin{cases} \text{d,} & I_x \leq I_p - t, \\ \text{s,} & I_p - t \leq I_x \leq I_p + t, \\ \text{b,} & I_p + t \leq I_x, \end{cases} \begin{matrix} \text{dark} \\ \text{similar} \\ \text{right} \end{matrix} \quad [4]$$

where  $I_p$  is the intensity of pixel  $p$ ,  $t$  is a threshold value and  $I_x$  the intensity of each pixel in these 16 pixels.

The pixel  $p$  is considered as a corner if there exists a set of 12 contiguous pixels in the circle (of 16 pixels) which are all brighter than  $I_p + t$  or all darker than  $I_p - t$ .

- **Matching:** the next step is the extracted feature points from two images to be matched and this can be done with two different strategies. One strategy is to extract a feature vector of a pixel from its neighbours and match it by defining and minimizing a cost function [13]. This matcher also gives an indication about the quality of matching, like the distances between the feature points. The other strategy is to match the images by directly comparing them and find the local minimum of the normalized correlation function [14]. The stitching algorithm that we will apply uses the method of optical flow [14] to match the detected features. This method was chosen because the displacement between the sequential video frames is very small.

The optical flow tries to minimize the following equation:

$$\epsilon(\delta_x, \delta_y) = \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} (I_1(u_x, u_y) - I_2(u_x + \delta_x, u_y + \delta_y)). \quad (4.1)$$

$[u_x, u_y]$  is the point in image  $I_1$  and  $[u_x + \delta_x, u_y + \delta_y]$  is the point in image  $I_2$ .

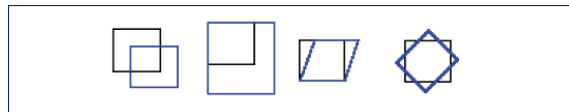


Figure 4.3: The four basic types of transformation: image translation, image scale, image shear and image rotation (from left to right) [4].

- **Match filtering and transformation estimation:** in order to have a better estimation of the transformation, a filtering of the matcher's result is needed. There are four basic types of transformation, as they can be seen in Figure 4.3 and also different combinations of them. Each type has different degrees of freedom which can influence the estimation's accuracy (the more degrees of freedom, the more noise is accepted). The stitching algorithm that we will apply to our project uses translation transformation. Since the video is taken

with linear strokes so the transformation model only includes the displacement of x and y direction ( $t_x$  and  $t_y$ ). Rotation or other deformations are excluded because the skin under the camera does not deform much in a stroke. It is always under the same pressure. The transformation equation is:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}. \quad (4.2)$$

- **Image transformation and blending:** image transformation represents a given image as a series summation of a set of matrices. In our case it can be expressed as a  $3 \times 3$  matrix. The image can be denoted as  $G(x; y; z)$ , where z means the distance between the observer and the image. In our case, z is kept as 1 since the distance is always stable. If we denote the transformation matrix as T, then the transformed image can be written as  $F(x; y; z) = G(T(x; y; z))$ . Finally, an interpolation algorithm is performed in order to fit the new image map into a grid map and to convert the color values into integers [4].

## 4.2 Stitching algorithm

Since we have already constructed the scanning device and synchronizing the smartphone's sensor with the camera, we are able to continue with the stitching part. For this part we used an already existing in-house stitching algorithm [4], which stitches video frames into an image. This algorithm follows the methods that we followed in Chapter 4.1.

In order to see how the algorithm works with the parameters of our system (different camera resolution, lower video frame rate) we scanned and then stitched different body areas, like face, hand and leg and we assessed the obtain resolution. All of the locations had satisfactory stitching results since the detection of skin features was possible. The best result was the one of the leg area, like it is shown in Figure 4.4(b) (which is our main target) where the stitching algorithm was able to detect a lot of skin features, as it is shown in Figure 4.4(a).

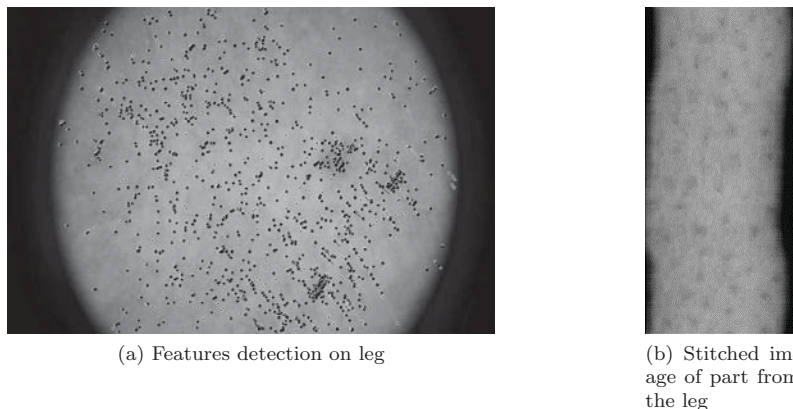


Figure 4.4: Samples from stitching algorithm

The stitching algorithm apart from the stitched image, it has one more output which is file with information from the stitching procedure. This file, as it is shown in Table 4.1, contains data from the parts of the image in x and y coordinates that correspond to each video frame. This data will be very useful for the 3D reconstruction of the stitched image.



Table 4.1: Data from csv file

# frame	x[pxl]	y[pxl]	dx[pxl]	dy[pxl]
1	100	5000	0.03897	-0.04600
2	100.03897	4999.95399	0.98084	-2.32974
3	101.01981	4997.62425	-0.16301	-0.163263
4	100.85680	4997.46099	0.05068	-0.103688

**# frame:** from which video frame where retrieved the image data on the stitched image.

**x:**  $x$  coordinate of the stitched image that the frame's data begin.

**y:**  $y$  coordinate of the stitched image that the frame's data begin.

**dx:** relative displacement on  $x$  direction in respect to the previous video frame.

**dy:** relative displacement on  $y$  direction in respect to the previous video frame.

This is a short example of the results of the four first frames of the video indicating for each location  $x$  and  $y$  the corresponding displacements between two successive video frames,  $dx$  and  $dy$ . The values are represented in pixel units indicating for this case that the motion is in sub-pixel per frame.

### 4.3 Conclusion

This chapter described the stitching procedure that we used in order to make a stitched image after using the scan device to record a video of the skin. The algorithm detects features in every video frame, finds the matchings and creates an image of the recorded area. After finishing with the stitching algorithm, we can continue with the core parts of the project. We are going to use this stitched image as an input in order to make a 3D body reconstruction and skin navigation.

## Chapter 5

# 3D visualization of skin body areas

In this chapter we discuss the theoretical work behind the 3D folding algorithm that we developed. We are going to use the stitched image that we created in Chapter 4 as an input in combination with the angles that we calculated in Chapter 3. Our 3D reconstruction is based on combining two channels of data, namely the video stream with the gyro sensor data. We mathematically and algorithmically describe each required step with some examples and sketching when necessary to help the reader better understand the full implementation of the method.

### 5.1 3D model reconstruction

Three-dimensional (3D) reconstruction is the virtual representation of the shape and appearance of a real object in 3D space using computer vision and computer graphics. There are a lot of different ways that this can be done but the most usual one is by using as an input two-dimensional (2D) images or the scanning of the actual object. The reconstructed 3D models can be manipulated or utilized in several ways, including medical uses, law enforcement reconstructions, or even the creation of 3D graphics for cinema and television. The 3D processing is divided into two parts: the construction of a grid and then the texture.

- The grid: the first step for creating a 3D image is to make the shape of the object that we want to represent in a 3D visual way. The 3D shape of the object can be a polygon grid, which is a complex of vertices and polygons [30].
- The texture: after creating the 3D-shape of the object that we scanned, we have to add the texture from the right parts of the stitched 2D-image into the new 3D-grid.

Figure 5.1 shows the 3D representation of a human leg with and without the visual use of the grid. In order to fold a 2D image into a 3D shape, the suitable rotation matrices need to be used.

#### 5.1.1 Rotation matrices

A 3D body and the local orientation of the body surface (e.g. skin) can be rotated about three orthogonal axes, as shown in Figure 3.4. These rotations are referred as yaw, pitch, and roll and their mathematical use of the given angles are the following [17]:

- A **yaw**  $\alpha$  is a counterclockwise rotation of  $\alpha$  about the  $z$ -axis. The yaw rotation matrix is given by Equation 5.1.



Figure 5.1: 3D representation of human legs [31].

$$R_z(\alpha) = \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 0 \end{pmatrix}. \quad (5.1)$$

- A **pitch**  $\beta$  is a counterclockwise rotation of  $\beta$  about the  $y$ -axis. The rotation matrix is given by Equation 5.2.

$$R_y(\beta) = \begin{pmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{pmatrix}. \quad (5.2)$$

- A **roll**  $\gamma$  is a counterclockwise rotation of  $\gamma$  about the  $x$ -axis. The rotation matrix is given by Equation 5.3.

$$R_x(\gamma) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \gamma & -\sin \gamma \\ 0 & \sin \gamma & \cos \gamma \end{pmatrix}. \quad (5.3)$$

In order to place a 3D body in any orientation, yaw, pitch, and roll rotations can be used. A single rotation matrix can be formed by multiplying the yaw, pitch, and roll rotation matrices. The rotation matrix is given by Equation 5.4.

$$R(\alpha, \beta, \gamma) = R_z(\alpha)R_y(\beta)R_x(\gamma) = \begin{pmatrix} \cos \alpha \cos \beta & \cos \alpha \sin \beta \sin \gamma - \sin \alpha \cos \gamma & \cos \alpha \sin \beta \cos \gamma + \sin \alpha \sin \gamma \\ \sin \alpha \cos \beta & \sin \alpha \sin \beta \sin \gamma + \cos \alpha \cos \gamma & \sin \alpha \sin \beta \cos \gamma - \cos \alpha \sin \gamma \\ -\sin \beta & \cos \beta \sin \gamma & \cos \beta \cos \gamma \end{pmatrix}. \quad (5.4)$$

Since these operations are not commutative, it is important to note that  $R(\alpha, \beta, \gamma)$  performs the roll first, then the pitch, and finally the yaw. A different rotation matrix results, in case the order of these operations is changed.

## 5.2 Theoretical Background

We explain the mechanism behind 3D folding where we first rely on generic illustrations for the sake of clarity. We use the stitched image and we aim to apply folding to transform it to a single 3D structure. The stitched image is initially in 2D where we associate the  $xy$  plane to it as a referencing plane. To get depth information, or 3D folding, we seek at associating with each  $[x,y]$ , pixel in the  $xy$  plane a  $z$  component. This can be derived using the angular information as obtained from the gyroscope after processing.

