DESIGN OF AN IMAGE ACQUISITION SETUP FOR MIMIC TRACKING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ÖZGÜLER MİNE AKÖNER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

AUGUST 2007

Approval of the thesis:

DESIGN OF AN IMAGE ACQUISITION SETUP FOR MIMIC TRACKING

submitted by **ÖZGÜLER MİNE AKÖNER** in partial fulfillment of the requirements for the degree of **Master of Science in Mechanical Engineering Department, Middle East Technical University** by,

Prof Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Kemal İder
Head of Department, **Mechanical Engineering** _____

Asst. Prof. Dr. A. Buğra Koku
Supervisor, **Mechanical Engineering Dept., METU** _____

Examining Committee Members:

Prof. Dr. Reşit Soylu
Mechanical Engineering Dept., METU _____

Asst. Prof. Dr. A. Buğra Koku
Mechanical Engineering Dept., METU _____

Asst. Prof. Dr. Melik Dölen
Mechanical Engineering Dept., METU _____

Assoc. Prof. Dr. Veysel Gazi
Electrical and Electronics Engineering Dept., TOBB-ETU _____

Asst. Prof. Dr. Erol Şahin
Computer Engineering Dept., METU _____

Date: 28.08.2007

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Özgüler Mine AKÖNER

Signature :

ABSTRACT

DESIGN OF AN IMAGE ACQUISITION SETUP FOR MIMIC TRACKING

Aköner, Özgüler Mine

M.S., Department of Mechanical Engineering

Supervisor: Asst. Prof. Dr. A. Buğra Koku

August 2007, 88 pages

With the advances in computer technology and the changing needs of people's daily lives, robots start to offer alternative solutions. As one of these solutions, the branch of humanoid robots emerged as advanced robots that can interact with people. Robot faces are one of the most effective means of interacting with people; since they can express their emotions and reactions through facial mimics. However, the development of realistic robot faces necessitates the knowledge of the trajectories and displacements of actual face mimics.

In this study, a setup (both hardware and software), that can be used for tracking critical points on human face while exhibiting mimics, is developed. From the outputs of this setup, the mimic trajectories are going to be extracted. The setup is designed and manufactured to be durable to external effects so that with a single camera calibration procedure the 3D reconstruction can be carried out several times. The setup consists of two webcams that are specially oriented for mimic tracking. The images taken from the cameras are corrected; their features are extracted using image processing algorithms; the centroids of the features are found; correspondence is carried out and the reconstruction is made. This system can also be used for any special point tracking or volumetric measurement purposes.

Keywords: Stereo Vision, Mimic Tracking, Image Processing, Distortion Correction.

ÖZ

YÜZ HAREKETLERİNİ TAKİP ETMEK İÇİN BİR GÖRÜNTÜ TOPLAMA DÜZENEĞİ TASARIMI

Aköner, Özgüler Mine
Yüksek Lisans, Makina Mühendisliği Bölümü
Tez Yöneticisi: Yrd. Doç. Dr. A. Buğra KOKU

Ağustos 2007, 88 sayfa

Bilgisayar teknolojisindeki gelişmeler ve insanoğlunun değişen günlük ihtiyaçlarıyla birlikte robotlar alternatif çözümler üretmeye başlamıştır. Bu çözümlerden biri olarak da insansı robotlar insanlarla etkileşim halinde bulunabilen gelişmiş robotlar olarak ortaya çıkmıştır. Robot yüzler insanların doğrudan iletişime girebilecekleri en etkili arayüzlerden biridir; çünkü duygularını ve tepkilerini yüz mimikleriyle gösterebilirler. Bununla birlikte, gerçekçi robot yüzlerin geliştirilebilmesi için insanların

yüz mimiklerinin takip ettiği yolların ve gittiği mesafelerin bilinmesi gereklidir.

Bu çalışmada,insan yüzündeki kritik noktaları, çeşitli yüz hareketleri yapılırken, takip edebilecek bir düzenek ve ilgili yazılımın geliştirilmesi planlanmıştır. Bu düzeneğin çıktılarından yüz hareketleri çıkartılabilir. Düzenek dış etkilere dayanabilecek şekilde tasarlanmış ve üretilmiştir.Böylece tek bir kamera kalibrasyon rutiniyle en az birkaç sefer 3 boyutlu nokta bulma yapılabilmesi amaçlanmıştır. Düzenek yüz videolarını çekebilmek için özel olarak yönlendirilmiş iki adet webcamden oluşmaktadır. Kameralardan alınan görüntüler önce düzeltilir; özellikleri görüntü işleme yordamlarıyla ortaya çıkartılır; merkezleri bulunur; ilişkilendirme yapılır ve 3 boyutlu nokta koordinatı bulunur. Bu sistem ayrıca başka 3 boyulu nokta takip ettirme veya hacimsel ölçülendirme uygulamaları için de kullanılabilir.

Anahtar Kelimeler: Stereo Görüntü, Mimik Takibi, Görüntü İşleme, Çarpılma Düzeltilmesi

To My Dear Mother and Father

To My Sweet Sister and Brother

And

To Kıvanç...

ACKNOWLEDGEMENTS

Firstly, I would like to thank to my thesis advisor Asst. Prof. Dr. Buğra Koku not only for his support and guidance but also for his friendly and optimistic attitude during this research.

I am grateful to Mr. Refik Toksöz and Mr. Özgür Geşli for their guidance on 3D face modeling and providing me with the 3D scan data of my own face. Also I would like to thank Mr. Mehmet Ali Uysal and Ms. Ödül Işıtman for their valuable help during the plaster molding process of my face.

I am very grateful to my mother and father for their invaluable efforts and their endless love, patience and encouragement. I also would like to thank my brother and sister for their love and friendship.

Special thanks go to my colleagues and friends Res. Asst. Müjde Sarı and Res. Asst. Orhan Ölçücüoğlu for their helpful discussions and friendship. Also I would like to thank all my friends in the Department of Mechanical Engineering for the enjoyable time spent together.

Finally, I would like to thank my dear husband Kıvanç Azgın for his understanding, invaluable support and for being in my life with his endless love.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

LIST OF TABLES

TABLE

CHAPTER 1

INTRODUCTION

1.1 Literature Survey

Robot, as a word, was first introduced by Karel Čapek in his play, Rossum's Universal Robots in 1920's. However, the idea of robots, although it was not named so, dates back to ancient times when people started to think of creating artificial agents mimicking animals and humans. In 15th century Leonardo da Vinci was the first inventor to design the ancestor of modern humanoids, a mechanical knight that was able to sit up, wave its arms and move his head and jaw. Robots, as it is understood today, emerged in 1960's when industrial robots started serving in manufacturing lines. Industrial robots were usually in the form of robot arms which completed their tasks with great speed and accuracy and therefore they were very beneficial for mass production. These robots were well-utilizable in environments where man did not need to interact with them directly; however with the developing technology people felt the necessity of interacting with them more frequently.

In the meantime, with the advances in computer technology and the changing needs of people's daily lives, robots came into a different view. Along with other technological products a new area of robot use emerged where they would directly interact not only with each other but also with people. Therefore, the "robots as a tool" philosophy was beginning to be revised and robots entered people's lives as agents that people communicate with, get their work done through direct interfaces and even entertain with. These developments gave birth to two concepts, namely social robots and human-robot interaction.

Dautenhahn and Billard [1] define a social robot as follows;

> *"Social robots are embodied agents that are part of a heterogeneous group: a society of robots or humans. They are able to recognize each other and engage in social interactions, they possess histories (perceive and interpret the world in terms of their own experience), and they explicitly communicate with and learn from each other."*

Fong et al. [2] discusses that for developing social robots social learning and imitation, gesture and natural language communication, emotion and recognition of interaction partners are important issues to be foreseen. A major area of the research efforts in human-robot interaction is anthropomorphism. Anthropomorphism comes from the merging of two Greek words, *anthropos* for man and *morphe* for form/structure. Duffy [3] defines anthropomorphism as the tendency to attribute human characteristics to inanimate objects, animals and others with a view to helping us rationalize their actions. Anthropomorphism is used to facilitate the social interaction between the human and the robot. Several

discussions are put forward for the design of anthropomorphic robots. For instance, while designing robot heads not all of the researchers designed facial features. There exist different humanoid robots which express their human qualities through *suggestion* of features [4]. Duffy [3] suggests that [anthropomorphic] social robots should exploit people's expectations of behaviors rather than necessarily trying to force people to believe that the robot has human reasoning capabilities. Mashiro Mori developed a theory called "The Uncanny Valley" (Figure 1.1) which states that as the robot's appearance starts to resemble that of a human, at a point where the robot's appearance is not 100%, people start to be disturbed by the imperfections of the robot's appearance. This may cause people to reject the robot as a social partner. Mori argues that a limit for familiarity should be set during the design of humanoid robots.

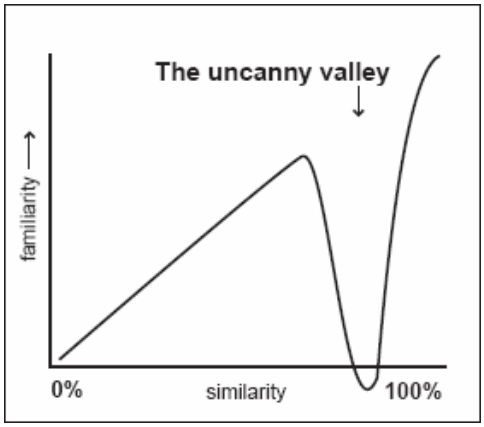


Figure 1.1 Mori's The Uncanny Valley [4]

In accordance with Mori's *The Uncanny Valley* theory DiSalvo et al. [4] proposed some considerations to take into account while designing a humanoid robot head. They studied 48 robot heads to conduct several surveys to extract people's perception of humanness of robot heads.

According to them, the robot should have wide head and wide eyes to preserve some robotness, by making the robot look less human. Also, the feature set from brow line to bottom of mouth should dominate the face. In addition to these, the face should have detailed eyes, nose, mouth and eyelids. Moreover, the robot should have a skin to cover the mechanical and electrical components and a curvy form in the forehead, back head and on the cheeks.

This research and the public interest on humanoid robots ignited the practical usage of these in people's every day lives. Every year The International Federation of Robotics (IFR) in cooperation with United Nations Economic Commission for Europe (UNECE) publishes the *World Robotics* survey. UNECE/IFR groups robots into three categories; industrial robots, service robots for professional use and service robots for personal/private use. Humanoid robots are classified in professional service robotics group. In the 2005 issue of World Robotics [5], the value of the stock for humanoid robots is expected to be approximately $420 million between 2005 and 2008 (Figure 1.2).

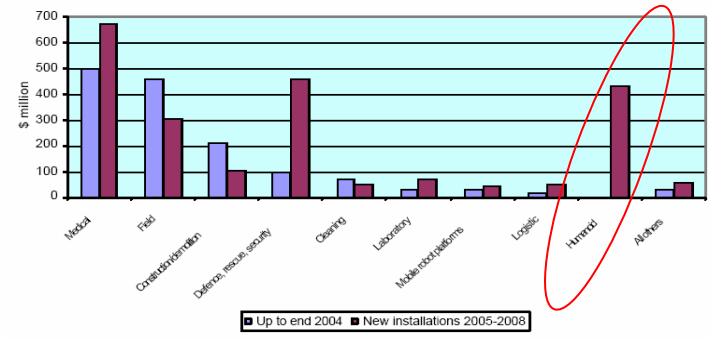
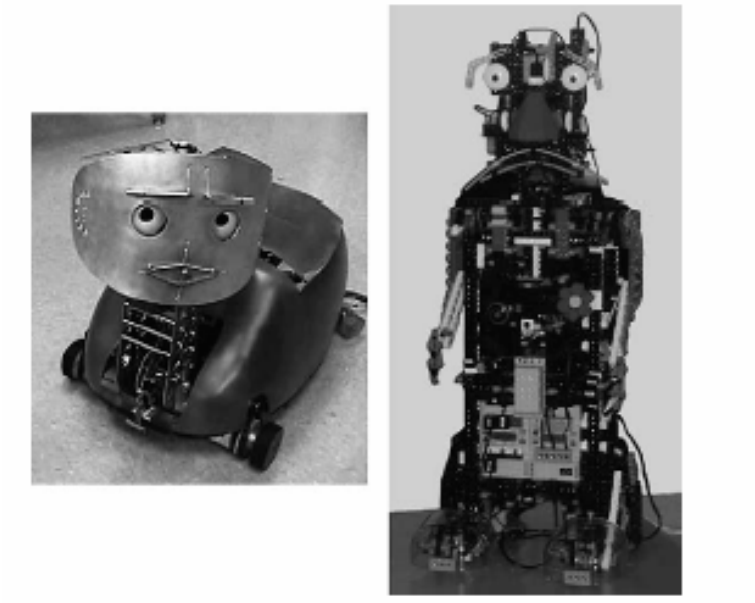


Figure 1.2 Service robots for professional use. Stock at the end of 2004 and projected installations in 2005-2008. [5]

While UNECE/IFR classifies humanoids as professional service robots, social robots can be both utilized for professional and personal use. They are usually designed as companions for the elderly [6,7], pets [8], as tour guides [9], for rehabilitation of autistic children [10] and for entertainment. Some also serve as platforms on which deeper models of human-robot interaction can be developed [11,12]. Within the context of this research, it will be only focused on social robots that exhibit facial expressions.

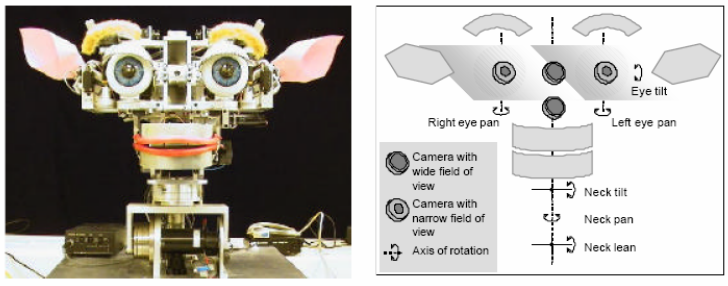
Of the simplest faces, Sparky [13] and Feelix [14] are noteworthy (Figure 1.3). Sparky has 4 degrees of freedom (DOF) (eyebrows, eyelids and lips) and Feelix has also 4 DOF (two eyebrows and two lips). These faces can mimic 6 basic facial expressions, i.e. anger, sadness, fear, happiness, surprise and neutral.



(a) (b)

Figure 1.3 (a) Sparky (b) Feelix

Compared to these two robots, Kismet [15,16] is much more complex with its 15 DOF including eyebrows, eyelids, ears, lips and a mouth (Figure 1.4). These actuators work together to display various expressional features. Besides, Kismet has an eye-vision system which imitates that of a human. It has eyes which can turn independently in horizontal direction also which can turn together in vertical direction. Two cameras with narrower field of view rotate with the eyes. There also exist two cameras with wider field of view whose directions are unaffected with the orientation of the eyes. It displays various motor actions to deal with the limitations of its visual perception. For instance, if a person is too far away to be imaged accurately, Kismet calls the person closer. Similarly, if the person is too near to be viewed by the narrow field cameras Kismet tries to draw its head back. Kismet generates its emotions based on an "affect-space" on which each emotion can be expressed out of three independent dimensions; arousal, valence and stance (Figure 1.5). These characteristics of Kismet make it unique among other anthropomorphic robot faces.



(a) (b)

Figure 1.4 (a)Kismet (b)Kismet's degrees of freedom relevant to its visual perception. [15]

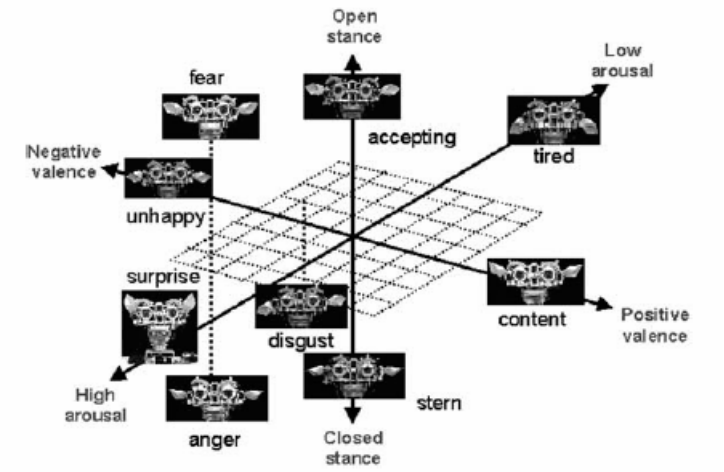


Figure 1.5 Kismet displays its emotions based on a three dimensional affect space

Takanishi et al. of Waseda University [17] developed a face robot, WE-3RII which has eyeballs, eyebrows, upper and lower lips, neck and jaw. The eyeballs are 2 DOF, the neck is 4 DOF, each eyebrow and lip has 3 DOF and the jaw is 1 DOF, all driven by stepper motors and tendons. After WE-3RII, they developed WE-3RIII [18] which has auditory sensation, tactile sensation and temperature sensation in addition to the previous version's properties. The robot can exhibit the six basic facial expressions, happiness, anger, surprise, fear, sadness and disgust (Figure 1.6).

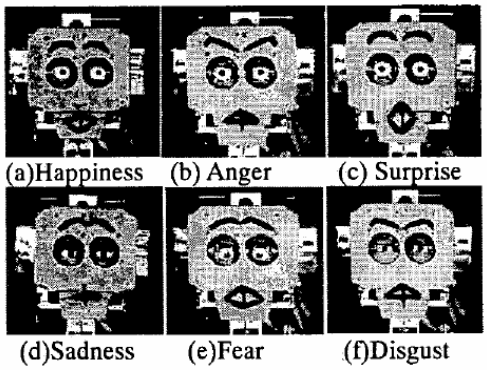


Figure 1.6 Six basic facial expressions of WE-3RIII

WE-3RIII, Kismet, Sparky and Feelix use a metal frame as the face. However, researchers have developed more realistic faces to imitate that of a human face. These often utilize soft skins to realistically display the facial expressions, hair, even teeth and tongue. There exist several robots in this context most of which were devised in Japan. Science University of Tokyo has developed one of the most realistic anthropomorphic robot faces, Saya [19]. They claim that face-to-face conversation is a multi-channel mode of interaction and it is ideal for human-robot interaction. They also argue that realizing human-like facial expressions is an engineering challenge; therefore, they use a soft silicone skin to realize these expressional features. SAYA can realize 6 basic facial expressions (Figure 1.7). To realize these expressions, 19 control points are used whose movements are extracted empirically. These control points are actuated from behind by a special pneumatic actuator which is designed by them (Figure 1.8). In addition to the facial expression degrees of freedom, the robot has a 2 DOF eye mechanism, 1 DOF eyelids and a 4 DOF neck mechanism.