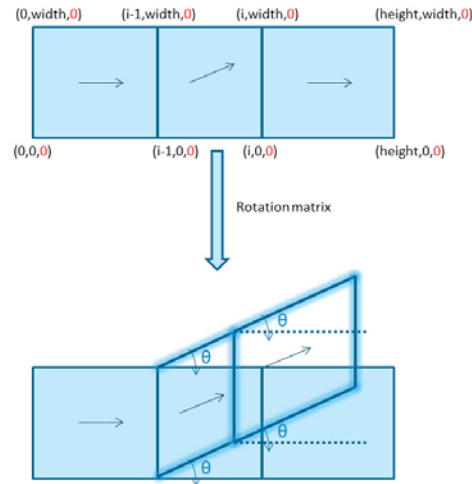


Figure 5.2: Theoretical idea for 3D folding

The mechanism is shown in Figure 5.2, first we are dividing a 2D image into segments, using the coordinates on the  $x$  and  $y$  axis from the csv file of the stitching algorithm (like the  $x$  and  $y$  from Table 4.1). The next step is to insert a third dimension in the image by giving for all the segments the value zero on the  $z$  axis. In order to introduce an angle into one of the segments, we are using the rotation matrix from Equation 5.4 and then we are giving the same rotation angle into all the following segments by using as a reference always the previous segment that we just rotated.

In the next parts we are going to explain the way of constructing a 3D body model by folding the stitched image that we have already obtained. We will use Java language in combination with the library OpenGL [20], which is used for rendering 2D and 3D vector graphics. The tool that we are going to use to program is Processing [21]. As we have already mentioned, the 3D processing is divided into two parts: the construction of the grid and then the texturing. In order to explain how the algorithm works, we are going to use the stitched image of a human jaw, shown in Figure 5.3.

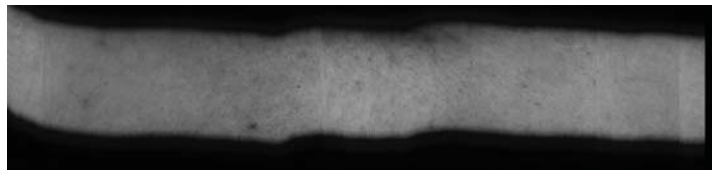


Figure 5.3: The 2D stitched image of a human jaw

### 5.3 The Tool

Processing is a software tool that combines programming with visual, motion and interaction principles. It is a program that can be used by students, artists, design professionals, and researchers for learning, prototyping, and production. The Processing language is text programming language designed for generating and modifying images. Many computer graphics and interaction techniques can be applied, including vector/raster drawing, image processing, color models, mouse and keyboard events, network communication, and object-oriented programming. Also, with the use of the libraries, Processings ability is easy to be extended to generate sound, send/receive data in diverse formats, and to import/export 2D and 3D file formats.

Processing was designed to make it easy to draw graphic elements such as lines, ellipses, and curves in the display window. These shapes are positioned with numbers that define their coordinates. The position of a line, for example, is defined by four numbers, two for each endpoint.

Processing has a specific default structure that gives a lot of sketching possibilities. The *setup()* and *draw()* functions make it possible for the program to run continuously in order to create animation and interactive programs. The code inside *setup()* function runs once when the program first starts, and the code inside *draw()* runs continuously. One image frame is drawn to the display window at the end of each loop through *draw()*. [32]

In order to create the 3D model we followed all the needed structures to make the grid of the object that we scanned and finally to add its texture.

### 5.4 The Grid

As we have mentioned before, the first step for creating a 3D image is to build the shape of the object that we want to represent in a 3D visual way. In our case, we are going to use the stitched image of Figure 5.3 and its corresponding csv file.

In the *setup()* function of our Processing program we are inserting the x and y coordinates of the stitched image from the csv file and also we are adding a z coordinate which always equals to zero. After this, we can use the *draw()* function to draw the right lines in order to make the representation of the size of the 2D stitched image in the 3D space. The result of this procedure is the one shown in Figure 5.4(a). Every segment of this shape corresponds to the part that we used from each video frame.

To continue with the 3D folding we are using the excel file with the corresponding angles for all the video frame that we made after synchronizing the data like we explained in Chapter 3. These angles with the use of the rotation matrices like we explained in Chapter 5.1.1, will give the 3D folding representation of the shape of the stitched image, as it can be seen in Figure 5.4(b).

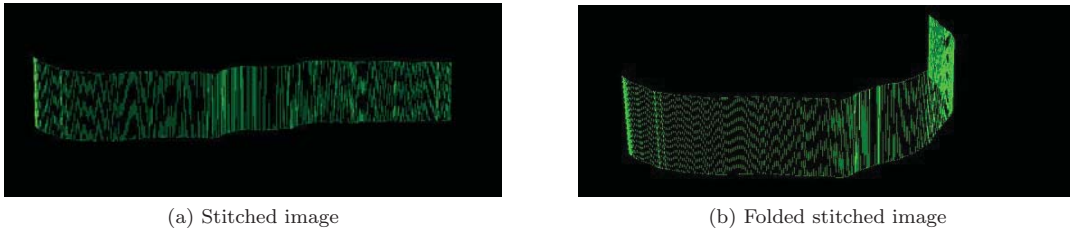


Figure 5.4: The 3D grid of a human jaw

### 5.5 The Texture

After creating the 3D shape of the object that we scanned, we have to add the texture from the right parts of the stitched 2D image into the new 3D grid. In order to succeed this, we had to

take every time the parts of the 2D image that correspond to each segment and insert it to the corresponding folded segments of Figure 5.4(b). The result of this procedure is Figure 5.5(a).

Finally, we remove from the visual part the lines that we used to separate the segments and we have the end-result of a 3D reconstructed image, like the one in Figure 5.5(b).

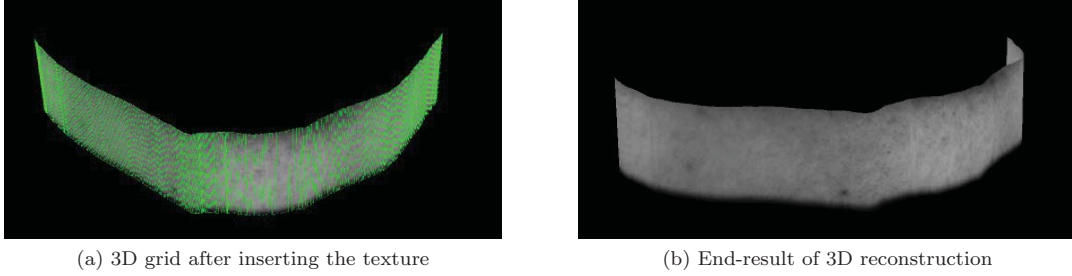


Figure 5.5: Texturing the 3D grid

## 5.6 Algorithmic Implementation of 3D Reconstruction

In a more algorithmic way the implementation of the 3D reconstruction that we already explained is the following. Algorithm 1 shows the *setup()* function in which the coordinates of all the segments of the folded 3D stitched image have been calculated.

---

**Algorithm 1** Algorithm of *setup()* function

---

- 1: Declare the size of the representation of the 3D object.
  - 2: Load the output file from the stitching (as it is shown in Table 4.1).
  - 3: Load the output file with roll, pitch and yaw angles after implementing the algorithm of Figure 3.7.
  - 4: Load a copy of the previous file with roll, pitch and yaw angles equal to 0.
  - 5: Read all the values of the files above.
  - 6: Store all the values from the excel files into 1D matrices.
  - 7: Initialization of all the matrices that we are going to use in the next steps.
  - 8: **for** k equal: total number of frames **do**
  - 9:     Create one matrix for the bottom and one for the top part of the image with the xy-coordinates of the stitching file and add z coordinate equal to zero.
  - 10:     Add these matrices into two lists for the bottom and the upper part of the image.
  - 11:     Retrieve from the lists from step(9) the matrices from the previous iteration (k-1).
  - 12:     Create a matrix  $A(x, y, z)$  for the bottom and one for the upper part of the image with the new xyz-coordinates after the multiplication of the coordinates from step(9) with the rotation matrices from Equation 5.5 where the angles  $\alpha, \beta, \gamma$  are the angles from step(3).
  - 13: **end for**
- 

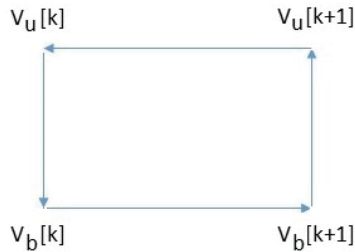
$$A(x, y, z) = \left( T \left( R(\alpha, \beta, \gamma) \left( T[k] \begin{pmatrix} x[k] - x[k-1] \\ y[k] - y[k-1] \\ z[k] - z[k-1] \end{pmatrix} \right) + \begin{pmatrix} x[k] \\ y[k] \\ z[k] \end{pmatrix} \right) \right). \quad (5.5)$$

The result of this algorithm is shown in Figure 5.4.

The second function is *draw()* in which the tool sketches the segments of the stitched image in a 3D way and for that purpose we use every time the coordinates of four points: two vertices of the bottom part of the segment  $V_b[k]$  and  $V_b[k+1]$  and two vertices of the upper part of the segment  $V_u[k+1]$  and  $V_u[k]$ , like it has been represented in Figure 5.6.

**Algorithm 2** Algorithm of *draw()* function

- 1: Load the stitched image.
- 2: Arrange the motion of the mouse.
- 3: **for** k equal:total number of video frames **do**
- 4:   Retrieve the elements of the lists from *setup()*.
- 5:   Begin the shape of the k segment and add texture to it.
- 6:   Connect the 4 coordinates for every segment calculated from the Equation 5.5 as it is shown in Figure 5.6 and then add a texture on it by using the same equation but with the angles values from step(4).
- 7:   Close the shape.
- 8: **end for**

Figure 5.6: How the *draw()* function draws the segment from the points coordinates

The result of this algorithm is shown in Figure 5.5.

## 5.7 Conclusion

This chapter described how we construct the 3D skin model in a theoretical and algorithmic way. To summarize, we divided a stitched image into segments corresponding to the video frames that we used for each one of them. Then we calculated the coordinates of this segment in the 3D space by inserting the angles from the gyroscope into the rotation matrices. The final result of this procedure was the 3D folding of the image in the shape of the scanned object. The next part of the project is map navigation in order to be able every time to localise the scanned area between two different videos. The procedure that we followed in order to achieve the map navigation will be discussed in the next chapter.

## Chapter 6

# Map Navigation

In this chapter we detail the various steps used for map navigation. During the usage, the system is able to localize the treated area by comparing the image acquire during the treatment to the map. Since the map was constructed by earlier recordings, our main purpose is to identify the part of a scanned body location in another sample recorded from the same area. Improving patch recognition methods is out of scope for this work. Thus we enable this functionality only to be able to test our proposed system for mapping and navigation. However, we did not deeply covered the navigation in this work, since navigation was used mainly for testing purposes.

### 6.1 Map Navigation

The procedure for the map navigation is similar to the map reconstruction. The aim is to locate a small patch of image on the constructed map. There are many other applications which use the navigation approach like GPS navigation and ultrasound surgery. Though, the skin navigation faces some difficulties because of the skin deformation. More specific, because the skin is deformed every time the device interacts with it, there is no possibility to achieve exactly the same information from different measurements in the same skin area. In order to overcome this difficulty and be able to find the same area between two stitched images, the same methods with map construction have to be used. These methods will be used in the following way:

- **Tracking:** in order to be able to track features in the stitched images, they should first filtered in a way that these features will be more sharp. For that reason a Laplacian of the Gaussian (LoG) feature detector can be used with focus on two parameters [4]:
  1. Contrast threshold: the lower the contrast threshold is, the more skin features can be detected.
  2. Noise filtering: a Gaussian smoothing can be used to eliminate the noises in the images. In that way the detected features will be from the skin morphology imprinted in the image.
- **Matching:** for this method a descriptor-matcher strategy has to be defined since the pairs of the stitched images that are gone to be compared will have a different optical angle.

### 6.2 Theoretical Background

The main idea of map navigation is to be able to match the part of a skin image with the corresponding skin part of another image. In order to succeed with that we will use two different procedures: the features detection and the image matching.



### 6.2.1 Feature Detector

Since we are going to use skin images for the map navigation the identification part is going to be challenging. For this reason we are using first a feature detection algorithm in order to draw on the image key descriptors which are going to help us for the matching part.

A main reason that we use keypoint features is for finding a sparse set of corresponding locations in different images, which is a prerequisite for computing a denser set of correspondences using stereo matching. There are two main approaches for finding feature points and their correspondences. The first approach is more suitable for images which were taken from a close distance of the object or for videos with a rapid motion. This method finds features in one image that with the use of a local search technique such as correlation or least squares, can be tracked in another similar image. The second approach is more suitable when a large amount of motion or appearance change is expected. This method detects features in all the images independently and then match features based on their local appearance. In our case we have stitched images created from videos taken from a close distance from the skin, so we are going to use the first approach for the image matching[33].

### 6.2.2 Matching

In order to do the image matching between the map and the patch we will use the cross correlation method. Cross-correlation is a technique to collect the displacement information of two similar images by comparing the similarities of the corresponding image signals. It is the simplest but robust method as a similarity measure, which is invariant to linear brightness and contrast variations. However, the drawbacks of using this method are the large amount of calculation that need a long running time for the computer and it is not suitable for large skin deformations.

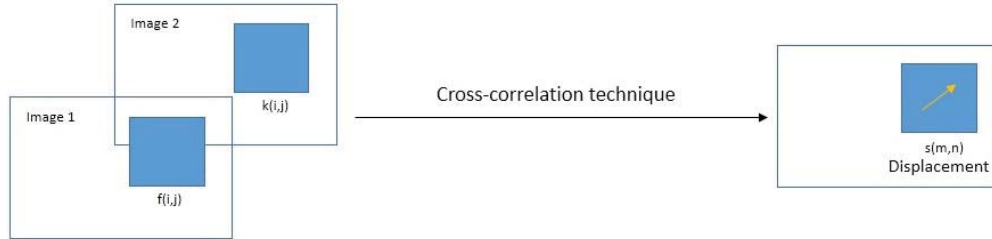


Figure 6.1: Cross-correlation method for image matching

The traditional cross-correlation method can be seen in Figure 6.1 and the cross-correlation coefficient is defined in the following formula

$$C = \frac{\sum_{i=1}^M \sum_{j=1}^N I(i, j) \times J(i + m, j + n)}{\sqrt{\sum_{i=1}^M \sum_{j=1}^N I(i, j)^2} \sqrt{\sum_{i=1}^M \sum_{j=1}^N (J(i + m, j + n))^2}} \quad (6.1)$$

Where  $I(i, j)$  and  $J(i, j)$  are the gray value of the template window and the inquiring window, both of the windows with a size  $M \times N$ . The aimed area is the random vector which has the maximum cross-correlation coefficient [34].

### 6.3 Implementation of Map Navigation

As we mentioned already, map navigation uses the same methods with map construction. We implemented this part using the programming language Python in combination with the library OpenCV [24]. In order to be able to compare a small patch of an image with another image used several tracking algorithms in order to find the one which gives the best results. The algorithms that we tried are SIFT [25], SURF [23] and FAST [27] and based on number of features comparison presented in Table 6.1 we concluded that the most suitable one for our case is FAST. In order to create strong key-features to be able to identify with the matching algorithm, we drew black circles on the points that the FAST algorithm detected. The result of this step can be seen on Figure 6.2.

Table 6.1: Number of features detected from each algorithm

Feature Detector	# of features
SIFT	110
SURF	455
FAST	1004

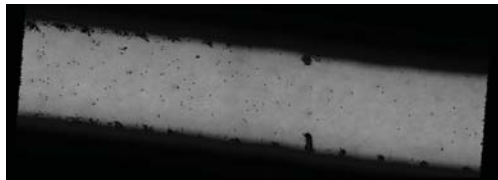


Figure 6.2: Image after the use of FAST detector

For the navigation part we need to use a matching algorithm. As an illustration we will use Figure 6.3 where (a) is the region to search, (b) is the patch and (c) is the representation of the cross correlation matrix which we used as a method for map navigation. For this purpose we used the Matlab implementation of the Normalized 2D cross-correlation [33]. This function gives as an output the x and y coordinates of the best matching position of the bottom-right pixel of the patch on the image that we used as a map. Then, in order to find the corresponding area on the image that we used as a map we subtract from these values the width and height values of the patch. In order to be able to visualize the matching area we used a red rectangle for representing the position as a result from the cross-correlation function, and a green rectangle that shows the ground truth location of the patch, like it is shown in Figure 6.3(a).

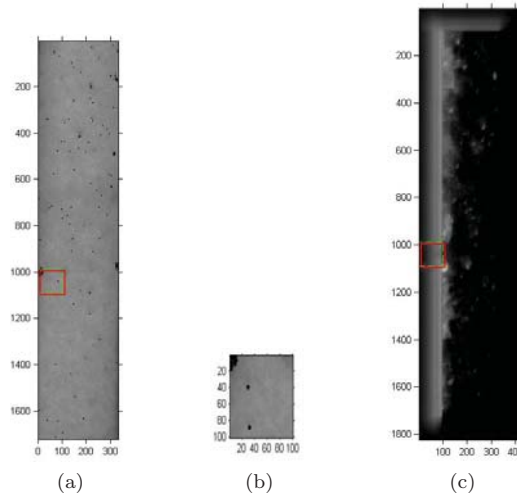


Figure 6.3: Cross-correlation result (a)Result from map navigation(Green rectangle: ground truth position, Red rectangle: position found from cross-correlation function, (b)Patch that we used for the image matching, (c)Cross-correlation matrix representation

### 6.3.1 Algorithmic Implementation of the map navigation.

---

#### Algorithm 3 Algorithm Implementation for Map Navigation

---

- 1: Read the image that will be used as a map.
  - 2: Read the image that will be used as a patch.
  - 3: Rotate both images in order to be aligned.
  - 4: Rotate again both images in the same direction in order to be vertical.
  - 5: Crop both images in order to not have a black frame around the image.
  - 6: Specify the area of the patch image that you want to identify into the map image.
  - 7: Make a normalized 2D cross-correlation between the image in step(6) and the map image.
  - 8: Find the position of the maximum value of cross-correlation function and its position.
  - 9: Find the corresponding coordinates of the position from step(8) on the map image.
  - 10: Draw a rectangle on the map with the real matching position of the patch and another rectangle with the position found from the cross-correlation function (like in Figure 6.3).
  - 11: Derive the distance between real and found position.
- 

## 6.4 Conclusion

We implemented map navigation functionality in order to investigate the localization power in our reconstructed map. We therefore used a patch in order to get a small frame and search in the map image. We use correlation technique to do the image matching. In the next chapter we are going to use the same algorithm of map navigation with different kind of images in order to define the future system specification.

## Chapter 7

# System specifications identification

One of the possible uses of map navigation technique is on an embedded platform of a consumer device. For this reason we would like to reduce the size of the map image in order to faster transferring them from the cloud to the device. In this chapter we aim at finding different ways of image representation and identifying possible combinations of various functional parameters enabling localization and navigation after body scanning. We are trying to keep the same results of map navigation and reduce the size of the image and we are going to make a Design of Experiment to show the various results.

### 7.1 Measured system and evaluation

In order to find the most suitable parameters for the map navigation technique we are going to make a Design of Experiment. DoE (design of experiments) is the design of controlled experiments of any information-gathering exercises where variation is present. This technique helps the investigation of the effects of input variables (factors) on an output variable (response) at the same time. It consists a series of runs, or tests, in which purposeful changes are made to the input variables and data are collected at each run. The end result of a DOE is to identify the process conditions and product components that affect quality, and then determine the factor settings that optimize results. [36] More specifically, DoE techniques enable designers to determine simultaneously the individual and interactive effects of many factors that could affect the output results in any design. DoE also provides a full insight of interaction between design elements; therefore, it helps turn any standard design into a robust one.

A designed experiment contains four different phases. The first one is the planning phase and it is very important because it gives significant information about the experiment and it can ensure a good evaluation of the effects that have been identified as important. During this phase a good problem statement should be developed which will ensure that the variables that are examined are correct. Also, there should be a well-defined goal, which ensures that the experiment answers the correct questions and it yields practical and useful information. Another important step is that the development of the experimental plan should provide meaningful information and for this reason it should be relevant to the background information of the project, like theoretical information and knowledge obtained through observation or previous experimentation.

The next step is the screening phase, which reduces the number of factors by identifying the key factors or process conditions that affect the quality of the result. This reduction helps to concentrate process improvement efforts on the few really important variable, or the vital few. Screening can identify the optimal settings for these factors, and show whether or not curvature exists in the responses. Optimization experiments can then be done to determine the best settings and define the nature of the curvature.

Another important stage of the DoE is the optimization phase. After the identification of the vital few by screening, there is a need of determining the optimal values of the experimental factors. Optimal factor values depend on the process goal, e.g. in order to maximize process yield or reduce product variability.

### 7.1.1 The DoE tool

The tool that we are going to use in order to implement the DOE is MINITAB[35]. MINITAB is a statistical tool and offers four types of designed experiments: factorial, response surface, mixture, and Taguchi (robust). The steps that should be followed in MINITAB to create, analyze, and graph an experimental design are similar for all design types. After you conduct the experiment and enter the results, MINITAB provides several analytical and graphing tools to help you understand the results. In our experiment we are going to use the factorial type in order to execute the DoE plots.

### 7.1.2 DoE plots

Before starting the analysis of our experiment's results we are going to explain the kind of plots that we will use. In order to be more clear we will use a simple example with the data that are shown in Table 7.1.