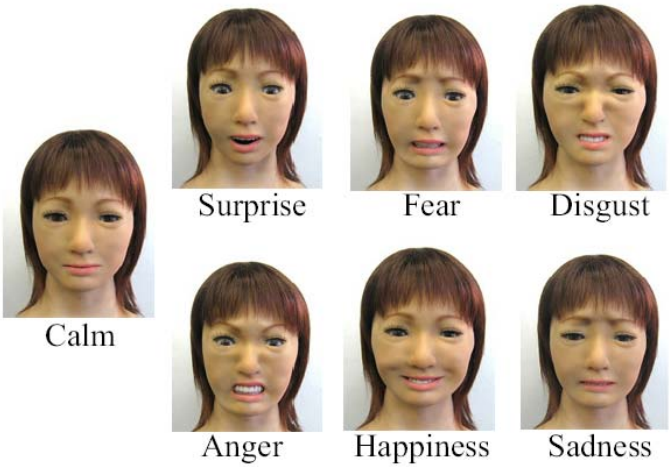


Figure 1.7 Six basic facial expressions of SAYA

Figure 1.8 Front side and reverse side of the skin

Another realistic anthropomorphic face is K-Bot of David Hanson (Figure 1.9). Among other very successful robot faces of Hanson (Albert Einstein HUBO, Philip K. Dick Robot, etc.) only K-Bot will be introduced since it is the first robot with which Hanson got internationally known and all the other robots can be considered as the clones of K-Bot. K-Bot has 24 servomotors covering various muscles of the human face. It has a special skin, developed by Hanson himself, called Frubber™. It matches the characteristics of human skin; therefore realistic facial expressions can be created.

Figure 1.9 K-Bot

Hanson, a former art student himself, claims that the traditional approach of "The Uncanny Valley" should be revised with the advances in

animatronics (entertainment robotics). The general approach of robotics researchers is to create toy-like robots to avoid the uncanny valley. Hanson claims that it is possible to create robots with human-like facial characteristics if a more aesthetics based point of view is picked [20].

Most of the robots above and many facial modeling systems were designed referring to the Facial Action Coding System (FACS) of Paul Ekman et al. [21]. FACS is a universally acclaimed method to describe the facial expressions. It relates the expressions to the action of muscles. Every expression is associated with an Action Unit (AU). AUs are used for measurement because for a single expression several muscles might be involved. The expressions of the human face are divided into 44 AUs. In Figure 1.10 several AUs are depicted.

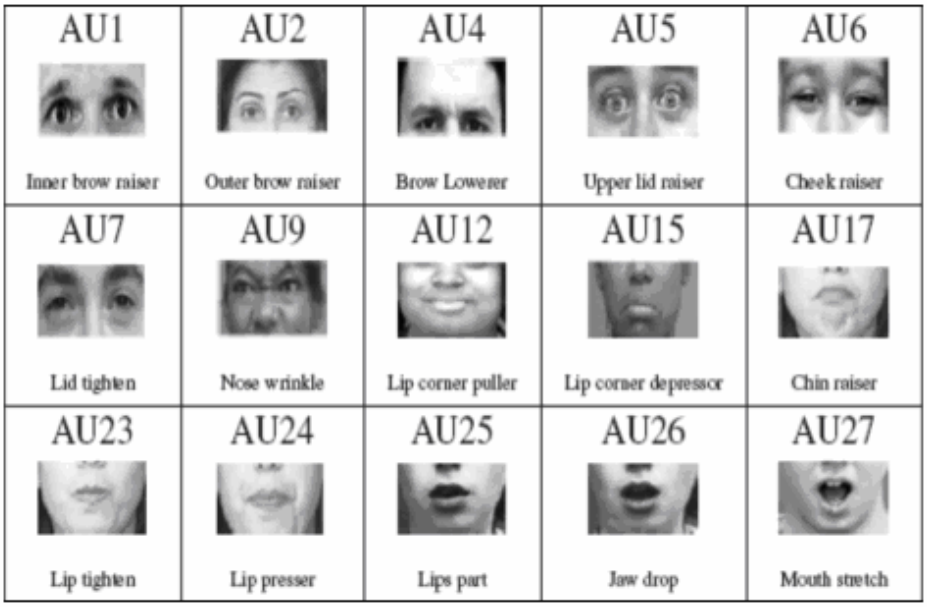


Figure 1.10 Examples of some action units [22]

1.2 Objective of the Thesis

For designing an anthropomorphic robot face with a soft skin, it is obvious that the skin should be actuated from various regions or points which can be called as control points. So far, the selection of these control points was done empirically [19]. However, we believe that a systematic way of determining the location of these control points can be found. This will guide the robot face designer towards a more systematic approach on anthropomorphic robot design. The work carried as a part of this thesis work is the first attempt for finding this aforementioned systematic. Our proposed method is based on tracking various marked points on a human's face while the person is performing certain mimics. By analyzing the motion of the marked points the regions that define the motion will be determined. Once this is done, next thing to do is to show that similar motions patterns can be realized on a silicone skin. This thesis focuses on 3D reconstruction algorithms of still images and extracting the locations of multiple points on the face and at the end a mimic tracking example is given.

For face tracking a 3D vision setup formed by two VGA webcams is used. This setup is used to extract the 3D locations of various points marked on the face. By using a 3D vision setup, it is aimed to minimize the pose variations. MATLAB Image Processing Toolbox is used to run basic image processing routines. While tracking the points on the face, the face is assumed to be symmetric with respect to vertical plane. The setup can be further developed for other high-resolution tracking purposes.

1.3 Scope of the Thesis

The first objective of the thesis, which was to design a robot face that is able to mimic human facial expressions, led the author to form a roadmap for such a design. However, as this roadmap started to reveal itself, it was noticed that this design procedure had several intermediate steps. In order to display fully realistic mimics, the silicone skin should be examined thoroughly and the necessary motion schemes should be realized on silicone. These motion schemes should be determined and their trajectories should be measured on a human face. This led to design such a stereo vision setup for tracking the human mimics. After the measurement of the mimic trajectories, trajectories similar to those on the human face should be realized on silicone. While imitating these motion schemes on the silicone, several other issues like choice of actuators and contact surfaces for actuator to silicone should be solved. Another output of mimic tracking is going to be the determination of some regions on the face that does not move on certain mimics. These will enable the minimization of the number of discrete actuators. After all of these are determined, the skull will be designed, a sample face should be molded and the motion schemes should be generated.

Although this thesis only deals with 3D reconstruction, the aim is to design a robot face on the greater scale. This work is the first step of this design procedure which is probably going to take several years. Since the initial aim was to design the face itself, the literature survey focused on recently developed robot faces. The rest of the thesis goes as follows:

In Chapter 2, basics of 3D stereo vision theory is explained in detail. The fundamentals of image processing algorithms used in the thesis are explained in Chapter 3. In Chapter 4, design of the setup, camera calibration, frame extraction, distortion correction algorithms are explained along with the overall software with an example on mimic tracking. Conclusions and future work will be given in Chapter 5.

CHAPTER 2

3D COMPUTER VISION THEORY

This chapter introduces the theory behind 3D coordinate extraction. Computation of the 3D coordinates of a point on the face requires knowledge on the perspective camera projection and the lens optics. Various color models enable feature extraction in different color spaces. Finally an introduction to the camera parameters, camera calibration and inverse perspective transformation forms the computer vision foundation of this thesis.

2.1 Image Formation Principles

An image is a two-dimensional pattern of brightness [23]. It can be classified in two main groups; *intensity images* and *range images*. Intensity images give information about light intensities which is imposed on a photosensitive device. Meanwhile, range images approximate the shape and distance (3D structure) of the scene through various sensors like sonars or laser scanners [24]. In this case, images are acquired through webcams which produce intensity images.

Formation of an intensity image depends on several parameters. These can be listed as:

- Optical Parameters which describe the optics of the sensor; like lens type, focal length, field of view, etc.
- Photometric Parameters give information about the light energy reaching the sensor after being reflected from the viewed object.
- Geometric Parameters which determine the image location of a 3D point. These include the projections, camera position and orientation, camera distortions due to imperfections of the lens.

Among these parameters only optical and geometric aspects of image formation are going to be reviewed in this section since photometric parameters are not directly used within the scope of this thesis.

2.1.1 The Perspective Camera Projection Model

Camera projection models are used to determine the image plane location of the viewed scene. Among various models that have been developed so far, the perspective camera model is the most commonly used one.

The perspective camera projection model (Figure 2.1) is also called the *pinhole model* due to the fact that the aperture through which the light enters can be modeled as a single point. Assuming the aperture as a single point defines a distinct ray of light from every point on the scene through the pinhole. This leads to sharp and undistorted images.

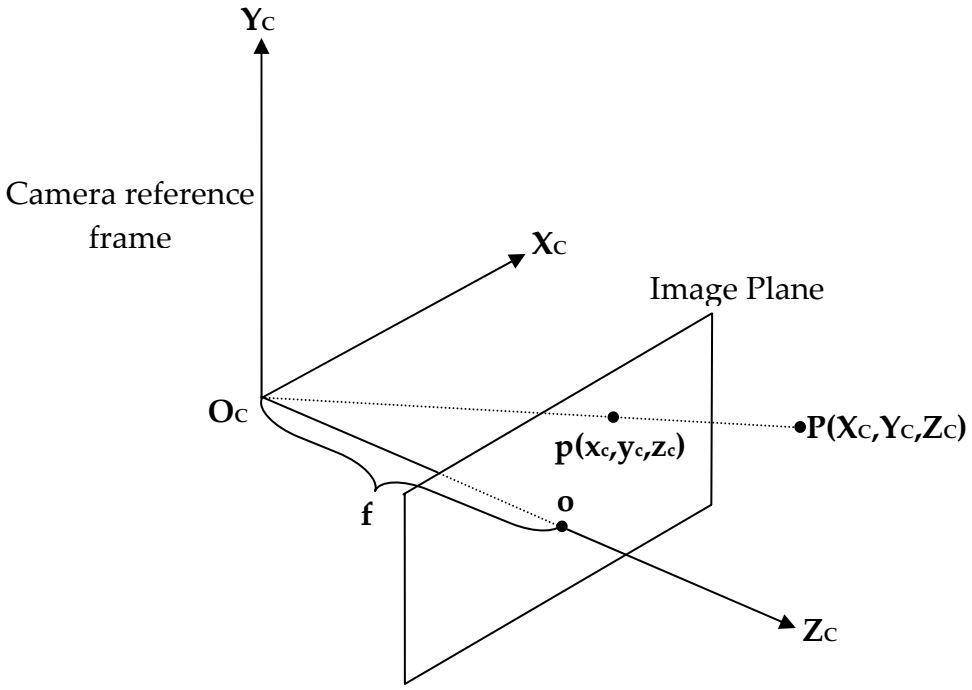


Figure 2.1 The Perspective Camera Projection Model

The model shown in the figure has its origin, O_c, at the pinhole. The *optical axis*, which is also defined as the z-axis, lies perpendicular from the pinhole to the image plane. The distance between the image plane and O_c is the *focal length*, *f*. The intersection of the optical axis and the image plane, *o*, is called the *principal point*. In perspective projection, it is desired to find the projection of the scene point **P** on the image plane. This projection, **p**, has coordinates **p** = (x_c, y_c, z_c) and the scene point has coordinates **P** = (X_c, Y_c, Z_c). Note that all the equations are written with respect to camera reference frame. By using similar triangles in Figure 2.1 we can derive the following relations:

$$\frac{x_c}{f} = \frac{X_c}{Z_c} \tag{2.1}$$

$$\frac{y_c}{f} = \frac{Y_c}{Z_c} \tag{2.2}$$

The image forms on the plane where the light focuses; therefore the third component of the image point is $z_c = f$.

2.1.2 Lens Optics

The perspective camera projection model assumes an infinitesimal aperture area, namely the pinhole. However, the basic problem of a pinhole aperture is the length of its exposure time. The photosensitive device should receive a certain amount of light to form an understandable image. Therefore, either the exposure time should be kept too long or a larger aperture should be used, which violates the perspective projection model.

Lenses allow the usage of wider apertures and are used to focus light rays coming from a finite area. In this fashion the same equations that are used for pinhole camera model can be used to determine the location of the image point. The simplest optical system, the thin lens model (Figure 2.2) can be used to model more complicated optical systems.

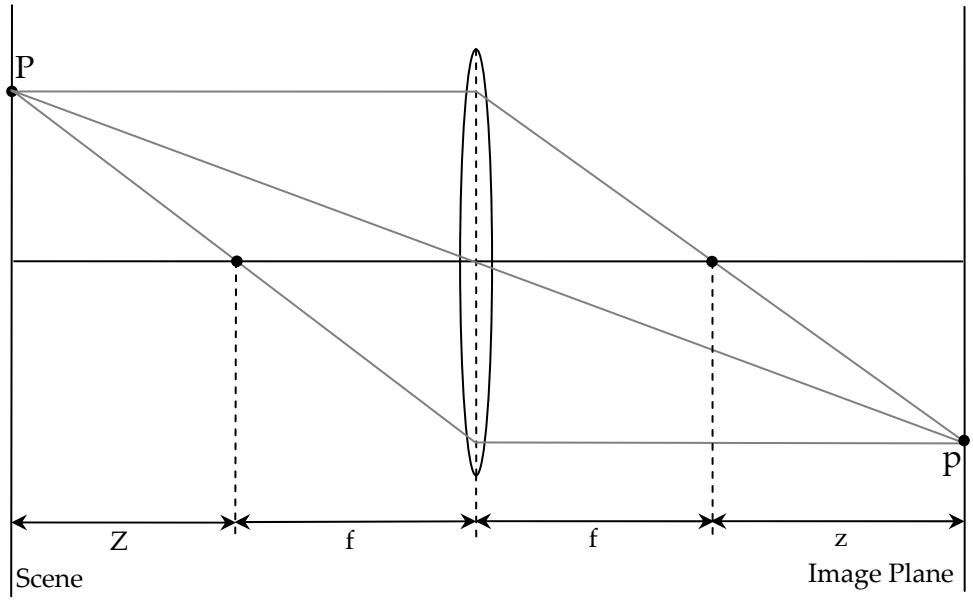


Figure 2.2 Thin Lens Model

The fundamental equation of thin lenses is:

$$\frac{1}{Z+f}+\frac{1}{z+f}=\frac{1}{f} \tag{2.3}$$

This equation reveals that objects at different distances come into focus at different image plane distances. Other points at different distances are viewed as blur circles. If those points in a range of distances can be viewed well enough, i.e. they are sufficiently in focus; this range of distances is called the *depth of field*. The "sufficiently" in the above sentence means, the diameter of the blur circle is below the resolution of the camera.

The *effective diameter*, *d*, of the lens is determined by the aperture as some of the light rays may be prevented by the aperture to reach the lens. Therefore, usually *d* is smaller than the real physical diameter of the lens.

Field of view of the lens is determined by the focal length and the effective diameter. It reveals how much of the screen is seen by the camera. Field of view, *w* is defined as;

$$\tan w = \left(\frac{d}{2f} \right) \tag{2.4}$$

2.1.3 Image Acquisition

An image acquisition system usually consists of 3 components; a camera, a frame grabber and a computer. Incoming light forms voltage on the photosensitive array, whose properties are going to be discussed on the following sections. The sensor outputs an analog video signal and frame grabber samples and quantizes this signal and buffers it in the memory. The buffered signal is acquired by the computer on which suitable image processing algorithms can be run.

In our case we use a webcam using the USB port as the image acquisition system. Webcams can be directly connected to the computer through the parallel, USB or the Firewire port omitting the frame grabber of the classical setups. The webcam software does the frame grabbing in preset intervals. Webcams provide an easy and cheap solution for computer vision setups. They are usually used for transferring images and videos over the internet; therefore the resolutions of webcam sensors do not go

higher than 640 x 480. Webcams typically use two kinds of photosensitive arrays;

- CMOS (Complementary-Metal-Oxide-Semiconductor)
- CCD (Charge-Coupled-Device)

Both of these sensor types convert the light intensity to voltage. CCD sensors read the charge on each array cell by transporting the charge across the chip and reading it from one corner. CMOS sensors have transistors at each pixel and transport the charge by using tiny wires. Usually CCD sensors produce a higher quality and less noisy images compared to CMOS sensors. Especially in low light CMOS sensors tend to fail. However, the production of CMOS sensors is much easier; therefore they are very inexpensive compared to CCD sensors and they use less power.

2.1.4 Digital Images and Color Spaces

Digital images are formed on photosensitive arrays which is the physical image plane. The photosensitive array, which is the image sensor, is a 2D m x n array which determines the resolution of the image. The resolution of the image is measured by the pixels, i.e. picture elements of the image. The pixels represent every array member of the image sensor.

Digital images can be of 3 types

- Binary images: Pixels of binary images are valued either 0 or 1. Binary images are usually created as a result of image processing operations like thresholding.

- Gray-scale images: Gray-scale images are formed of pixels which are shades of gray. The lowest intensity is black growing into white, the highest. For gray-scale images usually one byte is used to represent the image brightness; i.e. 256 gray levels are present. However, in applications where finer shades are required more brightness levels like 16 bits per sample may be used.
- Color images: Color images need three gray-level image components according to which color space it belongs. Usually color images are represented in RGB (red-green-blue) or HSV (hue-saturation-value) color spaces. However, different color models can also be used to represent digital color images. Below is a brief explanation of these models.

2.1.4.1 RGB Color Model

In RGB color space red, green and blue are additively combined to produce other colors. Each pixel is represented by red, green and blue components; usually all of them are represented by 8 bits. Therefore the image is stored as 24-bit. This produces a potential of 16-million colors and these images are called *truecolor images*.

2.1.4.2 HSV Color Model

HSV stands for Hue, Saturation and Value. Hue, H, represents the color, such as red, green, yellow etc., and ranges from 0-360°. Saturation, S, can be called as the "vibrancy" of the color. An image with low saturation values looks faded and more grayness is present. Saturation ranges from

0-100%. Value, V, is the brightness of the color and ranges from 0-100% (Figure 2.3).