Table 7.1: Data used in the example DoE

In-A	In-B	In-C	In-D	In-E	In-F	Out-A	Out-B
180	50	0	0	10	10	142	869
250	50	0	0	100	10	490	1493
180	100	0	0	100	100	413	1042
250	100	0	0	10	100	213	1327
180	50	1	0	100	100	753	1099
250	50	1	0	10	100	299	1382
180	100	1	0	10	10	175	853
250	100	1	0	100	10	429	1489
180	50	0	1	10	100	535	1090
250	50	0	1	100	100	552	1558
180	100	0	1	100	10	381	1075
250	100	0	1	10	10	249	1126
180	50	1	1	100	10	552	1085
250	50	1	1	10	10	262	1151
180	100	1	1	10	100	527	1103
250	100	1	1	100	100	622	1680

- Main effects plots

A main effects plot is used to examine differences between level means for one or more factors. A main effect is the effect of an independent variable (factor) on a dependent one averaging across the levels of any other independent variables. A main effect test is non-specific and it only can find if there is evidence of an effect of different treatments. In other words, main effects are essentially the overall effect of a factor.

A main effects plot graphs the response mean for each factor level connected by a line. The graph in Figure 7.1 uses the example data from Table 7.1 and it shows the effect of using value 10 versus value 100 for process In-E, or using In-F procedure with again value 10 versus value 100.

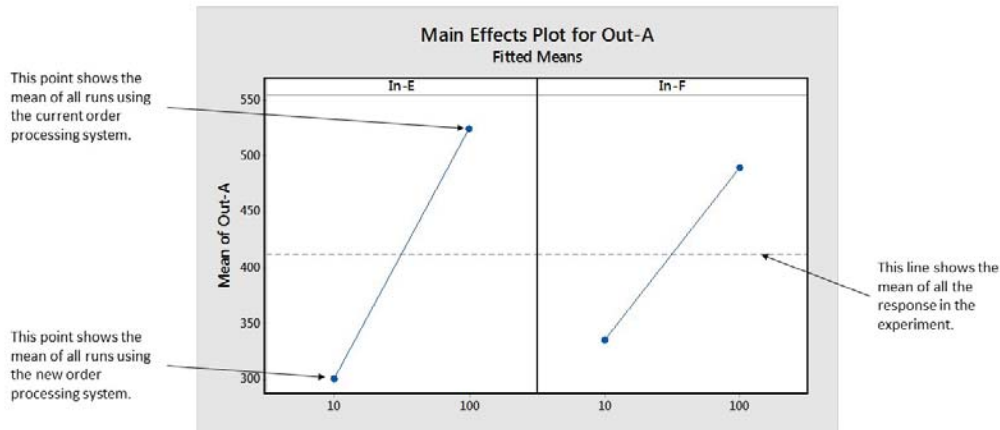


Figure 7.1: Main effects example plot

By analysing this graph we can see that the inputs In-E and In-F have a similar effect on order Out-A since the line connecting the mean responses for value 10 and 100 in order In-E has a slope similar to slope of the line connecting the mean response for values 10 and 100 in order In-F.

The reference line represents the overall mean and the general patterns we should look for are the following:

- When the line is horizontal (parallel to the x-axis), then there is no main effect. Each level of the factor affects the response in the same way, and the response mean is the same across all factor levels.
- When the line is not horizontal, then there is a main effect. Different levels of the factor affect the response differently. The steeper the slope of the line, the greater the magnitude of the main effect.

Main effects plots can not show interactions between factors, for that purpose we are going to use an interaction plot.

- Interaction plots

An interaction describes a situation in which the simultaneous influence of two variables on a third is not additive. The presence of interactions can have important implications, i.e. if two variables interact, the relationship between each of them and a third "dependent variable" depends on the value of the other interacting variable. This situation makes it more difficult to predict the consequences of changing the value of a variable, especially if the variables it interacts with are hard to measure or difficult to control. An interaction can be found in a factorial design, in which the two (or more) factors are "crossed" with one another so that there are observations at every combination of levels of the two independent variables.

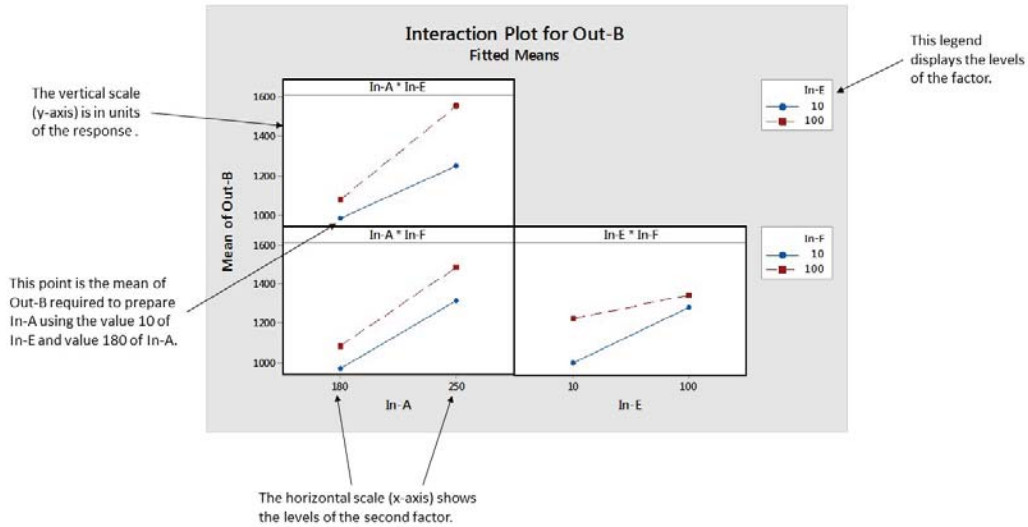


Figure 7.2: Interaction example plot

An interaction plot shows the impact that changing the settings of one factor has on another factor. The graph in Figure 7.2 uses the example data from Table 7.1 and it indicates an interaction between In-A, In-E and In-F. The nearly parallel lines especially for values 10 and 100 in In-F on the bottom left chart in Figure 7.2 indicate only a very weak interaction between the two levels of factor In-A. This plot displays data means, while you can use raw data to obtain a general idea of which effects may be evident. However, in case we want to find the most optimal combinations of the different factors, we have to use D-optimal plots.

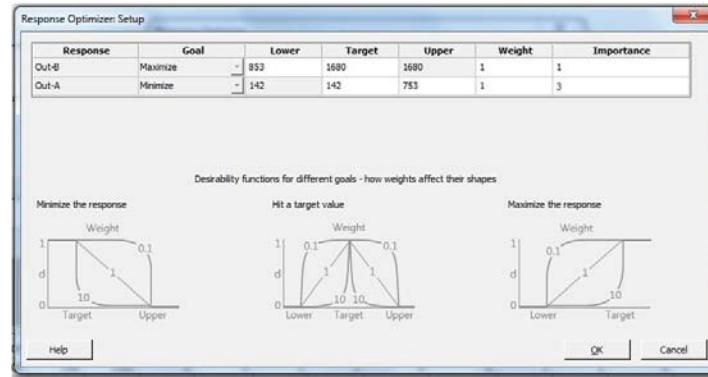
- D-optimal plots

An optimal design is a group of selected point with the best needed values when reducing or augmenting the number of experimental runs in the original design. D-optimality is used to determine an optimal design for factorial, response surface or mixture experiments, based on straight optimizations on a chosen optimality criterion and the model that will be fit. The optimality criterion result of a given D-optimal design is model dependent. The computer algorithm chooses the optimal set of design runs from a candidate set of possible design runs for an experiment and a specified model. This candidate set consists all possible combinations of various factor levels used in the experiment from which the D-optimal algorithm chooses the best combinations to include in the design.

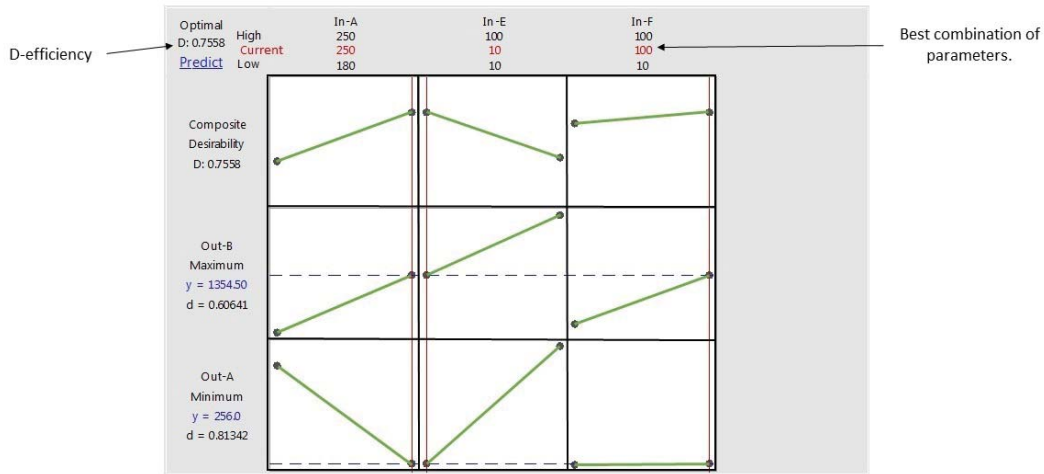
D-optimal designs are used instead of standard classical designs under two circumstances:

1. standard factorial or fractional factorial designs require too many runs for the amount of resources or time allowed for the experiment.
2. the design space is constrained (the process space contains factor settings that are not feasible or are impossible to run).

In D-optimal designs the purpose is to maximize the D-efficiency, which is a volume criterion on the generalized variance of the parameter estimates. The ideal D-efficiency of the standard fractional factorial is 100%. The D-efficiency values are a function of the number of points in the design, the number of independent variables in the model, and the maximum standard error for prediction over the design points. The best design is the one with the highest D-efficiency.



(a) Response optimizer setup



(b) Plot

Figure 7.3: D-optimal example plot

In optimal design there is the possibility to specify the boundaries, weight and importance for each response variable as it is shown on the Table 7.2. In the case of our example as it can be seen in Figure 7.3(a) we maximized Out-B, minimized Out-A and we add an importance equal to 3. In the plot of Figure 7.3(b) we can see that the most optimal combination of the chosen parameters is when In-A = 250, In-E = 10 and In-F = 100 with D-efficiency = 0.7558. In this combination of parameters we have as an output the minimum possible Out-A=256.0 and the maximum Out-B = 1354.50.

In Chapter 7.3 we are going to use the DoE plots in order to analyse the results of our experiment using different parameters as an input.



Table 7.2: Setup options for optimizer

<b>Response</b>	Displays all of the responses that are included in the optimization. This column does not take any input
<b>Goal</b>	Displays the goal for each response. This column does not take any input.
<b>Lower</b>	For each response that has a goal of Target or Maximize, enter a lower boundary.
<b>Target</b>	Enter a target value for each response.
<b>Upper</b>	For each response that has a goal of Minimize or Target under Goal, enter an upper boundary.
<b>Weight</b>	Enter a number from 0.1 to 10 to define the shape of the desirability function.
<b>Importance</b>	Enter a number from 0.1 to 10 to specify the comparative importance of the response.