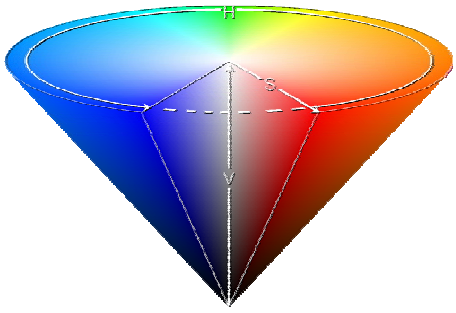


Figure 2.3 The HSV cone

2.1.4.3 YUV and YIQ Color Models

YUV color model is composed of one luminance (brightness) component, Y, and two chrominance (color) components, U and V. This color model is also called YPbPr for analog video applications and YCbCr for digital video. In this model all RGB signals are transformed into YUV signal by the following matrix representation.

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.5}$$

Another luminance-chrominance based color model is the YIQ color space used by the NTSC color TV system. This model is very similar to YUV, Y again models the luminance; I and Q models the chrominance by a simple 33° rotation of the U-V plane.

2.2 Photogrammetry

Image is basically a 2D mapping of a 3D scene. Therefore, during this mapping one of the 3 variables, the depth information, of 3D space is lost. If one desires to reconstruct the 3D information from a 2D projection, more than one image is necessary. The problem of computing 3D coordinates of an object via various photographic images is called *photogrammetry*. For computation of 3D location of a point various parameters need to be considered. These parameters are explained in the forthcoming sections.

2.2.1 Camera Parameters

The mapping of a 2D image to 3D world coordinates requires the usage of 3 different reference frames, the *image reference frame*, *camera reference frame* and *world reference frame*. The image reference frame defines the 2D pixel coordinates of the image projection of the scene. Camera reference frame displays the 3D viewer-centered coordinates of the scene and world reference frame is the absolute coordinates of the scene. The parameters that define the transformation between these frames are called *camera parameters*.

Camera characteristics are classified in two main groups, the *intrinsic parameters* and the *extrinsic parameters*. Intrinsic parameters link the pixel coordinates of an image point on the image reference frame to the corresponding 2D coordinates on the camera reference frame. Extrinsic parameters are the geometric relations which define the orientation and the location of the camera reference frame with respect to the world

reference frame. These values are extracted by solving the camera calibration problem.

2.2.1.1 Camera Calibration

Camera calibration is a vital step for 3D computer vision applications. It is necessary to determine the intrinsic and extrinsic parameters of the vision system. The main idea is to write to projection equations linking the known coordinates of a set of 3D points and their projections and solve for the camera parameters [24]. For calibration a MATLAB toolbox developed by Jean-Yves Bouguet of California Institute of Technology was used [29]. This toolbox is based on Zhang's [25] calibration procedure for finding the extrinsic parameters. Zhang's procedure makes use of a model plane with a calibration grid on it. Several images of the model plane under different orientations are taken to extract the positions of several image points. However Zhang's model deals only with radial lens distortions. To include the tangential distortions, the toolbox uses Heikkila and Silvén's model [26] to calculate the lens distortions. The details of Zhang's calibration are explained below.

The equations that relate the 3D world reference frame to 2D image plane needs a special notation to be followed. The 2D pixel coordinates of a point is represented by $m = [U, V]^T$ and the 3D absolute coordinates of the same point is represented by $M = [X, Y, Z]^T$. The augmented forms of

these vectors are $\tilde{m} = [u, v, w]^T$ and $\tilde{M} = [X, Y, Z, 1]^T$ where $U = u/w$ and $V = v/w$.

The relation between the 3D coordinates M and its image projection m is written as;

$$\tilde{m} = A[R \quad t]\tilde{M} \tag{2.6}$$

where

A: intrinsic camera parameter matrix

R: rotation matrix

t: translation vector

$[R \quad t]$ is the transformation matrix that relates the world coordinates to the camera reference frame. The rotation and translation matrices are the extrinsic parameters. A, the camera intrinsic matrix can be written as

$$A = \begin{bmatrix} \alpha & c & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

In this matrix, α and β are the scale factors in u and v axes, which come from the perspective projection equations, c is the skewness parameter and (u_0, v_0) is the pixel coordinates of principal point. To further understand these parameters, the following expressions should be written.

By equations (2.1) and (2.2), the pinhole model can be expressed as

$$\begin{bmatrix} x_c \\ y_c \end{bmatrix} = \frac{f}{Z_c} \begin{bmatrix} X_c \\ Y_c \end{bmatrix} \tag{2.8}$$

Neglecting all possible lens distortions and skewnesses

$$x_c = -(u - u_0)s_x \tag{2.9}$$

$$y_c = -(v - v_0)s_y \tag{2.10}$$

where u and v are the image pixel coordinates and (s_x, s_y) is the size of pixels in millimeters in horizontal and vertical direction respectively. Equations (2.9) and (2.10) can be rewritten in matrix form as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -1/s_x & 0 \\ 0 & -1/s_y \end{bmatrix} \begin{bmatrix} x_c \\ y_c \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \tag{2.11}$$

Combining this with pinhole projection model the camera reference frame based coordinates can be written as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -f/s_x & 0 & u_0 \\ 0 & -f/s_y & v_0 \end{bmatrix} \frac{1}{Z_c} \begin{bmatrix} X_c \\ Y_c \\ 1 \end{bmatrix} \tag{2.12}$$

Then $\alpha = \frac{-f}{s_x}$ and $\beta = \frac{-f}{s_y}$.

Without loss of generality, the model plane can be assumed to be on $Z = 0$. If one denotes the i[th] column of the rotation matrix R by r_i; equation (2.6) turns out to be

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = A\begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix}\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = A\begin{bmatrix} r_1 & r_2 & t \end{bmatrix}\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{2.13}$$

For all the forthcoming equations Z is equal to 0; therefore M can be denoted as $M = \begin{bmatrix} X & Y \end{bmatrix}^T$ and $\tilde{M} = \begin{bmatrix} X & Y & 1 \end{bmatrix}^T$. As a result, the point M and its image projection m can be related by

$$\tilde{m} = H\tilde{M} \tag{2.14}$$

where $H = A\begin{bmatrix} r_1 & r_2 & t \end{bmatrix}$.

Ideally (2.14) should be satisfied for all model and image points. However, in reality this is not the case because of the noise in the extracted image points. Therefore, an estimation should be done for the homography, H.

If this estimation is denoted by $H = \begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix}$, equation (2.14) yields to

$$\begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} = \lambda A\begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \tag{2.15}$$

where λ is an arbitrary scalar.

The orthonormality of r_1 and r_2 brings two constraints:

$$r_1^T r_2 = 0 \tag{2.16}$$

$$r_1^T r_1 = r_2^T r_2 \tag{2.17}$$

where $r_1 = \frac{1}{\lambda} A^{-1} h_1$ and $r_2 = \frac{1}{\lambda} A^{-1} h_2$

Equations (2.11) and (2.12) yield to

$$h_1^T \left(A^{-1}\right)^T A^{-1} h_2 = 0 \tag{2.18}$$

$$h_1^T \left(A^{-1}\right)^T A^{-1} h_1 = h_2^T \left(A^{-1}\right)^T A^{-1} h_2 \tag{2.19}$$

Let us call

$$B = \left(A^{-1}\right)^T A^{-1} \tag{2.20}$$

Then,

$$B = \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{c}{\alpha^2\beta} & -\frac{cv_0 - u_0\beta}{\alpha^2\beta} \\ -\frac{c}{\alpha^2\beta} & \frac{c^2}{\alpha^2\beta} + \frac{1}{\beta^2} & -\frac{c(cv_0 - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} \\ -\frac{cv_0 - u_0\beta}{\alpha^2\beta} & -\frac{c(cv_0 - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} & \frac{(cv_0 - u_0\beta)^2}{\alpha^2\beta^2} + \frac{v_0^2}{\beta^2} + 1 \end{bmatrix} \tag{2.21}$$

B is a symmetric matrix, therefore can be defined by a 6D vector,

$$b = \left[B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}\right]^T \tag{2.22}$$

The i[th] column vector H can be written as $h_i = [h_{i1}, h_{i2}, h_{i3}]^T$. Then the two constraints yields

$$h_i^T B h_j = v_{ij}^T b \tag{2.23}$$

where $v_{ij} = \left[h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}, h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3} \right]^T$

Then, the constraints (2.13) and (2.14) yield us to

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix} b = 0 \tag{2.24}$$

The above equations are for a single projection. If n model planes are observed, (2.19) can be rewritten as

$$Vb = 0 \tag{2.25}$$

where V is a $2n \times 6$ matrix. The solution b can be found using the eigenvector of $V^T V$ associated with the smallest eigenvalue. Once b is found, the intrinsic parameters can be extracted from B as

$$v_0 = \frac{B_{12}B_{13} - B_{11}B_{23}}{B_{11}B_{22} - B_{12}^2} \tag{2.26}$$

$$\lambda = \frac{B_{33} - \left[B_{13}^2 + v_0 \left(B_{12}B_{13} - B_{11}B_{23} \right) \right]}{B_{11}} \tag{2.27}$$

$$\alpha = \sqrt{\lambda / B_{11}} \tag{2.28}$$

$$\beta = \sqrt{\lambda B_{11} / \left(B_{11} B_{22} - B_{12}^{\ 2} \right)} \tag{2.29}$$

$$c = \frac{- B_{12} \alpha^2 \beta}{\lambda} \tag{2.30}$$

$$u_0 = \frac{c v_0}{\alpha} - \frac{B_{13} \alpha^2}{\lambda} \tag{2.31}$$

After A is computed, the extrinsic parameters of the system can be readily determined using (2.14)

$$r_1 = \lambda A^{-1} h_1 \tag{2.32}$$

$$r_2 = \lambda A^{-1} h_2 \tag{2.33}$$

$$r_3 = r_1 \times r_2 \tag{2.34}$$

$$t = \lambda A^{-1} h_3 \tag{2.35}$$

where $\lambda = \frac{1}{\left\| A^{-1} h_1 \right\|} = \frac{1}{\left\| A^{-1} h_2 \right\|}$

Considering the large noise in the images of model planes, the calculated image point coordinates are erroneous. Therefore, they can be refined using the maximum likelihood estimation. The following functional can be minimized to obtain the maximum likelihood estimate.

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \left\| m_{ij} - \hat{m}\left(A, R_i, t_i, M_j \right) \right\|^2 \tag{2.36}$$

where $\hat{m}\left(A, R_i, t_i, M_j\right)$ is the calculated projection of point M_j in image i according to equation (2.14).

Among with intrinsic and extrinsic parameters, lens distortions can be found with camera calibration. Lens distortions are due to imperfections of lens introduced during manufacturing. These distortions affect the calculations especially in applications where high accuracy is needed. Lens distortions can be modeled in two types; *radial distortions* and *tangential distortions*. Radial distortions cause the actual image point to be relocated radially on the image plane. Modeling lens distortions radially is usually sufficient; however to be more precise decentering distortion, which has a radial and tangential component, should be taken into account. The radial distortion can be expressed as

$$\delta x^{(r)} = x_c \left(k_1 r^2 + k_2 r^4 + k_3 r^6 + ...\right) \tag{2.37}$$

$$\delta y^{(r)} = y_c \left(k_1 r^2 + k_2 r^4 + k_3 r^6 + ...\right) \tag{2.38}$$

where k_1, k_2, k_3, … are radial distortion coefficients and $r = \sqrt{x_c{}^2 + y_c{}^2}$. Usually first two coefficients are sufficient for correction. The tangential distortion expression can be written as

$$\delta x^{(t)} = 2 p_1 x_c y_c + p_2 (r^2 + 2 x_c{}^2) \tag{2.39}$$

$$\delta y_c{}^{(t)} = p_1 (r^2 + 2 y_c{}^2) + 2 p_2 x_c y_c \tag{2.40}$$

where p_1 and p_2 are tangential distortion coefficients and $r = \sqrt{x_c{}^2 + y_c{}^2}$.

After applying distortion correction, the corresponding camera reference frame based coordinates of the image point, (x_c', y_c') become

$$x_c' = x_c + \delta x^{(r)} + \delta x^{(t)} \tag{2.41}$$

$$y_c' = y_c + \delta y^{(r)} + \delta y^{(t)} \tag{2.42}$$

When lens distortions are taken into account equation (2.11) should be updated with (2.41) and (2.42). Note that the toolbox uses [26] to estimate the distortion coefficients. However, in this thesis distortions are calculated by a different method which will be explained in the next chapter. The calibration images are first corrected by our method and fed into the toolbox. Therefore, the details of [26] will not be explained here.

2.2.2 The Inverse Perspective Transformation

Stereo vision aims to deduce the 3D information of a scene based on two or more images. The necessary inputs of such a problem are the system intrinsic and extrinsic parameters extracted from camera calibration. Also, the correspondence problem, which aims to determine the projections of the same scene point on different camera planes, needs to be solved. In this section, it is assumed that the correspondence problem had been solved and conjugate pairs had been determined.

Assume a two camera based vision system in arbitrary positions with respect to each other. These cameras will be labeled as Camera 1 and Camera 2. Let the 3D world coordinates of a point, P, seen by both

cameras be $M = [X, Y, Z]^T$. The projection of this point on Camera 1 is $\tilde{m}^{(1)} = A^{(1)} \begin{bmatrix} R^{(1)} & t^{(1)} \end{bmatrix} \tilde{M}$. Similarly, the projection of P on Camera 2 turns out to be $\tilde{m}^{(2)} = A^{(2)} \begin{bmatrix} R^{(2)} & t^{(2)} \end{bmatrix} \tilde{M}$. Note that $A^{(1)}$, $R^{(1)}$ and $t^{(1)}$ are the intrinsic and extrinsic matrices corresponding to the first camera and $A^{(2)}$, $R^{(2)}$ and $t^{(2)}$ are the intrinsic and extrinsic matrices corresponding to the second camera. All the rotation and translation matrices are determined with respect to the origin of the world reference frame.

The generic projection relations [27] can be rewritten as follows

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & 0 \\ A_{21} & A_{22} & A_{23} & 0 \\ A_{31} & A_{32} & A_{33} & 0 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.43}$$

These relations can be expressed in a much compressed form, using a transformation matrix, C,

$$\tilde{m} = C\tilde{M} \tag{2.44}$$

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} \\ C_{21} & C_{22} & C_{23} & C_{24} \\ C_{31} & C_{32} & C_{33} & C_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.45}$$

Previously, in Section 2.2.1.1 it was stated that

$$u - Uw = 0 \tag{2.46}$$

$$v - Vw = 0 \tag{2.47}$$

Combining (2.45) with (2.46) and (2.47)

$$\left(C_{11} - UC_{31}\right)X + \left(C_{12} - UC_{32}\right)Y + \left(C_{13} - UC_{33}\right)Z + \left(C_{14} - UC_{34}\right) = 0 \tag{2.48}$$

$$\left(C_{21} - VC_{31}\right)X + \left(C_{22} - VC_{32}\right)Y + \left(C_{23} - VC_{33}\right)Z + \left(C_{24} - VC_{34}\right) = 0 \tag{2.49}$$

Equations (2.48) and (2.49) can be written in the following form for the two cameras

$$a_1X + b_1Y + c_1Z + d_1 = 0 \tag{2.50}$$

$$a_2X + b_2Y + c_2Z + d_2 = 0 \tag{2.51}$$

$$m_1X + n_1Y + p_1Z + q_1 = 0 \tag{2.52}$$

$$m_2X + n_2Y + p_2Z + q_2 = 0 \tag{2.53}$$

with

$$a_1 = C_{11}^{(1)} - U^{(1)}C_{31}^{(1)} \tag{2.54}$$

$$b_1 = C_{12}^{(1)} - U^{(1)}C_{32}^{(1)} \tag{2.55}$$

$$c_1 = C_{13}^{(1)} - U^{(1)}C_{33}^{(1)} \tag{2.56}$$

$$d_1 = C_{14}^{(1)} - U^{(1)}C_{34}^{(1)} \tag{2.57}$$

$$a_2 = C_{21}^{(1)} - V^{(1)}C_{31}^{(1)} \tag{2.58}$$

$$b_2 = C_{22}^{(1)} - V^{(1)}C_{32}^{(1)} \tag{2.59}$$

$$c_2 = C_{23}^{(1)} - V^{(1)}C_{33}^{(1)} \tag{2.60}$$

$$d_2 = C_{24}^{(1)} - V^{(1)}C_{24}^{(1)} \tag{2.61}$$

$$m_1 = C_{11}^{(2)} - U^{(2)}C_{31}^{(2)} \tag{2.62}$$

$$n_1 = C_{12}^{(2)} - U^{(2)} C_{32}^{(2)} \tag{2.63}$$

$$p_1 = C_{13}^{(2)} - U^{(2)} C_{33}^{(2)} \tag{2.64}$$

$$q_1 = C_{14}^{(2)} - U^{(2)} C_{34}^{(2)} \tag{2.65}$$

$$m_2 = C_{21}^{(2)} - V^{(2)} C_{31}^{(2)} \tag{2.66}$$

$$n_2 = C_{22}^{(2)} - V^{(2)} C_{32}^{(2)} \tag{2.67}$$

$$p_2 = C_{23}^{(2)} - V^{(2)} C_{33}^{(2)} \tag{2.68}$$

$$q_2 = C_{24}^{(2)} - V^{(2)} C_{34}^{(2)} \tag{2.69}$$

Writing in matrix form

$$\begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ m_1 & n_1 & p_1 \\ m_2 & n_2 & p_2 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} -d_1 \\ -d_2 \\ -q_1 \\ -q_2 \end{bmatrix} \tag{2.70}$$

Four equations, (2.50), (2.51), (2.52) and (2.53), with 3 unknowns (X, Y, Z) makes the system overdetermined. To find a least-squares error solution the pseudo-inverse technique should be utilized. Consequently, the 3D world coordinates turns out to be

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \left(\begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ m_1 & n_1 & p_1 \\ m_2 & n_2 & p_2 \end{bmatrix}^T \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ m_1 & n_1 & p_1 \\ m_2 & n_2 & p_2 \end{bmatrix} \right)^{-1} \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ m_1 & n_1 & p_1 \\ m_2 & n_2 & p_2 \end{bmatrix}^T \begin{bmatrix} -d_1 \\ -d_2 \\ -q_1 \\ -q_2 \end{bmatrix} \tag{2.71}$$

CHAPTER 3

IMAGE PROCESSING ALGORITHMS

Before computing the 3D coordinates of an object, the image should be arranged such that the object should be recognized individually. The processing of the image by certain algorithms and extracting specific attributes of it is called *image processing*. In this chapter the fundamentals of digital image processing theory that are used in this study are explained in detail.

3.1 Image Representation Basics

Image is a two-dimensional function, $f(x, y)$, representing the intensities of each pixel at coordinates (x, y) [28]. For monochrome images, the term gray-level is used for image intensity. Color images usually consist of three individual gray-level images; therefore, those operations that can be utilized for gray-level images can be extended to color images by processing each individual image.

An image may be continuous with respect to x and y axes and its amplitude. A digital image is formed by quantizing a continuous image at sampled pixels.

3.1.1 Coordinate System Representation

The pixel coordinate system is shown in Figure 3.1. Note that the x and y axes are the rows and columns of the image array. An $M \times N$ image has M pixels on its x axis and N pixels on the y axis.

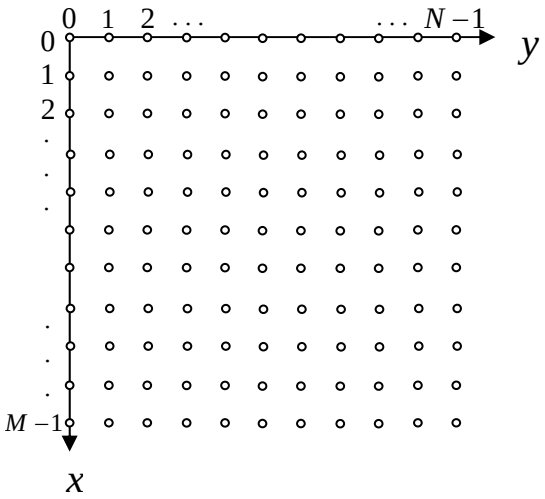


Figure 3.1 Coordinate representation in image processing

3.1.2 Matrix Representation of Images

It has been mentioned that images are two-dimensional arrays in essence. Therefore, they can be represented as two-dimensional $M \times N$ matrices, too.

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \dots & \dots & \dots & \dots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix} \quad (3.1)$$

$f(x,y)$ represents the image intensity of the pixel located at (x,y).

3.2 Morphological Image Processing

Morphological image processing is used to extract various image components by using basics of set theory. These operations are usually applied on binary images in which it is easier to extract the properties of the desired shape. Morphological processes range from basic logical operations to more complex reconstruction algorithms among which only those that are utilized in this thesis will be given. These are used for feature extraction purposes.

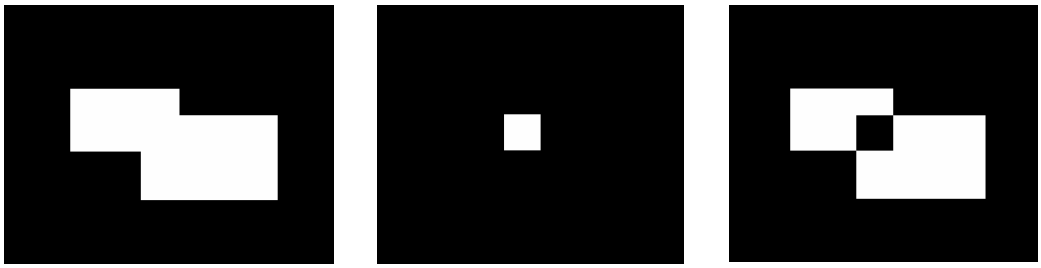
3.2.1 Logical Operations

Of the basic morphological operations, logical operations establish one of the most basic and commonly used processes. These operations are based on basic set theory applications; i.e. union, difference, intersection etc. In Figure 3.2, two binary images in (a) and (b) are shown.

(a) (b)

Figure 3.2 (a) First binary image (b) Second binary image

Figure 3.3 shows the results of various logical operations. In (a) the union of two images are displayed; all foreground pixels are shown from both. In (b) the intersection is shown in which only overlapping foreground pixels are displayed and in (c) symmetric difference is shown where all foreground pixels excluding the overlapping pixels are displayed.



(a) (b) (c)

Figure 3.3 (a) Union of two images (b) Intersection of two images

(c) Symmetric difference of two images

3.2.2 Dilation and Erosion

Dilation and erosion are two fundamental operations of image processing. They are used for thickening or thinning the object in a binary image. A *structuring element* is used for performing these operations.

Dilation is used for growing or thickening an object by means of a structuring element. This element determines the extent of thickening. Figure 3.4 shows such a structuring element with its origin marked in a black border.

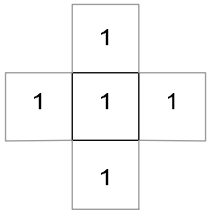


Figure 3.4 Structuring element

Using the structuring element in Figure 3.4, the following output image can be obtained (Figure 3.5). The origin is moved among the image and those 1-valued pixels that overlap with the origin are sought. When these are found the corresponding neighbors of those pixels are replaced with the value of the corresponding pixel of the structuring element.

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

(a)

0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0

(b)

Figure 3.5 (a) Original image with a square object (b) Its dilated form

Erosion is essentially the reverse of dilation where the image objects are thinned. The structuring element again controls the amount and extent of

thinning. The overlapping of the structuring element with the entire foreground of image is sought and only the pixels overlapping with the origin is kept. For instance, if a structuring element as in Figure 3.6 is used on an original image in Figure 3.7; an output image like the one in the same figure can be obtained.

Figure 3.6 Structuring element for erosion

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(a)

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(b)

Figure 3.7 (a) Original image (b) Its eroded output

Erosion and dilation are often used together to obtain better results. For instance, applying first erosion then dilation is called *opening*. Opening is mostly used to remove noise and thin connections and smooth the contours of the object. The reverse process, applying erosion after dilation is called *closing*. Like opening, closing smoothes object contours and contrarily fills holes and joins narrow breaks. Opening and closing are

used extensively in this thesis, especially to remove the unwanted background noise during image filtering.

3.2.3 Connected Component Labeling

Connected component labeling is used literally to "count" the objects in an image. The term connected components stands for the set of all foreground pixels connected to any foreground pixel, p located at the coordinates (x, y). The two horizontal and two vertical neighbors of p are pixels $(x+1, y)$, $(x-1, y)$, $(x, y+1)$ and $(x, y-1)$. These are called *4-neighbors* of p. The diagonal neighbors of p are $(x+1, y+1)$, $(x+1, y-1)$, $(x-1, y+1)$ and $(x-1, y-1)$. Along with the horizontal and vertical neighbors these neighbors are called *8-neighbors* of p (Figure 3.8). If a path between the 4-neighbors of p is defined, then the term *4-connected* can be used. Similarly, a path between 8-neighbors infers *8-connectivity*.

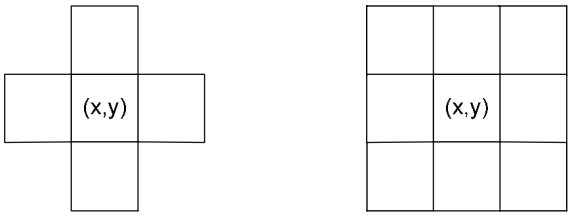
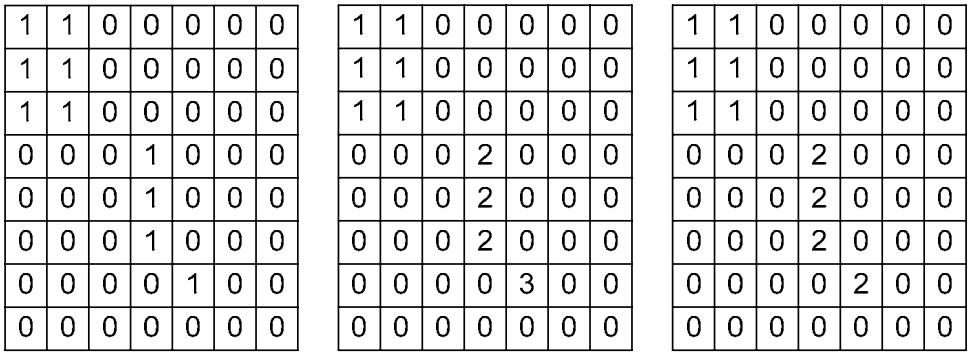


Figure 3.8 4-neighbors and 8-neighbors of p

An example of connected component labeling can be seen in Figure 3.9 for 4-connectivity and 8-connectivity. Using 8-connectivity reduces the number of objects (Figure 3.9). Therefore, it is very important how the connectivity is chosen. Connected-component labeling is a vital step for

centroid calculation algorithms of multiple points. Identification of each point is achieved through connected-component labeling. 8-connectivity is used since the points that are going to be extracted are not very close to each other.



(a) (b) (c)

Figure 3.9 (a) Original image (b) 4-connected components (c) 8-connected components

3.3 Image Segmentation

Image segmentation is used to distinguish the objects in the image from the background. The extent to which this distinguishing is carried out is determined by the aim of the problem. In this section only edge detection and thresholding will be discussed among various segmentation methods.

3.3.1 Edge Detection

Edge detection uses the discontinuities in image intensities. These discontinuities are detected using the first and second derivatives. The

first derivative, gradient, of any function, $f(x, y)$ can be defined as the vector;

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{3.2}$$

with magnitude

$$\nabla f = \left[G_x{}^2 + G_y{}^2 \right]^{1/2} \tag{3.3}$$

and orientation

$$\alpha(x, y) = \tan^{-1}\left(\frac{G_y}{G_x} \right) \tag{3.4}$$

The second derivative, namely the Laplacian is defined as

$$\nabla^2 f(x, y) = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2} \tag{3.5}$$

In practical applications, the Laplacian is rarely used due to its high susceptibility to noise, the fact that its magnitude produces double edges and its disability to detect edge direction. Usually two approaches are used for edge detection:

1. Find the image locations where the magnitude of the first derivative of intensity is greater than a certain threshold.

2. Find the image locations where the second derivative of intensity crosses zero axis.

Various edge detection algorithms do exist. Among these methods Sobel, Prewitt and Canny algorithms will be mentioned.

3.3.1.1 Sobel Edge Detector

Sobel edge detector uses (3.6), based on equation (3.3), which corresponds to the masks shown in Figure 3.11. Figure 3.10 shows the pixel designations.

$$G = \left\{ \begin{array}{l} \left[\left(p_7 + 2p_8 + p_9 \right) - \left(p_1 + 2p_2 + p_3 \right) \right]^2 \\ + \left[\left(p_3 + 2p_6 + p_9 \right) - \left(p_1 + 2p_4 + p_7 \right) \right]^2 \end{array} \right\}^{1/2} \tag{3.6}$$

p_1	p_2	p_3
p_4	p_5	p_6
p_7	p_8	p_9

Figure 3.10 Pixel designations

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Figure 3.11 Sobel filter masks for x and y axes

3.3.1.2 Prewitt Edge Detector

Prewitt edge detector utilizes the masks in Figure 3.12 to generate G_x and G_y. Although the masks are quite similar to that of Sobel detector, Prewitt detector produces noisier results.

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Figure 3.12 Prewitt edge detector masks

3.3.1.3 Canny Edge Detector

Canny edge detector is used in the scope of this thesis due to the fact that it is the most powerful edge detection method provided by Matlab's *edge* function. The method can be explained in 4 basic steps:

1. The image is smoothed using a Gaussian filter with a certain standard deviation to eliminate noise.
2. The gradient at each point; i.e. $g(x,y) = \left[G_x^{\,2} + G_y^{\,2} \right]^{1/2}$ and orientation $\alpha(x,y) = \tan^{-1}\left(G_y / G_x \right)$ are determined. Note that any of the aforementioned techniques can be utilized to compute G_x and G_y.The edge point's strength is locally maximum in the direction of the gradient.

3. The ridges produced by the edge points found in step 2 are tracked by a special algorithm that outputs a thin line as the edge. These are then put under a threshold and the edge linking is done according to the outputs of the thresholding.

In the thesis Canny edge detector was used to extract the points on the face along with thresholding methods. This powerful method gave perfect solutions for point extraction.

3.3.2 Thresholding

Thresholding holds an essential position in image segmentation applications. It is mainly used for identifying objects with a certain intensity value. The threshold value is usually chosen by inspecting the *histogram* of the image. The thresholded image $g(x,y)$ can be defined out of the original image $f(x,y)$ as

$$g(x,y)=\begin{cases}1 \text{ if } f(x,y)\geq T\\ 0 \text{ if } f(x,y)<T\end{cases} \tag{3.7}$$

In equation (3.7) an example of *global thresholding* is shown, where T is constant. In cases where the illumination is imbalanced; *local thresholding* can be applied and the threshold can be chosen to be a locally varying function $T(x,y)$ where

$$T(x,y)=f_o(x,y)+T_o \tag{3.8}$$

In this study global thresholding in RGB and HSV color spaces was used to extract the green points on the face. The color green was chosen specially for marker color because its hue value is relatively less found on human face.

CHAPTER 4

3D POINT RECONSTRUCTION

In this chapter the experimental setup is explained in accordance with the various procedures for distortion correction and camera calibration. The distortion correction method presented in this chapter introduces a novel way which is to correct the images before the camera calibration procedure. In the final section of the chapter the 3D reconstruction algorithm is explained on the example of a grid.

4.1 Experimental Setup

The experimental setup is able to track the 3D coordinates of an object moved within its workspace (Figure 4.1). The setup consists of

- 2 CMOS 640 x 480 resolution webcams
- A setup frame covering a volume of 60 cm x 60 cm x 40 cm
- A Personal Computer (PC)

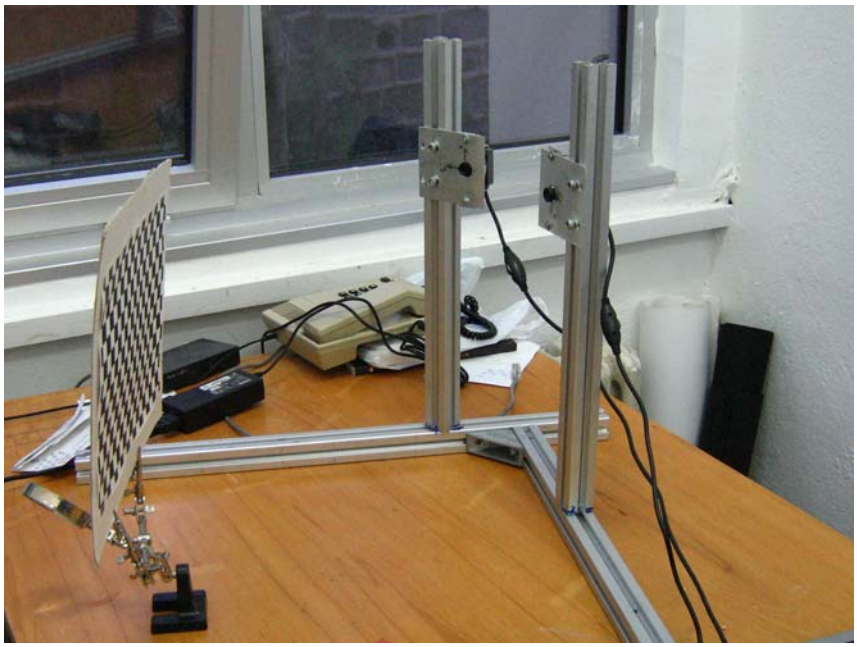


Figure 4.1 The experimental setup

The setup frame is built using 30 mm x 30 mm aluminum profiles. The frame is easily adjustable for different configurations. The setup allows four degrees of freedom for the position and orientation of each camera. Therefore, any stereo vision problem can be solved by using this easily configurable setup.

The two CMOS webcams have a resolution of 640 x 480 pixels and can capture 30 fps videos in this resolution. Their focal lengths can be adjusted manually. No external light sources are used and the image processing algorithms are devised such that day light produces acceptable results.

The webcams are connected to the PC via USB. The images are acquired by using a commercially available video capture software. Once the

videos are captured, a special algorithm coded in MATLAB extracts image frames from these videos. The synchronization of the videos are achieved using an external light-emitting-diode (LED) array circuit. This circuit consists of 8 LEDs that are lit up by a microcontroller (PIC) circuit. The software makes use of these LEDs in order to synchronize the images frames captured by different webcams.

Before starting 3D point reconstruction a vital step is the camera calibration. Calibration helps to determine the intrinsic and extrinsic parameters of a camera. Note that at first the individual intrinsic and extrinsic parameters are determined after which the position and orientation of the right camera with respect to the left camera is computed. The calibration procedure is carried out by the MATLAB toolbox written by Jean-Yves Bouguet of California Institute of Technology [29]. A calibration grid is shown to the cameras and the corners of the grid constitute the necessary calibration points. The position and orientation of the grid is changed in every shot. Figure 4.2 and Figure 4.3 show sample calibration images for right and left cameras, respectively.