## 7.2 Implementation of our DoE

In this part of the project in order to create the DoE, we must first create an experimental design and store it in a worksheet. In order to calculate these measurement data we had to analyse the requirements of our design and find out which are the most suitable parameters that we are going to use as an input in the DoE. The following Section contains the parameters that we used and then we present the plotting graphs created by the measurement data.

### 7.2.1 Parameters

In the DOE that we made the end-result is to find the best combination of different parameters of the images representation in order to have the best matching results. This means that the area that the cross correlation finds on the map image to be the closest possible to the ideal one. The different parameters that we examined are the following ones:

- Different versions of representation of the image:
  - the original one, Figure 7.4(a)
  - the original with black circles on the features that were detected in Chapter 6, Figure 7.4(b)
  - white background in same size of the image with black circles on the coordinates that were detected in Chapter 6, Figure 7.4(c)
  - black background in same size of the image with white circles on the coordinates that were detected in Chapter 6, Figure 7.4(d)
- Four different sizes of radius in features of the image, Figure 7.5.
- Two different contrasts of the original image, one 1% lighter contrast than the original Figure 7.6(a) and one with 1% darker contrast than the original Figure 7.6(c).
- Different sizes of the searching area in vertical and horizontal direction from 50 until 330 pixels length.
- Three different location of the patch with several densities.

After defining all the above parameters, we modified the map navigation script, by adding loops which were implementing all the combinations and it gives as an output an excel file which contains the results of image matching by calculating how far the position of the calculated match is from the reality.

From the DoE data we built a model which uses the above parameter as an input and the difference between the ideal and the calculated position of the patch as an output. The following Section presents the plotting results of this model.

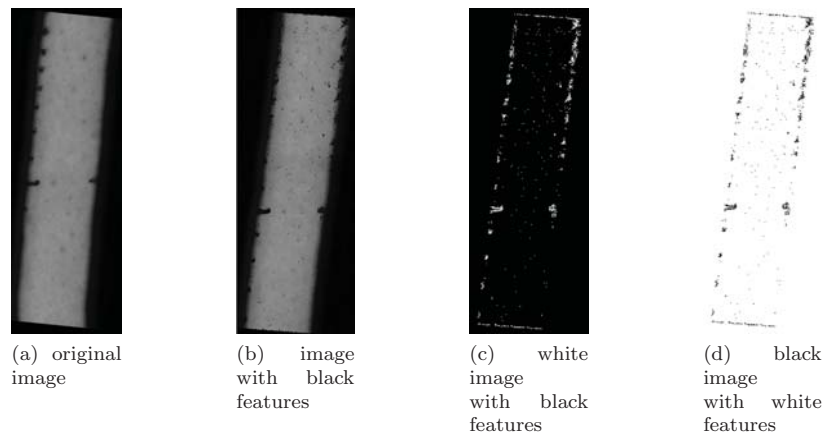


Figure 7.4: Different versions of image representation

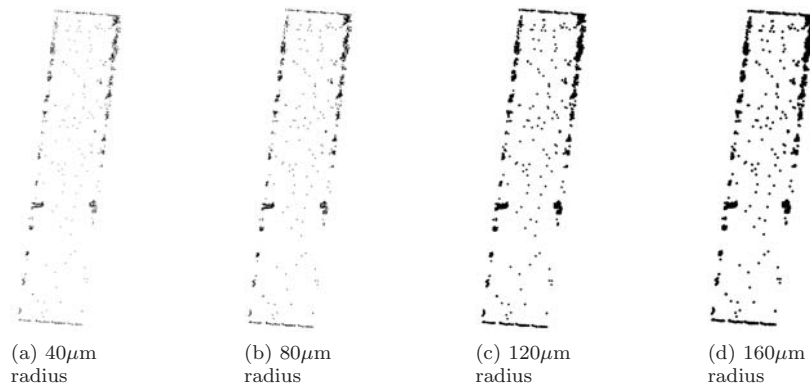


Figure 7.5: Samples of the different radius representation

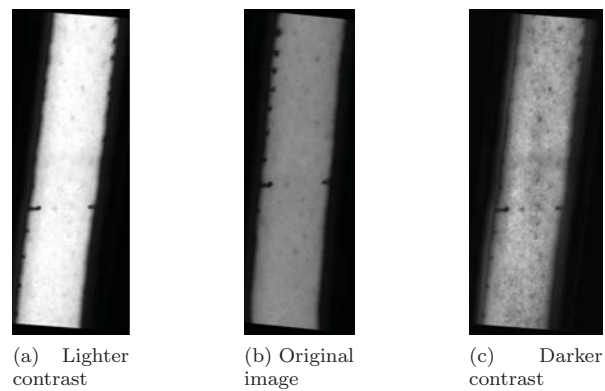


Figure 7.6: Different contrasts of the original image

## 7.2.2 Results

After running the script with the different parameters that we already mentioned, we inserted the values into the MINITAB in order to implement the DoE. The following lists contain the different factors that we used for each parameter and what these factors represent:

- Image Color:
  - 2: original-grayscale image, Figure 7.4(a)
  - -2: original-grayscale image with black features, Figure 7.4(b)
  - 1: white background with black features, Figure 7.4(c)
  - -1: black background with white features, Figure 7.4(d)
- Image Contrast:
  - 1: Image with lighter contrast than the original, Figure 7.6(a)
  - 0: Original image, Figure 7.6(b)
  - -1: Image with darker contrast than the original, Figure 7.6(c)
- dimX: different dimensions on x-direction
  - 50: 2mm length
  - 100: 4mm length
  - 150: 6mm length
  - 191: 7.64mm length
  - 273: 10.92mm length
  - 329: 13.16mm length
- dimY: different dimensions on y-direction
  - 50: 2mm length
  - 100: 4mm length
  - 150: 6mm length
  - 191: 7.64mm length
  - 273: 10.92mm length
  - 329: 13.16mm length
- Radius:
  - 0: no feature detection
  - 1: each feature has 2 pixels diameter, Figure 7.5(a)
  - 2: each feature has 4 pixels diameter, Figure 7.6(b)
  - 3: each feature has 6 pixels diameter, Figure 7.6(c)
  - 4: each feature has 8 pixels diameter, Figure 7.6(d)

During this experiment we want to find how the different parameters effect the difference between the ideal location of the patch location on the map and the location calculated from our algorithm. The DoE plots that created with MINITAB are the ones below in Figures 7.7 - 7.12 and we are going to analyse this results thoroughly.

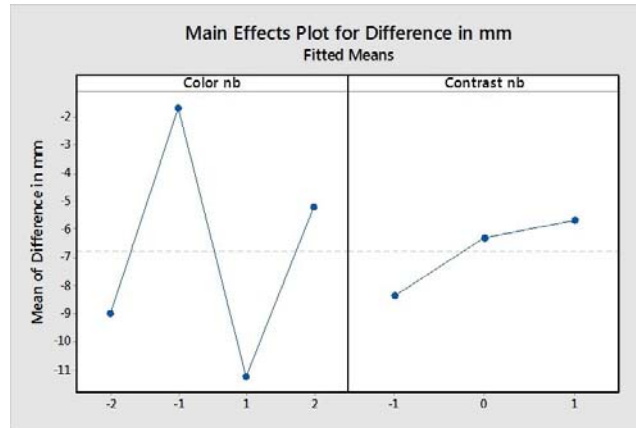


Figure 7.7: Main effects example plot

- Main effects plot

The graph in Figure 7.7 shows the main effects of different levels on the color and contrast factors of an image on the distance difference in millimetres. By analysing this graph we can see that each level of every factor affects the output differently. The basic conclusion that we may get from this plot is that in general the most good results are the ones that use the images with black background and white features. Also, the contrast has a small affection on the output with the best results when the image has a lighter contrast than the original.

- Interaction plot

The graph in Figure 7.8 shows the interactions between color of the image, contrast of the image and dimensions of patch on x-direction. We can notice that there are complex interactions between these three parameters and the output values are ranging between -15mm to 15mm. Also, since the other parameters were not included into the graphs then there is no interaction with them.

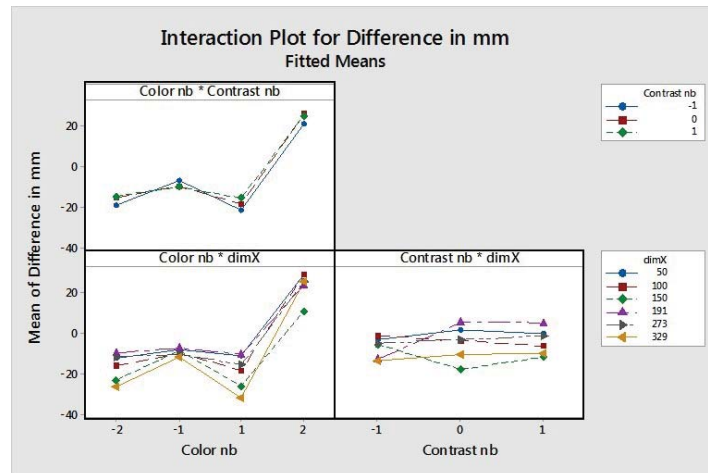
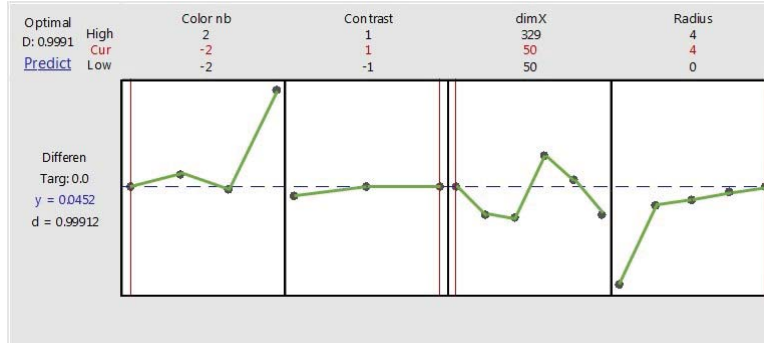


Figure 7.8: Interaction plot

- D-optimal plot

The plots in Figures 7.9 - Figure 7.12 show the most optimal combinations of parameters in order to succeed the closest difference to the ideal position. Since the value of D-efficiency equals to 0.9 in all of the d-optimal designs then the parameters estimation is almost ideal. The tables below these figures present the details of these combinations.