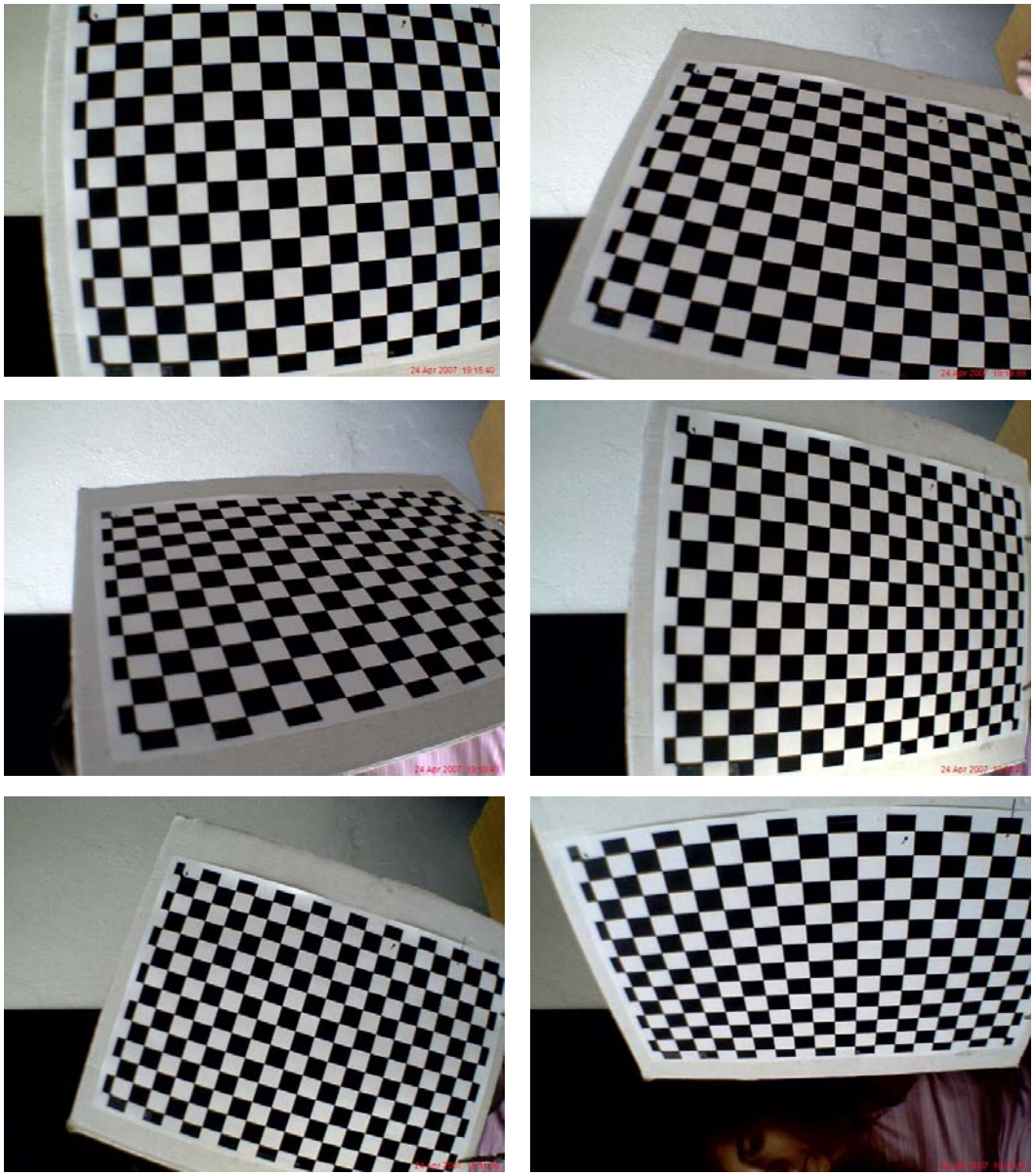


Figure 4.2 Calibration images for right camera

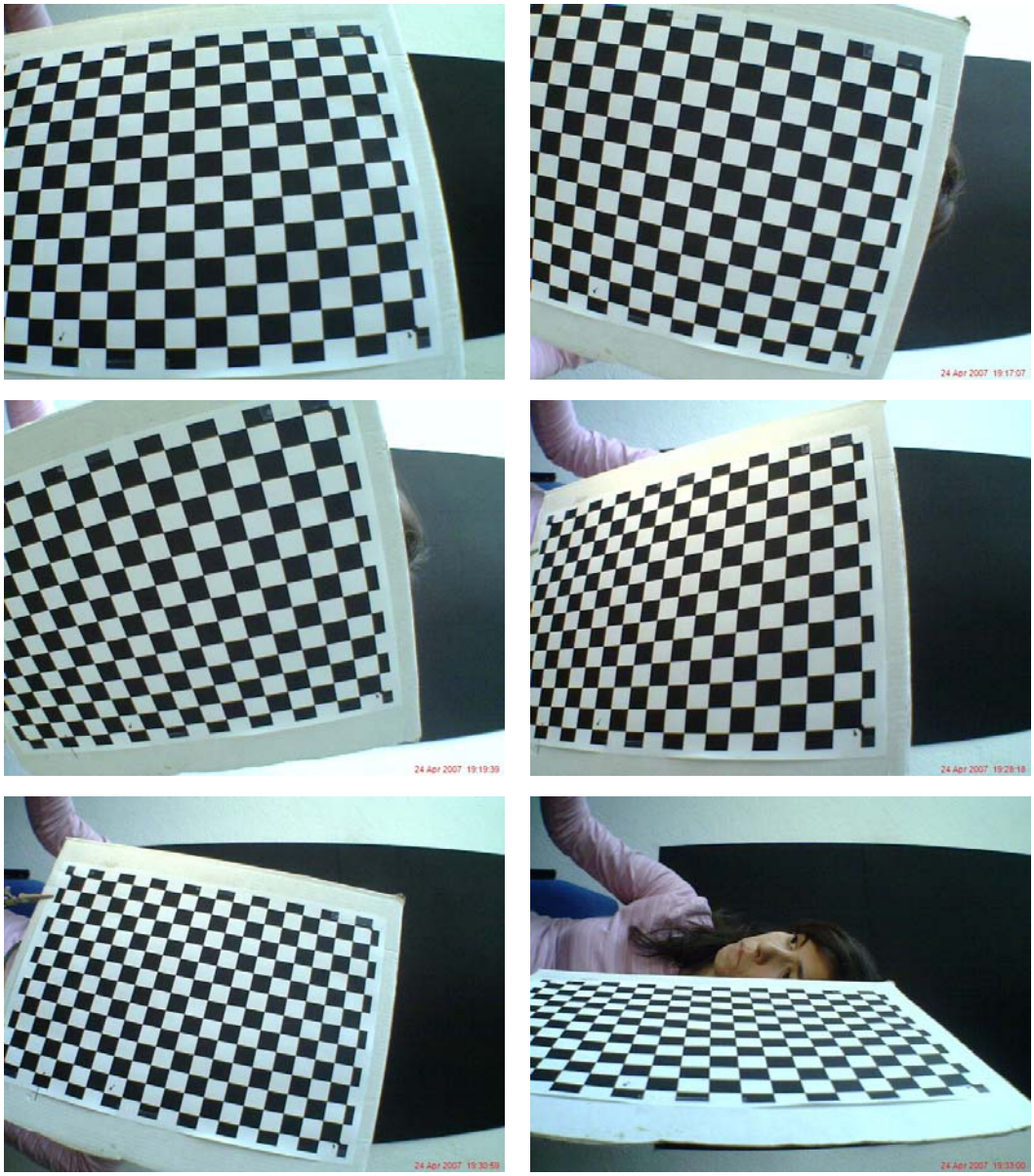


Figure 4.3 Calibration images for left camera

4.2 Distortion Correction

An important factor affecting the 3D coordinate calculation is the lens distortions. As explained in Section 2.2.1.1, lens distortions can be classified in two groups; radial and tangential distortions. Radial and tangential distortions are expressed in terms of a set of coefficients that

form the power series solution to the distortion problem. There exist many ways to determine these coefficients. In this thesis a different way of distortion correction is adopted without calculating the distortion coefficients of radial and tangential distortion individually.

Distortion can be simply assumed as the displacement of the pixels from their true positions to new distorted positions. Therefore, if this displacement for each pixel can be found, the calculation of the corrected positions is trivial. To determine the pixel displacements, a grid composed of 44x59 points (2596 points totally) is shown to each camera (Figure 4.4). The aim of this grid is to compute the displacements of the points on the distorted grid with respect to an original undistorted grid and fit these displacements for all the pixels of the image.

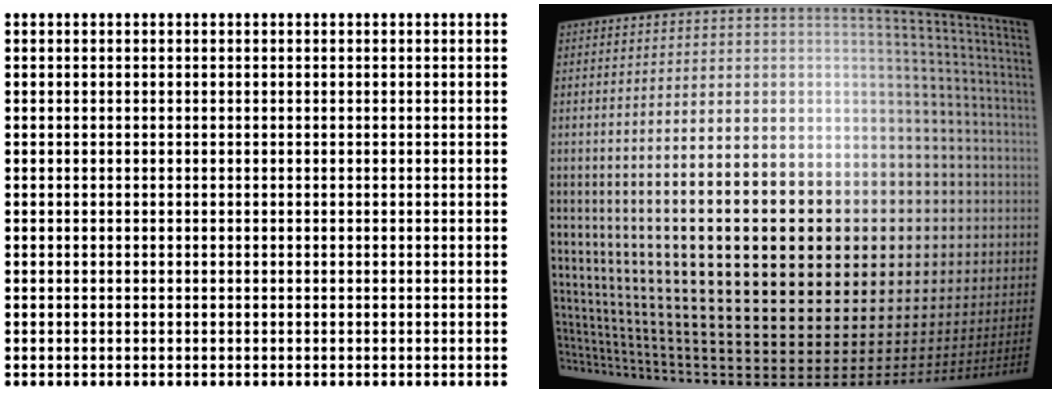


Figure 4.4 Original grid and the grid viewed by the right camera

To calculate the distortion errors, the centroids of each point is calculated using connected component labeling at first and then a special algorithm that uses the division of the first moment of inertia to the overall area is used. Then, the points are relabeled such that upper leftmost point is

counted as the first point. Assuming that the distortion is minimum on the regions closer to the image center, two horizontal central points next to each other are picked manually. The algorithm aligns the undistorted grid to the distorted grid using these two points. Certain scaling, rotation and translation operations between these points are carried on the undistorted image with respect to the distorted image. If any point on the original (undistorted) grid is labeled as p_u and the corresponding point on the viewed (distorted) grid is labeled as p_d, then these central points can be designated as p_{uc1}, p_{uc2} for the original grid and p_{dc1}, p_{dc2} for the distorted grid.

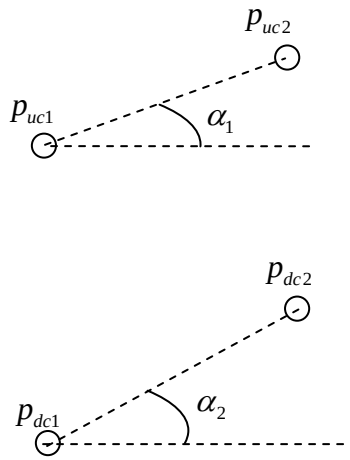


Figure 4.5 Example designation of the central points

The alignment operations described above can be formulated as follows

$$d_u^{\,2} = \left[x(p_{uc2}) - x(p_{uc1})\right]^2 + \left[y(p_{uc2}) - y(p_{uc1})\right]^2 \tag{4.1}$$

$$d_d^{\,2} = \left[x(p_{dc2}) - x(p_{dc1})\right]^2 + \left[y(p_{dc2}) - y(p_{dc1})\right]^2 \tag{4.2}$$

The scaling factor, s , is found and the original point coordinates are corrected by s.

$$s = \frac{d_d}{d_u} \tag{4.3}$$

$$p_u = s \cdot p_u \tag{4.4}$$

Following scaling, rotation is performed by rotating the all the points of the original grid by θ

$$\theta = \alpha_2 - \alpha_1 \tag{4.5}$$

where

$$\alpha_2 = \arctan 2\left[y\left(p_{dc2}\right) - y\left(p_{dc1}\right), x\left(p_{dc2}\right) - x\left(p_{dc1}\right) \right] \tag{4.6}$$

$$\alpha_1 = \arctan 2\left[y\left(p_{uc2}\right) - y\left(p_{uc1}\right), x\left(p_{uc2}\right) - x\left(p_{uc1}\right) \right] \tag{4.7}$$

Then, the rotated locations of the original grid points can be found as

$$\phi = \arctan 2\left[y\left(p_u\right) - y\left(p_{uc1}\right), x\left(p_u\right) - x\left(p_{uc1}\right) \right] \tag{4.8}$$

$$r = \left\{ \left[x\left(p_u\right) - x\left(p_{uc1}\right) \right]^2 + \left[y\left(p_u\right) - y\left(p_{uc1}\right) \right]^2 \right\}^{1/2} \tag{4.9}$$

$$x\left(p_u\right) = r\cos\left(\theta + \phi\right) \tag{4.10}$$

$$y\left(p_u\right) = r\sin\left(\theta + \phi\right) \tag{4.11}$$

Note that by aligning these points it is assumed that the original grid is centered with respect to the distorted grid and the correction is done according to the distorted grid. The reason for this is to prevent any errors

that could form while correcting the distorted grid. Any rotations or transformed scenes due to the camera's position and orientation should remain on the distorted grid so that they can be calculated during calibration.

Following rotation translation can be performed using the following equations

$$d_1 = x(p_{dc1}) - x(p_{uc1}) \tag{4.12}$$

$$d_2 = y(p_{dc1}) - y(p_{uc1}) \tag{4.13}$$

$$x(p_u) = x(p_u) + d_1 \tag{4.14}$$

$$y(p_u) = y(p_u) + d_2 \tag{4.15}$$

After the original grid and the distorted grid are aligned, the difference in the x and y coordinates of the distorted grid points with respect to the original grid points is taken. These 2596 differences, which will be referred to as the errors, are used to fit a surface covering the whole image plane. This surface is generated for x and y coordinates separately. Hence, for every pixel of the image a correction value can be found. These values form a 640 x 480 correction matrix which is used before every calibration procedure. Therefore, during calibration only the intrinsic and extrinsic matrices are calculated and distortion parameters are not computed. The error plots for x and y coordinates can be seen in Figure 4.6 and Figure 4.7.

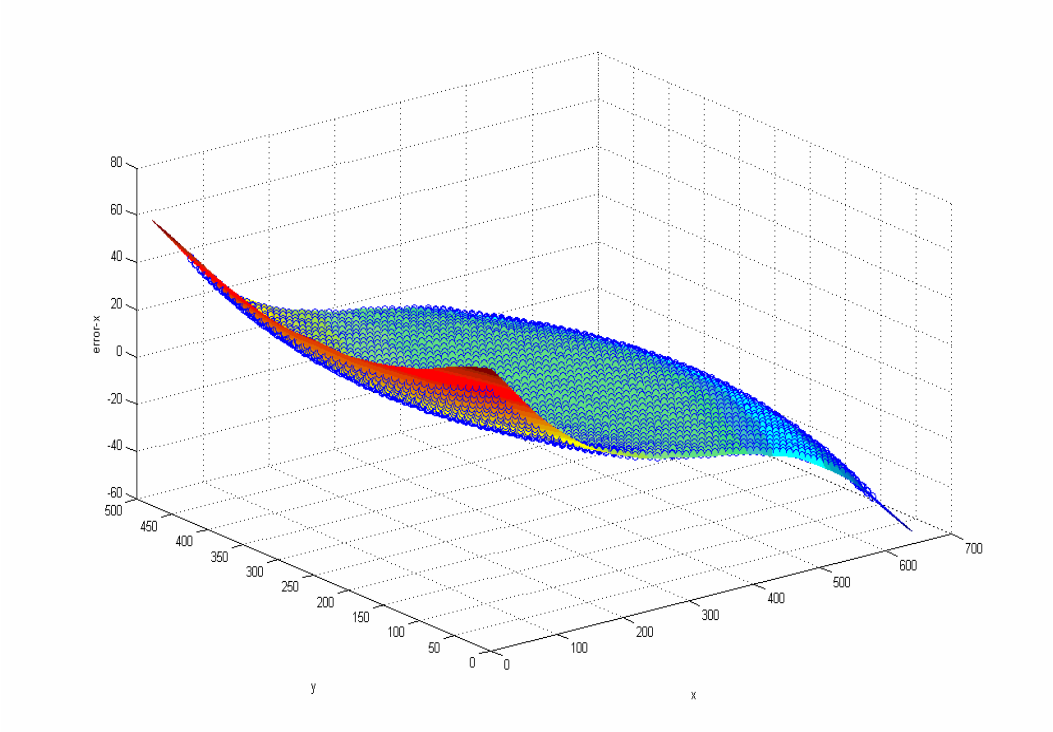


Figure 4.6 Distortion error in x-axis

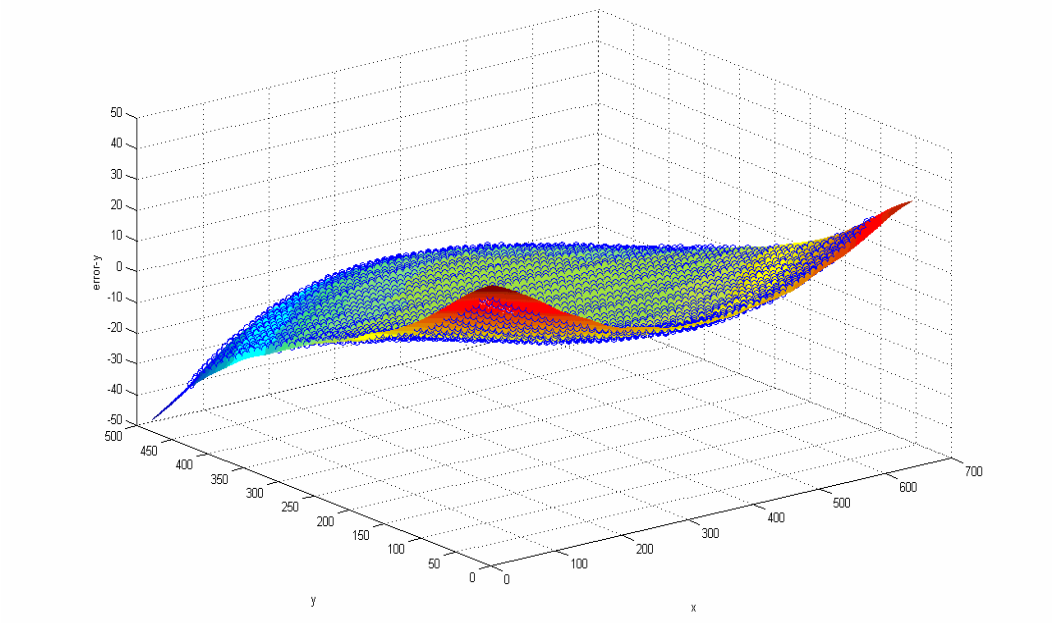


Figure 4.7 Distortion error in y-axis

The original image and the corrected image can be seen in Figure 4.8. This method presents an advantage while 3D point reconstruction. In most of the calibration procedures, coefficients for radial and tangential distortion are calculated. The correction is done at a stage between the perspective projection and 3D reconstruction. Therefore, another step is included to the middle of the reconstruction process. Moreover, the high degree distortion models complicate the solution of the correction problem. In our case, the images are first corrected and then the camera calibration is executed. Hence, the overall 3D reconstruction can be simplified into a single matrix as in (2.45).