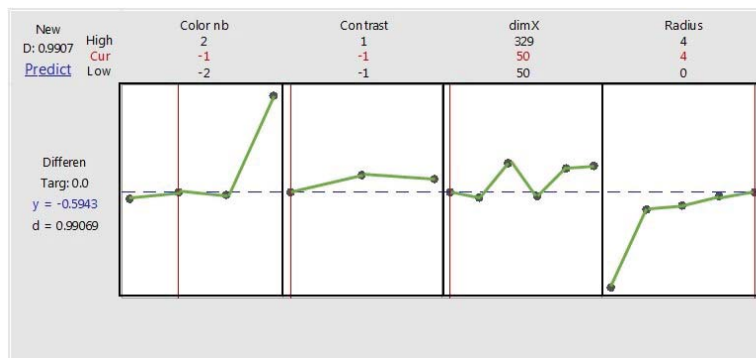


(a)

Figure 7.9: 1st D-optimal plot

Table 7.3: 1st Optimal combination of parameters

Image Color	Image Contrast	X-dimensions	Y-dimensions	Radius	Difference	D-efficiency
-2	1	50	50, 100, 150	4	0.0452	0.9991

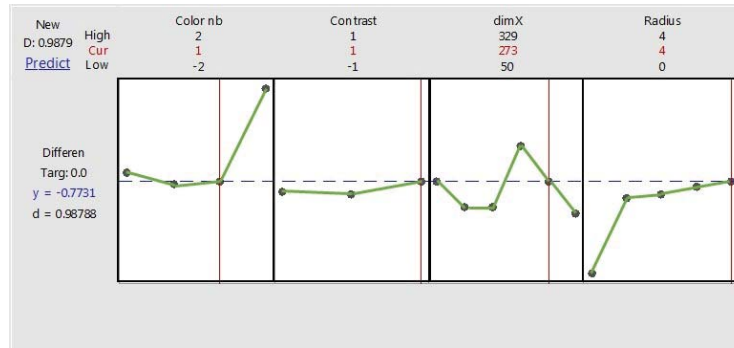


(a)

Figure 7.10: 2nd D-optimal plot

Table 7.4: 2nd Optimal combination of parameters

Image Color	Image Contrast	X-dimensions	Y-dimensions	Radius	Difference	D-efficiency
-1	-1	50	100, 150	4	-0.5943	0.9907

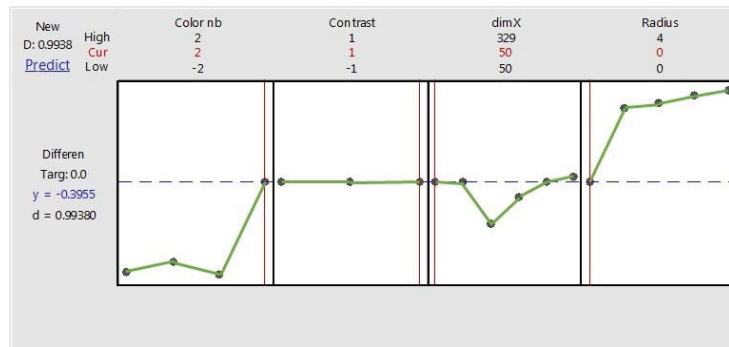


(a)

Figure 7.11: 3rd D-optimal plot

Table 7.5: 3rd Optimal combination of parameters

Image Color	Image Contrast	X-dimensions	Y-dimensions	Radius	Difference	D-efficiency
1	1	273	191, 273, 329	4	-0.7731	0.9879



(a)

Figure 7.12: 4th D-optimal plot

Table 7.6: 4th Optimal combination of parameters

Image Color	Image Contrast	X-dimensions	Y-dimensions	Radius	Difference	D-efficiency
2	1	50	50, 100, 150	0	-0.3955	0.9938

The above experiment helped us to search in depth all the combinations of different parameters for different ways of image representation in order to keep a very close distance between the cross-correlation window location and the calculated location. As it is shown in Table 7.3, in these combinations there is also the feature based representation with a difference distance of -0.5943 mm when we have black background and white features. That means that this representation is sufficient for the map navigation procedure and in this way we are able to reduce the image

footprint 150 times from the one of the original image and sequentially that reduces the processing time. In order to be able to stream the xy-coordinates of feature points we could use a cheap platform to load this information.

### 7.3 Conclusion

This chapter described the way we implemented a DoE using different parameters as an input for the experiment and finding as an output the closest possible distance to the real location of the patch on the map image. In that way we estimated the best combinations of parameters to succeed the closest possible difference to the ideal location of the patch on the map image.

## Chapter 8

# Conclusions

In this master thesis we propose a solution for "3D skin body reconstruction and navigation". Our solution consists of a scanning device that records videos from the skin and the software functionalities to enable the 3D body reconstruction and navigation. In addition to the recorded videos, our system collects and processes data from a gyroscope that, together with the videos, enable the 3D reconstruction.

To achieve this, we needed a solution which is suitable for the requirements of a consumer device. On the hardware side, a 3D scanning device was constructed with the use of a smartphone Samsung Galaxy s5 and the adjustment of a dermoscope device on its camera for better resolution skin images. This device has also an embedded gyroscope sensor that will deliver real time data to help with the reconstruction of the 3D skin model.

On the software side, this report contains image processing routines that will help us to achieve 3D image reconstruction and navigation. 3D image reconstruction required a stitched image as an input, for which we used a ready in-house algorithm for stitching images. The algorithm has as an input a video, synchronized with the gyroscopic data, in such a way that each part of the stitched image corresponds to an angle orientation.

The main part of this project is the 3D reconstruction of the stitched image. The procedure of the image 3D folding is divided into two parts: the grid and the texturing. In the beginning we are making a 3D grid in the shape and size of the 2D stitched image. We are separating this shape into segments same with the sizes of the parts of the video frames that construct the stitched image. The next step is to fold these segments into the right angles. For that purpose we are inserting the roll, pitch and yaw angle from the gyroscope into the rotation matrices in order to calculate the right coordinates of the grid in the 3D space. Finally, we are introducing the texture of the 2D stitched image into the 3D grid. After all these steps we have as a result the 3D reconstruction of the scanned body area.

Another part of this project is the skin navigation. The purpose of this part is to be able to identify the location of a part of an image in another image taken from the same body part in an earlier acquisition that enabled the map construction. In order to succeed the navigation we followed the cross-correlation technique.

The last aim of this report is to find different ways of the representation of the stitched image in order to reduce its size while we are able to have the same localization results during map navigation. Only by solving this we can enable our solution implementation in a consumer product. For optimizing the system parameters, we used a Design of Experiment (DoE) with different versions of the same image. The versions that we used were with different contrasts and with only the coordinated of different features on a blank image. The result of this experiments was to enable a successful identification of the wanted location in an image with a size 150 times smaller than the original one.

This project was able to prove that it is possible to overstep a major problem that cosmetic home-devices are facing. In the future the customer will be able to see the representation of his/her body in a digital form and know which area of the body were treated or not.



# Bibliography

- [1] Personal care devices: is it worth crossing the medical chasm? 1
- [2] <http://www.lorealparisusa.com/en/brands/makeup/makeup-genius-virtual-makeup-tool.aspx> 2
- [3] <http://www.philips.com/e/groomingapp/> 2
- [4] Chen, Y. "Face Navigation with On-board Sensors for Shaving" 16, 17, 18, 26
- [5] Regis, M. "Predictive Analysis for Selective Photothermolysis: Modeling, Simulation and Analysis" 6
- [6] Proksch, Ehrhardt, Johanna M. Brandner, and JensMichael Jensen. "The skin: an indispensable barrier." *Experimental dermatology* 17.12 (2008): 1063-1072. 6
- [7] Igarashi, Takanori, Ko Nishino, and Shree K. Nayar. "The appearance of human skin." (2005). 6, 7
- [8] Wagner, Heike, et al. "Human skin and skin equivalents to study dermal penetration and permeation." *Cell culture models of biological barriers: in vitro test systems for drug absorption and delivery* (2002): 289. 6
- [9] <http://www.webmd.com/skin-problems-and-treatments/picture-of-the-skin> 6
- [10] <http://www.samsung.com/global/microsite/galaxy5/specs.html> 9
- [11] <https://play.google.com/store/apps/details?id=com.cellbots.logger&hl=en> 12
- [12] Rosten, Edward, and Tom Drummond. "Machine learning for high-speed corner detection." *Computer VisionECCV 2006*. Springer Berlin Heidelberg, 2006. 430-443. 16
- [13] Rublee, Ethan, et al. "ORB: an efficient alternative to SIFT or SURF." *Computer Vision (ICCV), 2011 IEEE International Conference on*. pp. 2564-2571. IEEE, 2011. 17
- [14] Lucas, Bruce D., and Takeo Kanade. "An iterative image registration technique with an application to stereo vision." *IJCAI*. Vol. 81. pp. 674-679. 1981. 17
- [15] <http://code.google.com/p/cellbots/source/browse/trunk/android/java/dataLogger/> 13
- [16] <http://developer.android.com/reference/android/hardware/SensorEvent.html> 12, 13
- [17] <http://planning.cs.uiuc.edu/node102.html> 20
- [18] <http://dermlite.com/products/dermlite-dl3>
- [19] Mullani, Nizar A. "Dermoscopy epiluminescence device employing cross and parallel polarization." U.S. Patent No. 7,006,223. 28 Feb. 2006.
- [20] <http://en.wikipedia.org/wiki/OpenGL> 22

- [21] <https://processing.org/> 22
- [22] Hyung Gi Min, Eun Tae Jeung "Complementary Filter Design for Angle Estimation using MEMS Accelerometer and Gyroscope"
- [23] Huijuan, Zhang, and Hu Qiong. "Fast image matching based-on improved SURF algorithm." Electronics, Communications and Control (ICECC), 2011 International Conference on. IEEE, 2011. 28
- [24] <http://opencv.org/> 28
- [25] Markel, John. "The SIFT algorithm for fundamental frequency estimation." Audio and Electroacoustics, IEEE Transactions on 20.5 (1972): 367-377. 28
- [26] C. Tomasi and T. Kanade (2004). "Detection and Tracking of Point Features". Pattern Recognition 37: 165168
- [27] M. Trajkovic and M. Hedley (1998). "Fast corner detection". Image and Vision Computing 16 (2): 7587. 28
- [28] [http://docs.opencv.org/trunk/doc/py\\_tutorials/py\\_feature2d/py\\_matcher/py\\_matcher.html](http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_matcher/py_matcher.html)
- [29] <http://planning.cs.uiuc.edu/node101.html> 12
- [30] <http://encyclopedia.jrank.org/articles/pages/6790/Mesh-3D.html> 20
- [31] [https://facultypages.scad.edu/gjohnson/web\\_reference/modeling\\_examples/\\_topoconstruction07c.jpg](https://facultypages.scad.edu/gjohnson/web_reference/modeling_examples/_topoconstruction07c.jpg) 21
- [32] Reas, Casey, Ben Fry, and J. Maeda. "Processing. A Programming Handbook for Visual Designers." (2007). 23
- [33] Szeliski, Richard. "Feature detection and matching." Computer Vision. Springer London, 2011. 181-234. 27, 28
- [34] Dou, Jian-fang, and Jian-xun Li. "Robust image matching based on wavelet transform and SIFT." 3rd International Conference on Digital Image Processing. International Society for Optics and Photonics, 2011. 27
- [35] <http://www.minitab.com/en-us/> 31
- [36] <http://support.minitab.com/en-us/minitab/17/getting-started/designing-an-experiment/> 30