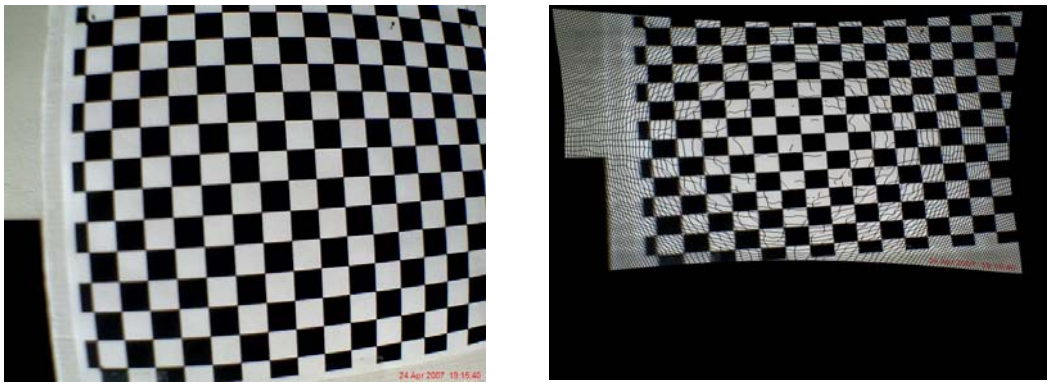


Figure 4.8 Original distorted image and its corrected form

4.3 3D Reconstruction

To check the accuracy of the system a sample grid is captured by the cameras (Figure 4.9).

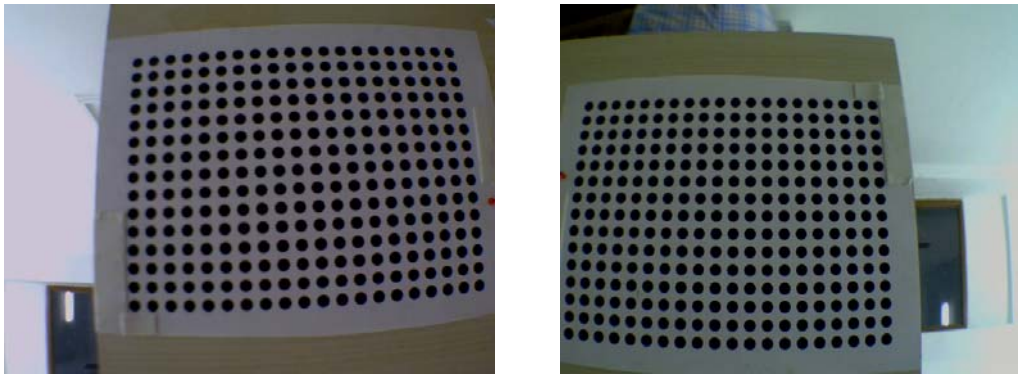


Figure 4.9 Grid images taken from the right and left cameras

The images captured by the cameras are first converted to binary image by a certain threshold and dilation and erosion are conducted so as to produce a noiseless image. After this filtering they are corrected (Figure 4.10).

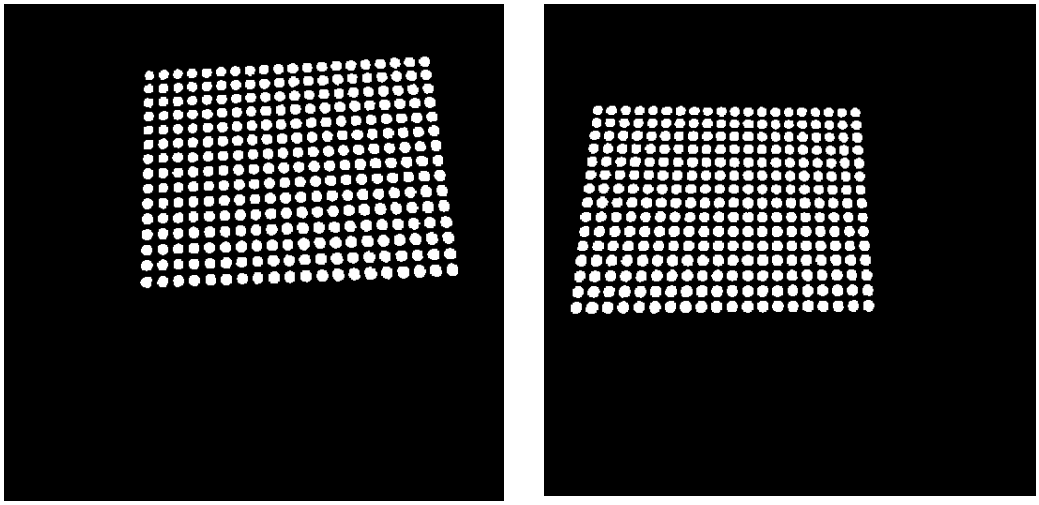


Figure 4.10 Filtered and corrected images

Subsequently, the points are labeled utilizing connected component labeling and their centroids are calculated. Since connected component labeling algorithm starts to label the points from the leftmost point; a special algorithm is written to sort the labels of the points so as to label

the points from the upper left point. This algorithm enables a standard of labeling to be established and is also used in face tracking.

After the proper ordering of points, it only remains to calculate the 3D world coordinates of every point with respect to the origin. These coordinates are calculated using the inverse perspective transformation principles explained in Section 2.2.2. Figure 4.11 shows the 3D reconstruction of the points plotted on the Cartesian coordinates.

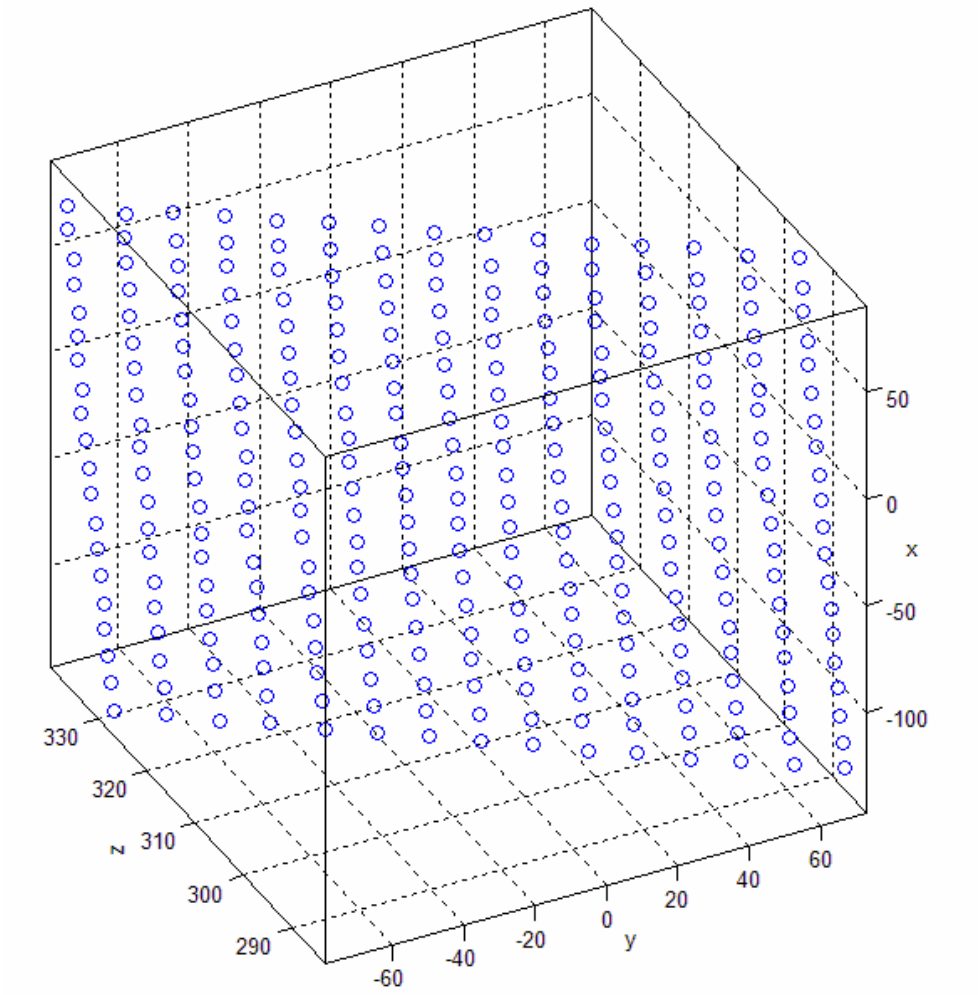


Figure 4.11 Calculated 3D coordinates

4.4 Error Analysis

To calculate the accuracy of the system, an error analysis is carried out. A best plane is fit to the calculated 3D coordinates of the grid using orthogonal linear regression. (Figure 4.12) In this method, the sum of the perpendicular distances of the points to the plane is calculated. As a result, total sum of orthogonal distances to the plane turned out to be 89.1236 mm and the average distance can be calculated approximately as 0.3 mm.

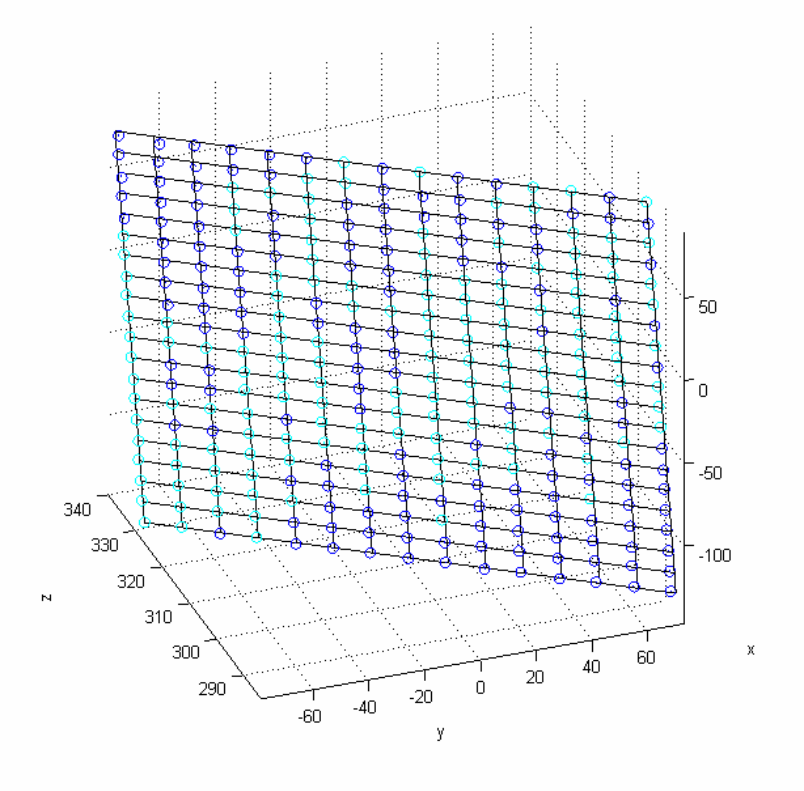


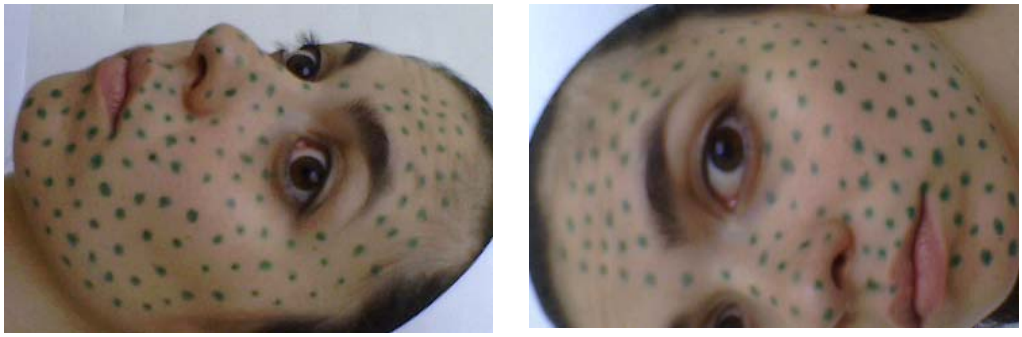
Figure 4.12 Best plane using orthogonal linear regression

In order to calculate a scale factor to the algorithm, various actual and calculated distances were extracted. The actual distance between each point on the grid both in horizontal and vertical directions is 10.3 mm.

The actual corner-to-corner distance in vertical direction is 195.8 mm and the actual corner-to-corner distance in horizontal direction is 144.2 mm. The calculated versions of all these distances are 10.98 mm, 211.73 mm and 155.17 mm, respectively. Note that these are average calculated distances. Subsequently, the scale factor for point distance can be found as 1.066, for corner-to-corner distance in vertical direction is 1.081 and for horizontal direction is 1.076. An average scale factor can be found as 1.074. If the calculated distances are corrected by using this scale factor, the point distance turns out to be 10.227 mm, the corner-to-corner distance in vertical direction is 197.165 mm and the corner-to-corner distance in horizontal direction is 144.510 mm. Note that these values are very close to the actual ones and the remaining errors are mainly due to the round-off errors in distortion correction.

4.5 Face Reconstruction

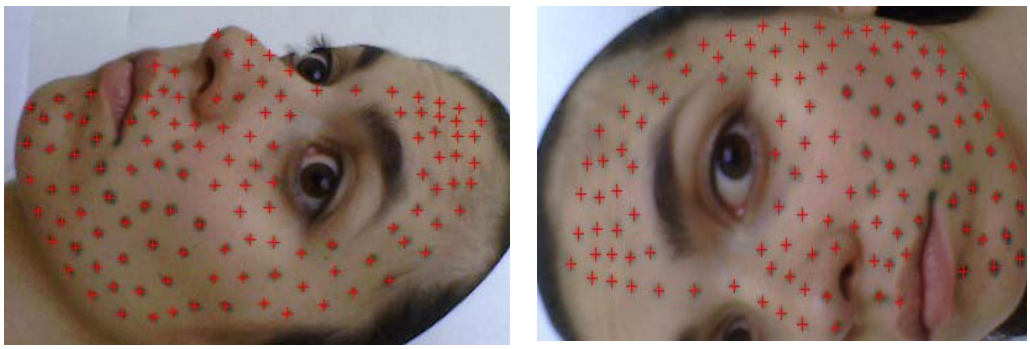
Face reconstruction algorithm works very similarly to the grid reconstruction algorithm described in the previous section. The face is assumed to be symmetric with respect to the nose-jaw line; therefore, only half of the face was marked with points (Figure 4.13).



(a) (b)

Figure 4.13 Marked face seen from the (a) left and (b) right camera

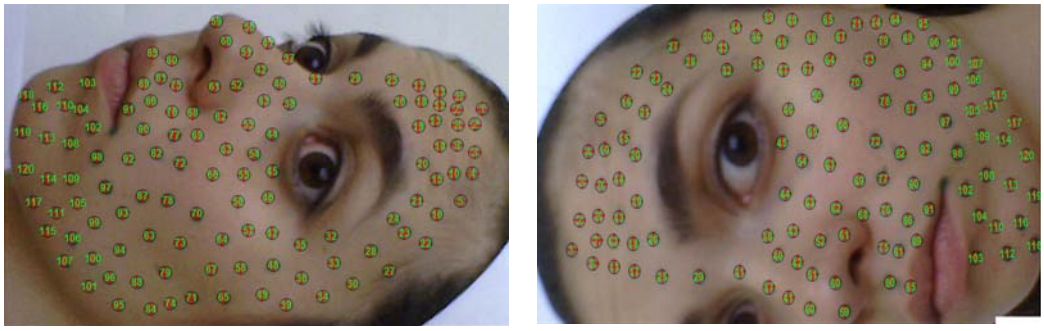
The marking color of the face was selected such that the hue values of the points make it easier to threshold the image. The filtered image of the markers is fed into the centroid finding algorithm described in the previous sections (Figure 4.14).



(a) (b)

Figure 4.14 Images from the (a) left and (b) right camera after the centroids are found

After the centroids are found in the two images, the points should be related to each other so that the 3D reconstruction algorithm can identify the same point in the two images. This correspondence, which was done manually in this case, is shown in Figure 4.15.



(a) (b)

Figure 4.15 Image from the (a) left and (b) right camera after labeling is done

Following the relation of the points in left and right cameras there only remains to compute the 3D coordinates of the centroids of the marker points. The centroids of the images were fit on a rectangular surface mesh and the result can be seen in Figure 4.16. The outline of the face can be seen in this image.

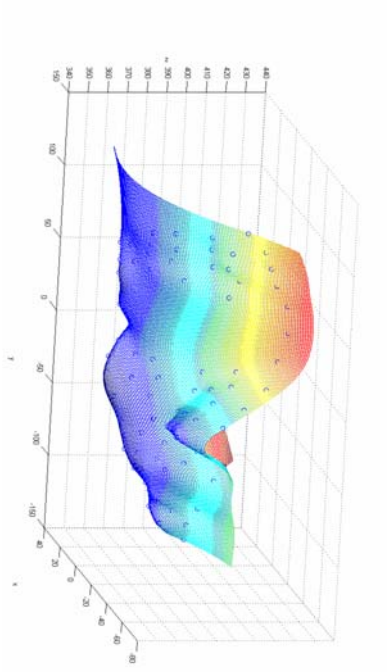


Figure 4.16 Reconstructed face in MATLAB - front view

Also, the marker coordinates were fed into Kubotek Keycreator CAD program to get an outline of the face as in Figure 4.17 and Figure 4.18.