## Appendix A

# 3D Reconstruction Code

This appendix contains the Python code that we created for the 3D body representation that we are explaining in details in Chapter 5.

```
// import library , (install a Library from menu:
//” Sketch -> Import Library... -> Add Library... ->
//Install” here the Library is XlsReader
import de.bezier.data.*;
// import library for ArrayList
import java.util.ArrayList;

// Enable the XlsReader for the three excel files
XlsReader reader;
XlsReader reader_gyro;
XlsReader reader_2D;

float rotx = PI/3;
float roty = PI/3;

//number of frames
int NOF = #;

String name = "name_of_stitched_image";

//create the lists to store the data from stitching
float [] frame = new float [NOF];
float [] x_s = new float [NOF];
float [] y_s = new float [NOF];
float [] dx = new float [NOF];
float [] dy = new float [NOF];

//create the lists to store the data from gyroscope
float [] roll = new float [NOF];
float [] pitch = new float [NOF];
float [] yaw = new float [NOF];

//create the lists to store the calculated roll , pitch , yaw
float [] a = new float [NOF];
float [] b = new float [NOF];
float [] c = new float [NOF];

//create two lists for storing the xyz values
ArrayList<float [][]> list_b = new ArrayList<float [][]> ();
ArrayList<float [][]> list_u = new ArrayList<float [][]> ();

//create two lists for texturing
//(one for the bottom and one for the upper part of every tile)
ArrayList<float [][]> list_b_text = new ArrayList<float [][]> ();
ArrayList<float [][]> list_u_text = new ArrayList<float [][]> ();
ArrayList info = new ArrayList ();
```

```

//enable the image
PImage img;

void setup() {

//size of displaying window and use of P3D library
//(P3D makes use of OpenGL-compatible graphics hardware)
  size(1000, 1000, P3D);
  matrices();
}

void matrices(){

//excel created from the stitching algorithm
//(first I have to save it from .csv into .xls format)
  reader = new XlsReader( this, name + ".xls" );

//excel created with python by processing the video's gyroscope values
  reader_gyro = new XlsReader( this, "gyro_angles.xls" );

//excel file with same size of the gyro_angles.xls
//but with all the values equal to 0
  reader_2D = new XlsReader( this, "gyro_angles_2D.xls" );

  int m = reader.getLastRowNum();

// go to page 1 (at address 0) of each excel file
  reader.openSheet(0);
  reader_gyro.openSheet(0);
  reader_2D.openSheet(0);

  for (int k = 1; k < NOF; k++) {

//store the values from the excel files into 1-d matrices
  frame[k] = reader.getFloat(k,0) ;
  x_s[k] = reader.getFloat(k,1) ;
  y_s[k] = reader.getFloat(k,2) ;
  dx[k] = reader.getFloat(k,3) ;
  dy[k] = reader.getFloat(k,4) ;

  roll[k] = reader_gyro.getFloat(k,0) ;
  pitch[k] = reader_gyro.getFloat(k,1) ;
  yaw[k] = reader_gyro.getFloat(k,2) ;

  c2[k] = reader_2D.getFloat(k,0) ;
  b2[k] = reader_2D.getFloat(k,1) ;
  a2[k] = reader_2D.getFloat(k,2) ;

//roll pitch yaw without integration
  a[k] = yaw[k];
  b[k] = pitch[k];
  c[k] = roll[k];

  }

//Initialize the first values of the matrices

  float [][] xy_points_init_b = new float [][] {
  {0, 0},
  {0, 0}};

  float [][] xy_points_init_angle_b = new float [][] {
  {0, 0},
  {283, 283},

```

```

{0, 0}};

list_b.add(xy_points_init_angle_b);
list_b_text.add(xy_points_init_angle_b);

float [][] xy_points_init_u = new float [][] {
{0, 0},
{0, 0}};

float [][] xy_points_init_angle_u = new float [][] {
{0, 0},
{700, 700},
{0, 0}};

list_u.add(xy_points_init_angle_u);
list_u_text.add(xy_points_init_angle_u);

for (int i = 2; i < NOF; i++) {

    float [][] xy_points_pr_b = new float [3][4];
    xy_points_pr_b = list_b.get(i-2);

    float [][] xy_points_pr_2D_b = new float [3][4];
    xy_points_pr_2D_b = list_b_text.get(i-2);

    float [][] xy_points_b_text = new float [3][4];
    xy_points_b_text = list_b_text.get(i-2);

    float [][] xy_points_u_text = new float [3][4];
    xy_points_u_text = list_u_text.get(i-2);

    float [][] xy_points_b = new float [][] {
    {(y_s[i] - y_s[1]), ((y_s[i] - dy[i]) - y_s[1])},
    {x_s[i], x_s[i] + dx[i]},
    {0, 0}};

//calculate the rotation matrices after inserting the roll ,pitch and yaw angles
//then store the calculated values into lists

    float [][] xy_points_angle_b = new float [][] {
    {xy_points_pr_b[0][1], xy_points_pr_b[0][1] + (xy_points_b[0][1] - xy_points_b
    [0][0]) * cos(a[i]) * cos(b[i])
    + (xy_points_b[1][1] - xy_points_b[1][0]) * ((cos(a[i]) * sin(b[i]) * sin(c[i]))
    - (sin(a[i]) * cos(c[i])))
    + (xy_points_b[2][1] - xy_points_b[2][0]) * ((cos(a[i]) * sin(b[i]) * cos(c[i]))
    + (sin(a[i]) * sin(c[i])))},

    {xy_points_pr_b[1][1], xy_points_pr_b[1][1] + (xy_points_b[0][1] - xy_points_b
    [0][0]) * sin(a[i]) * cos(b[i])
    + (xy_points_b[1][1] - xy_points_b[1][0]) * ((sin(a[i]) * sin(b[i]) * sin(c[i]))
    + (cos(a[i]) * cos(c[i])))
    + (xy_points_b[2][1] - xy_points_b[2][0]) * ((sin(a[i]) * sin(b[i]) * cos(c[i]))
    - (cos(a[i]) * sin(c[i])))},

    {xy_points_pr_b[2][1], xy_points_pr_b[2][1] + ((xy_points_b[0][1] - xy_points_b
    [0][0]) * (-sin(b[i])))
    + ((xy_points_b[1][1] - xy_points_b[1][0]) * cos(b[i]) * sin(c[i]))
    + ((xy_points_b[2][1] - xy_points_b[2][0]) * cos(b[i]) * cos(c[i]))}};

    float [][] xy_points_2D_b = new float [][] {
    {xy_points_pr_2D_b[0][1], xy_points_pr_2D_b[0][1] + (xy_points_b[0][1] -
    xy_points_b[0][0]) * cos(a2[i]) * cos(b2[i]) +
    (xy_points_b[1][1] - xy_points_b[1][0]) * ((cos(a2[i]) * sin(b2[i]) * sin(c2[i])
    ) - (sin(a2[i]) * cos(c2[i]))) +

```

```

    (xy_points_b[2][1] - xy_points_b[2][0]) * ((cos(a2[i]) * sin(b2[i]) * cos(c2[i]))
    ) + (sin(a2[i]) * sin(c2[i]))),

    {xy_points_pr_2D_b[1][1], xy_points_pr_2D_b[1][1] + (xy_points_b[0][1] -
    xy_points_b[0][0]) * sin(a2[i]) * cos(b2[i])
    + (xy_points_b[1][1] - xy_points_b[1][0]) * ((sin(a2[i]) * sin(b2[i]) * sin(c2[i]
    )) + (cos(a2[i]) * cos(c2[i])))
    + (xy_points_b[2][1] - xy_points_b[2][0]) * ((sin(a2[i]) * sin(b2[i]) * cos(c2[i]
    )) - (cos(a2[i]) * sin(c2[i])))},

    {xy_points_pr_2D_b[2][1], xy_points_pr_2D_b[2][1] + ((xy_points_b[0][1] -
    xy_points_b[0][0]) * (-sin(b2[i])))
    + ((xy_points_b[1][1] - xy_points_b[1][0]) * cos(b2[i]) * sin(c2[i])) + ((
    xy_points_b[2][1] - xy_points_b[2][0]) * cos(b2[i]) * cos(c2[i]))};

    list_b.add(xy_points_angle_b);
    list_b_text.add(xy_points_2D_b);

    float [][] xy_points_pr_u = new float [3][4];
    xy_points_pr_u = list_u.get(i-2);

    float [][] xy_points_pr_2D_u = new float [3][4];
    xy_points_pr_2D_u = list_u_text.get(i-2);

    float [][] xy_points_u = new float [][] {
    {(y_s[i] - y_s[1]), ((y_s[i] - dy[i]) - y_s[1])},
    {x_s[i] + 417, x_s[i] + dx[i] + 417},
    {0, 0}};

    float [][] xy_points_angle_u = new float [][] {
    {xy_points_pr_u[0][1], xy_points_pr_u[0][1] + (xy_points_u[0][1] - xy_points_u
    [0][0]) * cos(a[i]) * cos(b[i])
    + (xy_points_u[1][1] - xy_points_u[1][0]) * ((cos(a[i]) * sin(b[i]) * sin(c[i]))
    - (sin(a[i]) * cos(c[i])))
    + (xy_points_u[2][1] - xy_points_u[2][0]) * ((cos(a[i]) * sin(b[i]) * cos(c[i]))
    + (sin(a[i]) * sin(c[i])))},

    {xy_points_pr_u[1][1], xy_points_pr_u[1][1] + (xy_points_u[0][1] - xy_points_u
    [0][0]) * sin(a[i]) * cos(b[i])
    + (xy_points_u[1][1] - xy_points_u[1][0]) * ((sin(a[i]) * sin(b[i]) * sin(c[i]))
    + (cos(a[i]) * cos(c[i])))
    + (xy_points_u[2][1] - xy_points_u[2][0]) * ((sin(a[i]) * sin(b[i]) * cos(c[i]))
    - (cos(a[i]) * sin(c[i])))},

    {xy_points_pr_u[2][1], xy_points_pr_u[2][1] + (xy_points_u[0][1] - xy_points_u
    [0][0]) * (-sin(b[i]))
    + (xy_points_u[1][1] - xy_points_u[1][0]) * cos(b[i]) * sin(c[i])
    + (xy_points_u[2][1] - xy_points_u[2][0]) * cos(b[i]) * cos(c[i])};

    float [][] xy_points_2D_u = new float [][] {
    {xy_points_pr_2D_u[0][1], xy_points_pr_2D_u[0][1] + (xy_points_u[0][1] -
    xy_points_u[0][0]) * cos(a2[i]) * cos(b2[i])
    + (xy_points_u[1][1] - xy_points_u[1][0]) * ((cos(a2[i]) * sin(b2[i]) * sin(c2[
    i])) - (sin(a2[i]) * cos(c2[i])))
    + (xy_points_u[2][1] - xy_points_u[2][0]) * ((cos(a2[i]) * sin(b2[i]) * cos(c2[
    i])) + (sin(a2[i]) * sin(c2[i])))},

    {xy_points_pr_2D_u[1][1], xy_points_pr_2D_u[1][1] + (xy_points_u[0][1] -
    xy_points_u[0][0]) * sin(a2[i]) * cos(b2[i])
    + (xy_points_u[1][1] - xy_points_u[1][0]) * ((sin(a2[i]) * sin(b2[i]) * sin(c2[i]
    )) + (cos(a2[i]) * cos(c2[i])))
    + (xy_points_u[2][1] - xy_points_u[2][0]) * ((sin(a2[i]) * sin(b2[i]) * cos(c2[i]
    )) - (cos(a2[i]) * sin(c2[i])))},

    {xy_points_pr_2D_u[2][1], xy_points_pr_2D_u[2][1] + (xy_points_u[0][1] -
    xy_points_u[0][0]) * (-sin(b2[i]))