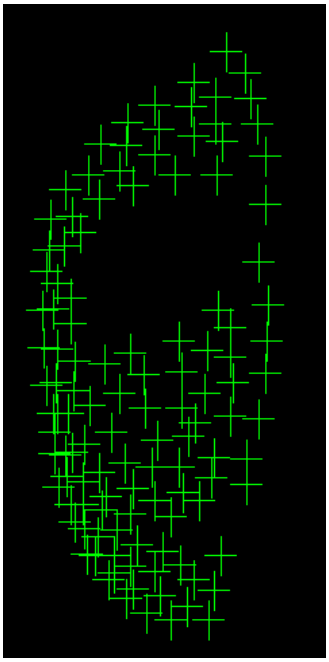


Figure 4.17 Reconstruction made in Kubotek Keycreator-front view

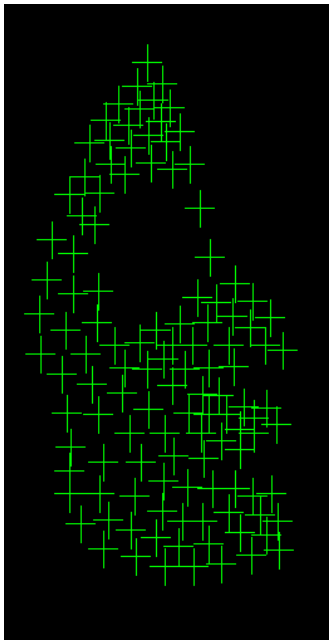


Figure 4.18 Reconstruction made in Kubotek Keycreator-side view

As a summary, the 3D reconstruction procedure starts by correcting the distorted images and extracting the camera parameters by camera calibration. The setup is quite robust to external shocks and there is no need to calibrate the camera every time before the reconstruction code is run. After calibration, the scene that is to be reconstructed should be marked properly and the necessary filters for that scene should be devised. By using the centroid calculating algorithm, the marker centroids are determined and proper correspondence should be achieved. The only thing that remains to be done is to feed the coordinates of the labeled marker centroids to the reconstruction algorithm and calculate the 3D coordinates.

4.6 Mimic Tracking

To test the setup, a sample mimic; i.e. smiling, was exhibited. The mimic video was caught in 30 fps and the face was similarly marked, only with fewer markers. The aim is to determine the trajectory and the displacement of each marker during smiling. The aforementioned procedures were followed during centroid calculation and distortion correction. The markers are traced in each frame using an algorithm that finds the marker with the closest coordinates to the marker in the previous frame. In this fashion, the markers are traced from the neutral face to the apex of the smiling face in 52 frames. The marker labels can be seen in Figure 4.19 and the trajectories can be seen in Figure 4.20, Figure 4.21, and Figure 4.22. Figure 4.24. shows the 3D trajectories of the markers. The raw displacement results and the corrected results are given in Table 4.1.

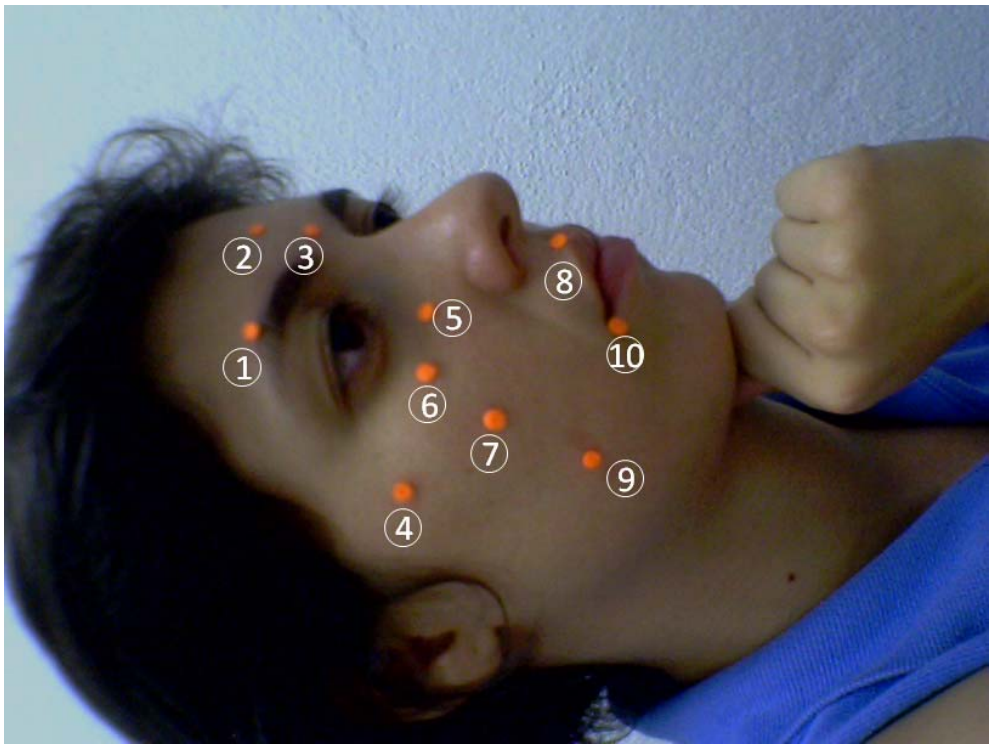


Figure 4.19 Marker labels


Figure 4.20 Trajectories viewed from the right camera before smiling

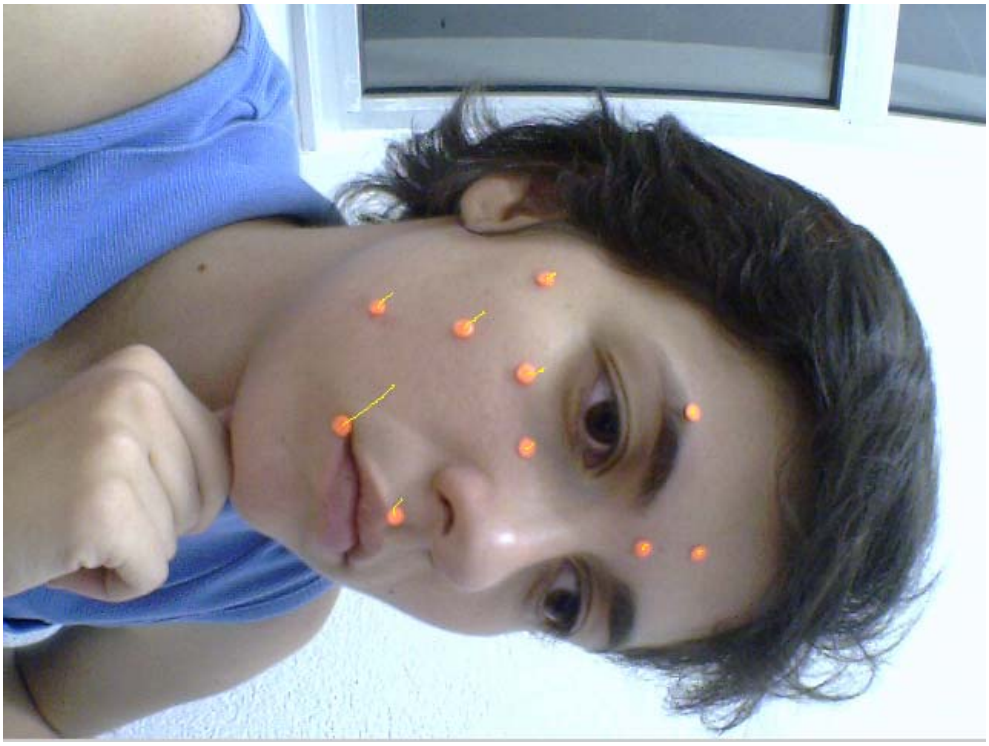


Figure 4.21 Trajectories viewed from the left camera before smiling

Figure 4.22 Trajectories viewed from the right camera after smiling

Figure 4.23 Trajectories viewed from the left camera after smiling

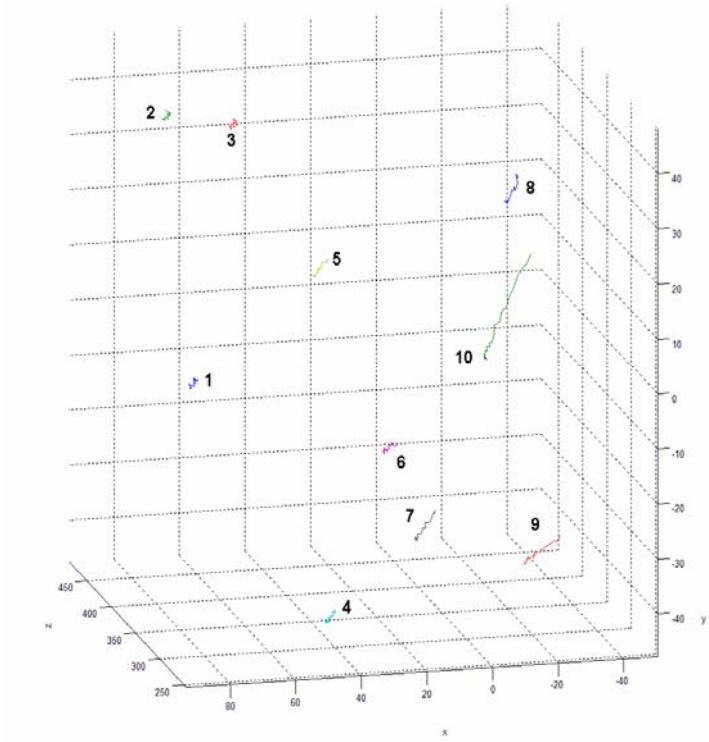


Figure 4.24 3D trajectories of the points

Table 4.1 Displacements of markers for smiling

Label Number	Displacement of Markers	Displacement with Scale Factor
1	1.5608 mm	1.4506 mm
2	1.7919 mm	1.6653 mm
3	2.2436 mm	2.0851 mm
4	3.4793 mm	3.2336 mm
5	3.8836 mm	3.6093 mm
6	6.3639 mm	5.9144 mm
7	8.6376 mm	8.0275 mm
8	9.5659 mm	8.8902 mm
9	22.2238 mm	20.6541 mm
10	33.9584 mm	31.5599 mm

Possible errors for this measurement are slight head movements during the shooting, errors introduced during calibration, distortion correction and filtering. Note that these values can lead the robot face designer for achieving realistic mimics on the silicone skin of a robot face. Similar tests can be carried out with this setup for different mimics.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

This study is the first attempt through the development of a robot face that can mimic human facial expressions. While trying to form a road map for developing such a face, the inevitable necessity of developing a setup for face mimic tracking revealed itself. A highly configurable setup for different face orientations is built and a MATLAB library for 3D reconstruction is written. Two different applications are presented in this thesis to demonstrate the setup and a mimic tracking example is provided. The following conclusions are derived as a result of this study.

1. A low-cost setup that can calculate the world coordinates with approximately 0.3 mm error is developed.
2. Other than face mimic tracking, this setup can be used for other volumetric measurement and point tracking applications.
3. The lens imperfections vary for each camera; therefore different distortion correction procedures should be run.
4. Round-off errors are presented to the system during distortion correction.

5. As the object to be viewed gets away, the quantization error increases. Larger camera resolutions will give more accurate results. However, higher resolution needs fast computation and more time.
6. Smarter filters should be used for mimic tracking. Those filters that use conventional image processing techniques are subject to noise. A trade off must be found. Pattern recognition may be a solution.
7. Camera positions are important. As the optical axes of the cameras get parallel with respect to each other, the error increases and the accuracy is lowered. However, as the angle between the optical axes increases, it becomes harder to shoot all the points on the face in both images.
8. Synchronous frame grabbing is very hard for webcams. This problem was solved by shooting them manually.
9. Since calibration is run manually; errors of around 2 pixels are presented to camera parameters.
10. As the object to be found gets near to the edges of the image, the error increases. Although distortion correction is done, still some errors persist.

5.1 Future Work

In this study the very first step towards designing a robot face is taken, i.e. a setup for face tracking is developed. The necessary algorithms for computing the 3D coordinate of any object moved in the workspace of the setup frame are written and a MATLAB library is formed The most important consideration should be solving the correspondence problem easily. To solve this problem, gray level matching can be used along with

epipolar line theory. Once this problem is solved, each marker point should be tracked in consecutive frames. By exhibiting the mimics slowly and shooting the videos at a high frame rate, the tracking can be done by calculating the nearest marker position in the next frame. Since human skin is a highly deformable material, on some mimics, especially while smiling, some markers overlap. This problem can be solved similarly to the method explained in [30].

Following the extraction of the trajectories of the markers on human skin, these trajectories should be mimicked on artificial silicone. Room Temperature Vulcanized (RTV) silicone has special types produced specifically for animatronic face production. At the result of the mimic tracking, some regions may be found to move together for certain mimics or some regions may be found not to move at all. To imitate these regions various mechanisms should be designed and the type of actuators should be decided. The actuators include not only the motors or the pistons that will generate the motion but also the means of pulling the silicone like cords, strings, etc. The contact points of the actuators to the silicone should be designed. This step is highly experimental, based mainly on trial and error procedures with silicone. Also, the skull of the robot face must be designed very carefully. The skull plays a vital role in robot face design such that the silicone skin should be attached to the skull at regions where it does not move for any of the mimics. Moreover, the actuators are going to be fixed inside the skull most probably. Following the skull design, the face should be molded and the silicone face should be generated. After the connection of all the actuators, the necessary

electrical circuitry should be devised and the relevant motion schemes should be exhibited on the face. This roadmap, which was the included in the scope of this thesis, should help the future researchers extensively.

To sum up, for 3D reconstruction, the inverse perspective mapping should be utilized. When a 3D point coordinate is mapped onto a 2D image pixel coordinate, one dimension is lost. To recover this missing dimension, the method of binocular stereo is implemented. In this method 2 cameras view the scene and the scenery point is mapped onto two different image planes, from which the lost dimension is extracted. The correspondence problem which is the identification of one point in both images was solved by different algorithms. The solution of correspondence problems give the image plane coordinates of the same point on both the image planes.

By using the camera parameters that are extracted by camera calibration, the exact coordinates of the point can be computed. The camera calibration is done using a toolbox in MATLAB. A rectangular grid is shown to both of the cameras and the corners of the grid pattern are used as the calibration points. Various snapshots of the grid at different positions and orientations are taken to solve a homography leading to the intrinsic and extrinsic parameters of the camera.

The distortion correction is done independently from the camera calibration. If one assumes that the distortion is a deviation of each pixel from its ideal, undistorted case, our method solves this deviation for each

pixel. In order to solve the deviations, first a circular grid is shown to the cameras. Then the viewed, distorted grid and the original undistorted grid are aligned on each other and the original grid is scaled with respect to the distorted grid. After this scaling, the differences of the original grid circle coordinates and the distorted grid circle coordinates are calculated. Then these differences are fit on to the whole image plane to find each pixel's deviation.

After distortion correction, each image is filtered to extract the feature of the object or the point. This filtering is mainly done by color thresholding and edge detection. For edge detection, Canny edge filter is used. Thresholding uses the RGB and HSV color spaces to extract the points or the objects. The filters from HSV and RGB thresholding combined with the Canny edge detection filter combined with various opening and closing algorithms yields the final filtered image.

The 3D reconstruction function takes in the pixel coordinates of each extracted point. The transformation matrix is formed by multiplying the intrinsic camera parameters matrix by extrinsic camera parameters matrix. In order to solve the inverse perspective transformation, the overdetermined transformation matrix should be solved by using the pseudo-inverse technique.

REFERENCES

[1] K. Dautenhahn, A. Billard, Bringing Up Robots or-The Psychology of Socially Intelligent Robots: From Theory to Implementation, Proceedings of the Autonomous Agents, 1999.

[2] T. Fong, I. Nourbakhsh, K. Dautenhahn, A Survey of Socially Interactive Robots, Robotics and Autonomous Systems, 42, 2003, 143-166.

[3] B. Duffy, Anthropomorphism and the Social Robot, Robotics and Autonomous Systems, 42, 2003, 177–190.

[4] C. DiSalvo, F. Gemperle, J. Forlizzi, and S. Kiesler, All Robots Are Not Created Equal: The Design and Perception of Humanoid Robot Heads, Proceedings of the Conference on Designing Interactive Systems: processes, practices, methods, and techniques, 2002, 321-326.

[5] United Nations Economic Commission for Europe and The International Federation of Robotics: World Robotics 2005, United Nations, Geneva, 2005.

[6] M. Montemerlo, J. Pineau, N. Roy, S. Thrun, and V. Verma, Experiences with a Mobile Robotic Guide for the Elderly, Proceedings of the AAAI National Conference on Artificial Intelligence, Edmonton, Canada, 2002.

[7] R. D. Schraft, C. Schaeffer and, T. May, Care-o-bot: The Concept of a System for Assisting Elderly or Disabled Persons in Home Environments. IECON: Proceedings of the IEEE 24th Annual Conference, 4, 1998, 2476–2481.

[8] H. Kitano, editor, Proceedings of RoboCup-97: The First Robot World Cup Soccer Games and Conferences, Springer-Verlag, Berlin, 1998.

[9] S. Thrun, M. Bennewitz, W. Burgard, A.B. Cremers, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, MINERVA: A Second-Generation Museum Tour-Guide Robot, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'99), 1999.

[10] F. Michaud, A. Clavet, G. Lachiver, and M. Lucas, Designing Toy Robots to Help Autistic Children - An Open Design Project for Electrical and Computer Engineering Education, Proceedings of American Society for Engineering Education, 2000.

[11] C. Breazeal, Towards Sociable Robots, Robotics and Autonomous Systems, 42, 2003, 167–175.

[12] R. A. Brooks, C. Breazeal, M. Marjanovic, B. Scassellati, M. M. Williamson, The Cog Project: Building a Humanoid Robot, C. Nehaniv (Ed.): Computation for Metaphors, Analogy, and Agents, LNCS 1562, Springer-Verlag, Heidelberg Berlin, 1999, 52-87.

[13] M. Scheef, J. Pinto, K. Rahardja, S. Snibbe, R. Tow, Experiences with Sparky: A Social Robot, Proceedings of the Workshop on Interactive Robot Entertainment, 2000.

[14] L. Cañamero, J. Fredslund, I show you how I like you—can you read it in my face? IEEE Transactions on Systems, Man and Cybernetics 31, 5, 2001, 454-459.

[15] C. Breazeal, P. Fitzpatrick, That Certain Look: Social Amplification of Animate Vision, Proceedings of the AAAI Fall Symposium on Society of Intelligence Agents—The Human in the Loop, 2000.

[16] C. Breazeal, Designing Sociable Robots, MIT Press, Cambridge, MA, 2002.

[17] A. Takanishi, H. Takanobu, I. Kato, T. Umetsu, Development of the Anthropomorphic Head-Eye Robot WE-3RII with an Autonomous Facial Expression Mechanism, Proceedings of the 1999 IEEE International Conference on Robotics & Automation, 1999, 3255-3260.

[18] A. Takanishi, K. Sato, K. Segawa, H. Takanobu, H. Miwa, An Anthropomorphic Head-Eye Robot expressing Emotions based on Equations of Emotion, The Proceedings of the 2000 IEEE International Conference on Robotics & Automation, 2000, 2243-2249.

[19] T. Hashimoto, S. Hitramatsu, T. Tsuji, H. Kobayashi, Development of the Face Robot SAYA for Rich Facial Expressions, Proceedings of SICE-ICASE International Joint Conference 2006, 2006, 5423-5428.

[20] D. Hanson et al., Upending the Uncanny Valley, Proceedings of the 20th National Conference on Artificial Intelligence, 2005.

[21] P. Ekman and W.V. Friesen, The Facial Action Coding System, Consulting Psychologists Press, 1978.

[22] Y. Tong, Automated Facial Action Units Recognition, http://www.ecse.rpi.edu/~cvrl/tongy/aurecognition.html, 2007

[23] B. Klaus, P. Horn, Robot Vision, The MIT Press, McGraw-Hill, 1986

[24] E. Trucco, A. Verri, Introductory Techniques for 3-D Computer Vision, Prentice Hall, 1998

[25] Z. Zhang, Flexible Camera Calibration By Viewing a Plane From Unknown Orientations, The Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999, 666-673.

[26] J. Heikkila, O. Silvén, A Four-Step Camera Calibration Procedure with Implicit Image Correction, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97), 1997, 1106.

[27] D. Vernon, Machine Vision- Automated Visual Inspection and Robot Vision, Prentice Hall, 1991.

[28] R. Gonzalez, R. E. Woods, S. L. Eddins, Digital Image Processing Using MATLAB®, Prentice Hall, 2004.

[29] J. Y. Bouguet, Camera Calibration Toolbox for MATLAB, http://www.vision.caltech.edu/bouguetj/calib_doc/, April 2007.

[30] M. S. Shafiq, S. T. Tümer, H. C. Güler, Marker Detection and Trajectory Generation Algorithms for a Multicamera Based Gait Analysis System, Mechatronics, 11, 2001, 409-437

APPENDIX A

MATLAB LIBRARY USER MANUAL

1. Open the "Codes" folder.
2. If the cameras have already been calibrated, go to step 6.
3. Firstly the camera calibration should be executed to determine the intrinsic and extrinsic camera parameters. The grid pattern in **calib_grid.pdf** should be shown to both of the cameras at different positions and orientations. It is important to place the grid so that both cameras see the same region of the grid. Therefore, the snapshots should be taken at the same time from the cameras. 15-20 image snapshots are enough to get an accurate calibration result. All of the images should be of the size 640 x 480. Please note that all the images should be saved with the same prefix like, right1.jpg, right2.jpg,..,left1.jpg, left2.jpg…, etc.
4. After the snapshots of the grid pattern are taken, these snapshots should be corrected for distortion before the calibration procedure. The m-file **correct_calibration_right.m** corrects the images taken by the right camera. The first line of the code should be changed according to the prefix and the format of the images. The code

reads the distortion error values from an Excel file given in the same folder and outputs the corrected 800 x 800 pixel images. Please repeat the same procedure for the m-file **correct_calibration_left.m.**

5. After correcting the distortions for the grid pattern images, apply the calibration procedure given in http://www.vision.caltech.edu/bouguetj/calib_doc/. Do not forget to execute the stereo calibration at the end. Also, note that while correcting for distortions during calibration, the kc value should be entered as zero since the images are already corrected.
6. Take the snapshots of the object/point(s) whose 3D coordinates you would like to calculate. Color green is more easily found on the face and the library has already a filtering code for green, **green_filter.m**. This code works on global thresholding on hue values and edge filtering on green level of RGB space and value level of HSV space. Due to lighting conditions the threshold values may be changed and those points that could not be found by the filter may be added by hand. See the comments in the m-file for more detail.
7. If you seek to determine the coordinates of multiple points you should use the m-file **multi_center_find.m** to determine the centroids of these points and label them using connected component labeling. For determining the center of a single point/object, the code **center.m** can be used. Note that these two functions take the binary filtered images as the input and output

the centroids, number of total objects in the images and the labeled image.

8. Note that all the filtering was done on the distorted image. Finally after filtering and centroid finding run the code **centroid_correct_right.m** and **centroid_correct_left.m** to correct the centroids calculated on the distorted image. Also, run **correct_right.m** and **correct_left.m** for correcting the distorted image. The former couple of functions takes the distorted centers as the input and output the corrected centers. The latter functions take the distorted image and output the corrected image. Do not forget to take into account the necessary offsets if you have cropped the image previously.
9. Since the matching or correspondence problem is not pressed in this thesis; the corresponding points on the face were numbered manually. If a circular grid, like the one that is given at the end of this appendix, is shown to the cameras, the codes **ordered_right.m** and **ordered_left.m** can be used. The details of these codes can be read in the comments of the code. However, these codes only address to very specific cases, therefore it is advised to develop a new smarter algorithm which uses gray-level matching or feature-matching depending on the case.
10. Once the matching is done and the conjugate pairs are determined the x and y pixel coordinates should be fed into the code **ultimate.m** to calculate the 3D coordinates with respect to the left camera. Note that after camera calibration the intrinsic and extrinsic parameters should be written to the relevant constants in

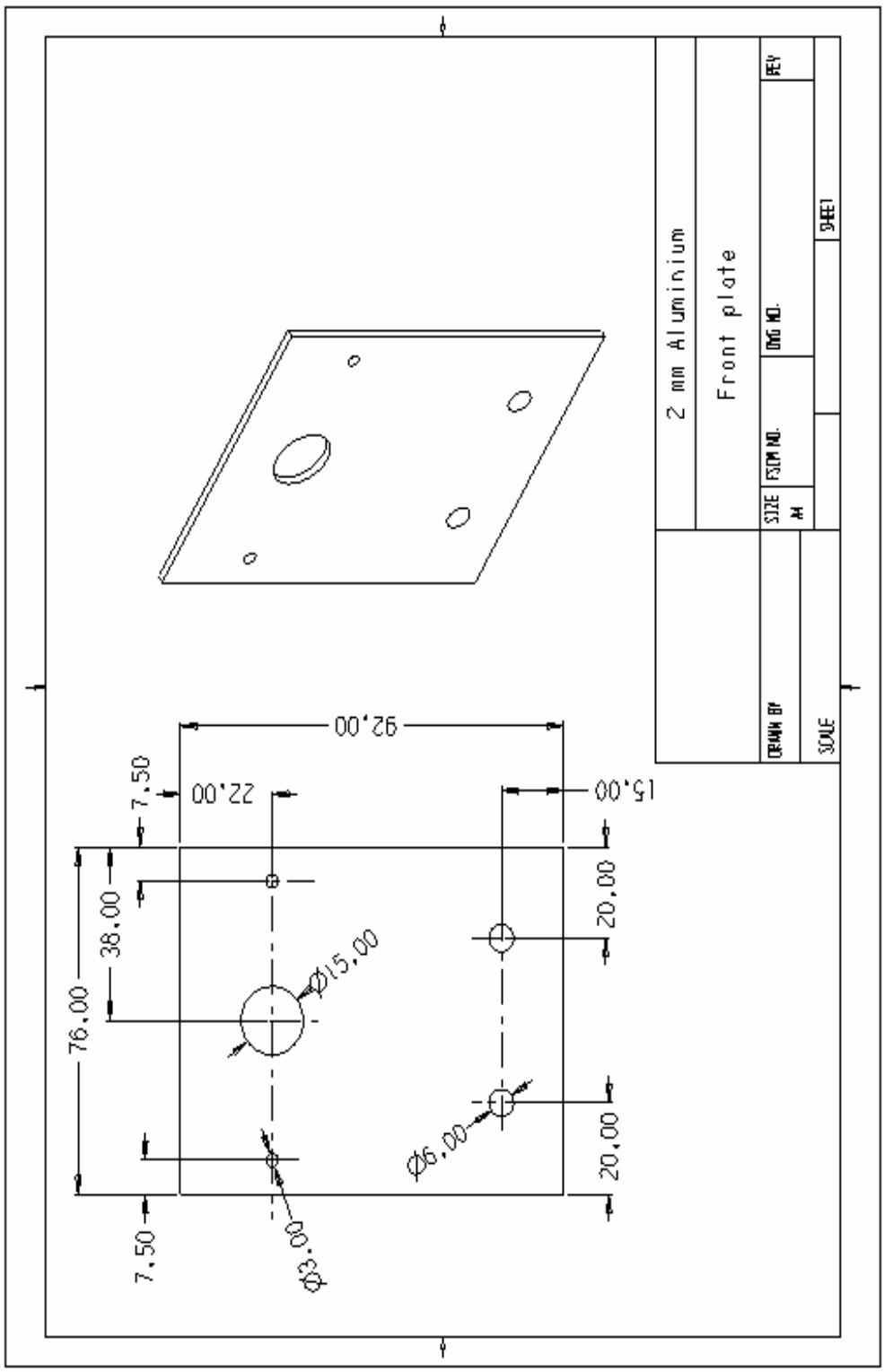
the code. The rotation and translation matrices are calculated with respect to the left camera; therefore the left camera is the origin of the setup. The orientations of the axes are drawn on the setup. The code calculates the 3D coordinates and plots them on a figure.

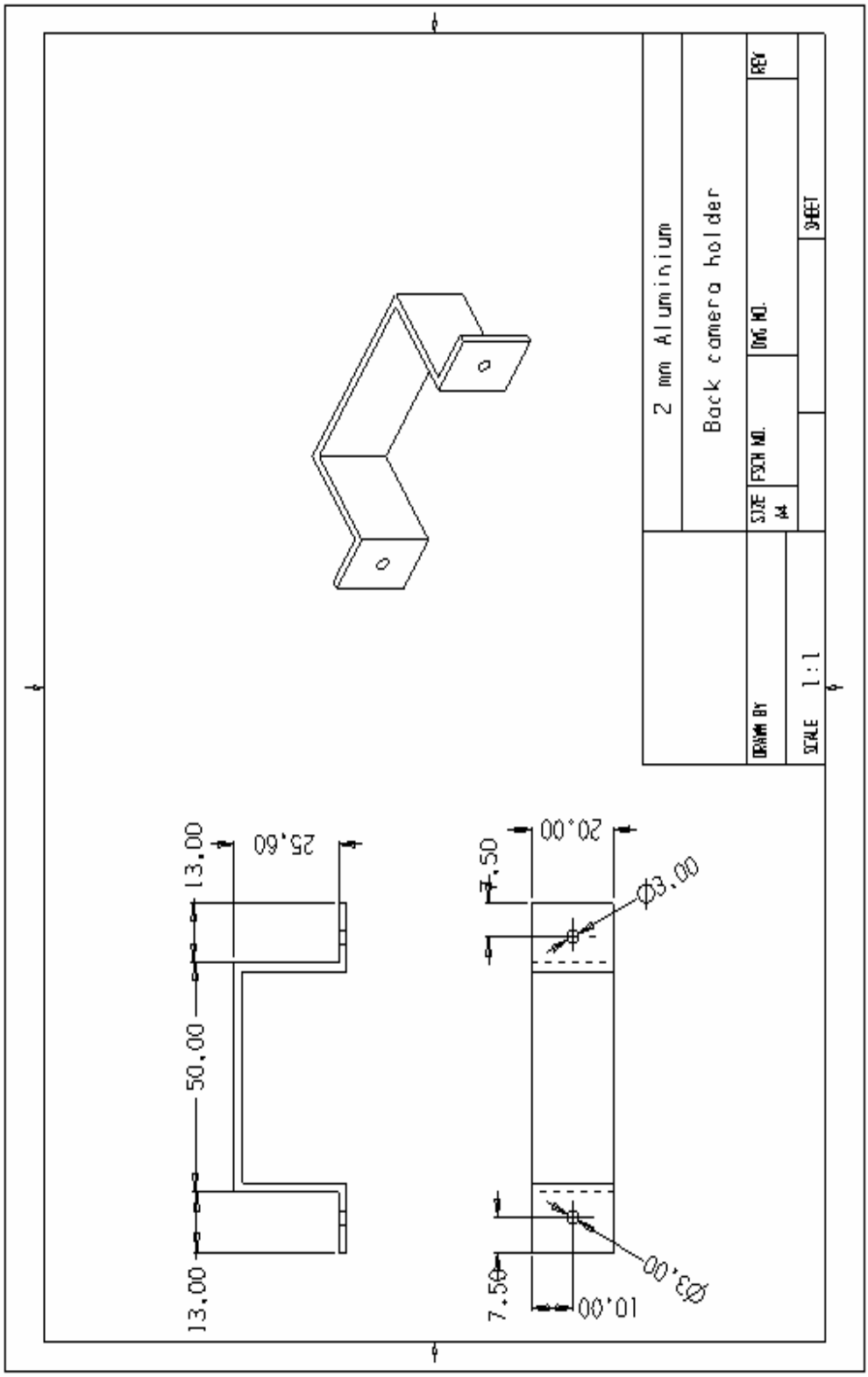
11. Apart from all these codes several codes that were devised in this thesis are given with this library to set an example and maybe to be further developed. Out of these the code **orange.m** finds the 3D coordinates of an orange ball while it is being moved in the setup workspace. The algorithm **grid.m** calculates the coordinates of multiple circular points on a planar grid. **face.m** contains the algorithm written for the face reconstruction explained in this thesis. The relevant images for these codes are also included in the library.
12. If one would like to calculate the distortion error for the cameras once again the m-files **calc_dist_right.m** and **calc_dist_left.m** are also included in the library. These codes plot the distortion error surfaces and write the pixel distortions to an Excel file. The grid for distortion correction is also included under the name **grid.ppt**. One should show this open this PowerPoint file on the monitor and show it to the camera and take the snapshot of the monitor.
13. The special frame extracting code which extracts the frames from two .avi formatted videos is also included in the library under the name **extract_frames.m**. One can include this code at the beginning of the mimic tracking algorithm.
14. All these aforementioned codes contain the necessary commenting for future help.

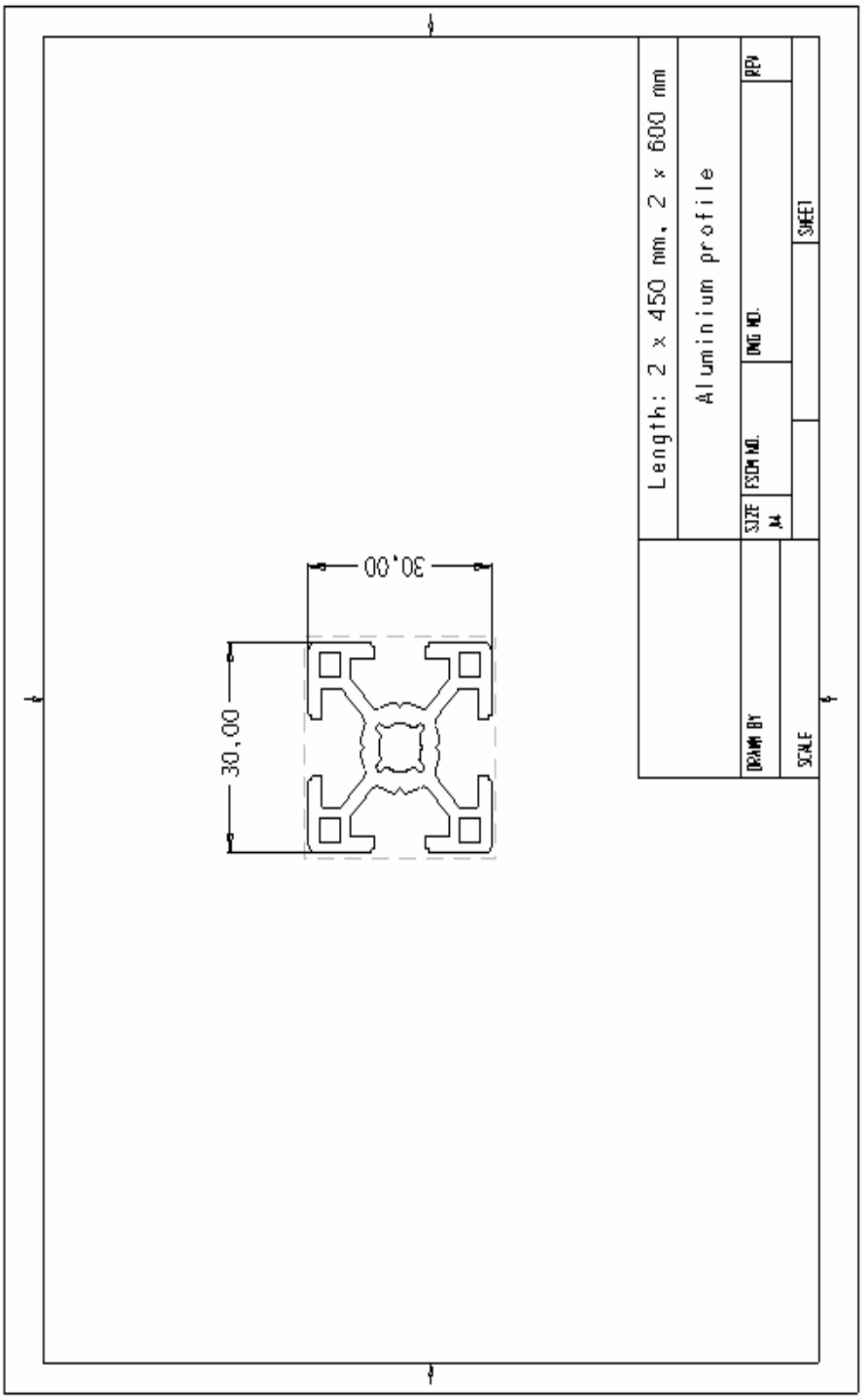
APPENDIX B

TECHNICAL DRAWINGS

In this appendix, the technical drawings of the parts used in the setup are given. All the materials are aluminium and all the dimensions are in mm.


2 mm Aluminium
Front plate
SIZE A4
FSTCM NO.
DWG NO.
REV
SHEET
DRAWN BY
SCALE
92.00
7.50
22.00
15.00
76.00
38.00
20.00
Ø15.00
Ø6.00
20.00
7.50
Ø3.00


2 mm Aluminium
Back camera holder
SIZE
A4
FSCM NO.
DWG NO.
REV
SHEET
DRAWN BY
SCALE 1:1
25,60
13,00
50,00
13,00
20,00
7,50
Ø3,00
Ø3,00
10,00
7,50


Length: 2 x 450 mm, 2 x 600 mm
Aluminium profile
SIZE
A4
FSDM NO.
DWG NO.
REV
SHEET
DRAWN BY
SCALE
30,00
30,00