```

```

+ (xy_points_u[1][1] - xy_points_u[1][0]) * cos(b2[i]) * sin(c2[i])
+ (xy_points_u[2][1] - xy_points_u[2][0]) * cos(b2[i]) * cos(c2[i])});

    list_u.add(xy_points_angle_u);
    list_u_text.add(xy_points_2D_u);
}
}

void draw() {
    background(0);
    //load the stitched image
    img = loadImage(name + ".jpg");
    //set the way that the 3D rotates
    translate(width / 4.0, height / 2.0, 100);
    rotateX(rotx);
    rotateY(roty);

    Shape();
}

void Shape(){

    noFill();
    //    fill(255, 0, 255);
    scale(0.2);
    stroke(color(0, 255, 0));
    strokeWeight(0);

    for (int i = 1; i < NOF - 1; i++){

        beginShape();
        texture(img);

        //retrieve the point positions calculated in matrices()
        float [][] rpy_b = new float [3][4];
        float [][] rpy_u = new float [3][4];
        rpy_b = list_b.get(i);
        rpy_u = list_u.get(i);
        float [][] rpy_b_text = new float [3][4];
        float [][] rpy_u_text = new float [3][4];
        rpy_b_text = list_b_text.get(i);
        rpy_u_text = list_u_text.get(i);

        //construct the 4 vertices
        //(connect the points of each segment to each other)
        vertex(rpy_b[0][0], rpy_b[1][0], rpy_b[2][0], rpy_b_text[1][0],
img.height - rpy_b_text[0][0]); //Vb[k]
        vertex(rpy_b[0][1], rpy_b[1][1], rpy_b[2][1], rpy_b_text[1][1],
img.height - rpy_b_text[0][1]); //Vb[k+1]
        vertex(rpy_u[0][1], rpy_u[1][1], rpy_u[2][1], rpy_u_text[1][1],
img.height - rpy_u_text[0][1]); //Vu[k+1]
        vertex(rpy_u[0][0], rpy_u[1][0], rpy_u[2][0], rpy_u_text[1][0],
img.height - rpy_u_text[0][0]); //Vu[k]
        endShape(CLOSE);
    }
}

void mouseDragged() {
    float rate = 0.01;
    rotx += (pmouseY - mouseY) * rate;
    roty += (mouseX - pmouseX) * rate;
}
}

```

## Appendix B

# Map navigation Code in use for a DoE

This appendix contains the Matlab code that we constructed for the DoE that we are explaining in details in Chapter 7.

```
% Load the two images that we are going to use as map and patch
map = 'map.jpg';
patch = 'patch.jpg';

lin = 2;

% Use for loops to choose everytime different combination of parameters
for l = 1:4 % size of radius in features
    for i = 1:4 % color of image
        for j = 1:3 % contrast of image
            for k = 1:3 % size of patch on y direction
                for m = 1:3 % size of patch on x direction
                    for n = 1:3 % location of patch (in y direction)

% Define the parameters that we are going to use
% Different color types of the image and different radius in the features that were
    detected

        grayscale = {map, patch};
        radius = {'1_', '2_', '3_', '4_'};
        gray_black = {[radius{1},map], [radius{1},patch]};
        white_black = {'white_and_black_', radius{1},map}, ['white_and_black_', radius{1},patch];
        black_white = {'black_and_white_', radius{1},map}, ['black_and_white_', radius{1},patch];

        color = {grayscale, gray_black, white_black, black_white};
        nam_color = {'grayscale', 'gray_black', 'white_black', 'black_white'};
        number_color = {'+256', '-256', '1', '-1'};

% Different contrast types
        lighter = {'Lighter_', color{i}{1}}, ['Lighter_', color{i}{2}];
        darker = {'Darker_', color{i}{1}}, ['Darker_', color{i}{2}];
        original = {color{i}{1}, color{i}{2}};

        contrast = {original, lighter, darker};
        nam_contrast = {'original', 'lighter', 'darker'};
        number_contrast = {'0', '+0.01', '-0.01'};

% Location of the patch in Y-direction
        middle = 700;
        up = 59;
```



```

down = 900;

density = {middle, up , down};
nam_density = {'middle', 'up', 'down'};

% Define the pixels of the patch in x-direction
bigX1 = 1;
bigX2 = 330; % bigX2 - bigX1 = 329

medX1 = 41;
medX2 = 314; % medX2 - medX1 = 273

smX1 = 82;
smX2 = 273; % smX2 - smX1 = 191

dimX = { bigX1:bigX2, medX1:medX2, smX1:smX2};

% Define the size of the patch we are going to use
bigW = bigX2 - bigX1;
medW = medX2 - medX1;
smW = smX2 - smX1;

Size = {bigW, medW, smW};
nam_Size = {'big', 'medium', 'small'};

% Begin of image matching procedure
A = imread(contrast{j}{1});
B = imread(contrast{j}{2});

A = imrotate(A,5,'bilinear');
A = A(3315:5036, 770:1100, 1);

B = imrotate(B,5,'bilinear');
B = B(3315:5036, 686:1080, 1);

% Crop a part from the patch image to use it for the image matching
C = B(density{n}:density{n}+Size{k},dimX{m},1);
C=A(1:300,1:300);
figure(10);
imshow(C);
axis on
axis equal;
[x,y] = size(C);

% Caclulation of number of features in the cropped patch
if i==1
    feat_nb = 1;
elseif i==2
    feat_nb = sum(C(:)==0)/(pi*1^2);
elseif i==3
    feat_nb = sum(C(:)==0)/(pi*1^2);
else
    feat_nb = sum(C(:)==255)/(pi*1^2);
end

if sum(C(:))==0
    c = -1;
    xpeak = inf;
    ypeak = inf;
else
    c = normxcorr2(C,A);
    [max_c, imax] = max(abs(c(:)));
% max_c = 1 means the img a contain img b (overlapping)
    [ypeak, xpeak] = ind2sub(size(c),imax(1));
end

```

```

% Calculation of ideal location in the matching procedure
id_middle = middle + 89;
id_up = up + 89;
id_down = down + 89;
% 89 pixels is the displacement between the two images in y-direction

id_result_y = {id_middle, id_up, id_down};
dif_y = id_result_y{n} - (ypeak - Size{k});

id_left = bigX1-48;
id_right = medX1-48;
id_center = smX1-48;
% 48 pixels is the displacement that we manually calculated between the
two images in y-direction

id_result_x = {id_left, id_right, id_center, id_left_1,
id_right_1, id_center_1};
dif_x = id_result_x{m} - (xpeak - Size{m});

dist = sqrt(dif_x^2+dif_y^2);
abs_dist = dist;

if (dif_x < 0 && abs(dif_y) < abs(dif_x))
    dist = -dist;
end

if (dif_y < 0 && abs(dif_x) < abs(dif_y))
    dist = -dist;
end

% Visual representation of the image matching combination on the map
D = imread(contrast{j}{1});
D = imrotate(D,5,'bilinear');
D = D(3315:5036, 770:1100, 1);
figure(1);
imshow(D);
axis on
axis equal;
hold on;
rectangle('Position',[xpeak - Size{m},ypeak - Size{k},Size{
m},Size{k}], 'EdgeColor','red', 'LineWidth',2) % cross
result on the image
rectangle('Position',[id_result_x{m},id_result_y{n},Size{m
},Size{k}], 'EdgeColor','green', 'LineStyle','--',
'LineWidth',1) % visual ideal result

% Visual representation of the cross-correlation matrix
figure(2);
imshow(c);
axis on
rectangle('Position',[xpeak - Size{m},ypeak - Size{k},Size{
m},Size{k}], 'EdgeColor','red', 'LineWidth',2) % cross
result
rectangle('Position',[id_result_x{m},id_result_y{n},Size{m
},Size{k}], 'EdgeColor','green', 'LineStyle','--',
'LineWidth',1) % ideal cross visual result

% Fill the results in an excel file
stor{1,1} = 'Image for map';
stor{lin,1} = ['',contrast{j}{1},''];
stor{1,2} = 'Image for patch';
stor{lin,2} = ['',contrast{j}{2},''];
stor{1,3} = 'Image Color';
stor{lin,3} = [nam_color{i}];
stor{1,4} = '# Image Color';
stor{lin,4} = [number_color{i}];
stor{1,5} = 'Image Contrast';

```

```

stor{lin,5} = [nam_contrast{j}];
stor{1,6} = '# Image Contrast';
stor{lin,6} = [number_contrast{j}];
stor{1,7} = 'Location of patch';
stor{lin,7} = density{n};
stor{1,8} = 'dimX';
stor{lin,8} = Size{m};
stor{1,9} = 'dimY';
stor{lin,9} = Size{k};
stor{1,10} = 'Radius';
stor{lin,10} = 1;
stor{1,11} = 'Ypeak';
stor{lin,11} = ypeak - Size{k};
stor{1,12} = 'Id_Ypeak';
stor{lin,12} = id_result_y{n};
stor{1,13} = 'Xpeak';
stor{lin,13} = xpeak - Size{m};
stor{1,14} = 'Id_Xpeak';
stor{lin,14} = id_result_x{m};
stor{1,15} = 'Difference on y';
stor{lin,15} = dif_y;
stor{1,16} = 'Difference on y in mm';
stor{lin,16} = dif_y/25;
stor{1,17} = 'Abs Difference on y';
stor{lin,17} = abs(dif_y);
stor{1,18} = 'Difference on x';
stor{lin,18} = dif_x;
stor{1,19} = 'Difference on x in mm';
stor{lin,19} = dif_x/25;
stor{1,20} = 'Abs Difference on x';
stor{lin,20} = abs(dif_x);
stor{1,21} = 'Difference';
stor{lin,21} = abs_dist;
stor{1,22} = 'Difference in mm';
stor{lin,22} = abs_dist/25;
stor{1,23} = 'Difference';
stor{lin,23} = dist;
stor{1,24} = 'Difference in mm';
stor{lin,24} = dist/25;
stor{1,25} = 'Correlation';
stor{lin,25} = max_c;
stor{1,26} = 'Features nb';
stor{lin,26} = feat_nb;
lin = lin+1;
xlswrite('DoE.xlsx',stor);

end
end
end
end
end
end

```