3D HAND TRACKING IN VIDEO SEQUENCES

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

BY

AYKUT TOKATLI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2005

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Canan Özgen

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. İsmet Erkmen

Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Prof. Dr. Uğur Halıcı

Supervisor

Examining Committee Members

Prof. Dr. Kemal LEBLEBİCİOĞLU	(METU,EE)	_____
Prof. Dr. Uğur HALICI	(METU,EE)	_____
Assist. Prof. Dr. İlkay ULUSOY	(METU,EE)	_____
Assist. Prof. Dr. Kürşat ÇAĞILTAY	(METU,CEIT)	_____
Assist. Prof. Dr. İlhan KONUKSEVEN	(METU,ME)	_____

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 : Aykut Tokatlı

Signature :

ABSTRACT

3D HAND TRACKING IN VIDEO SEQUENCES

Tokatlı, Aykut

MS, Department of electrical and Electronics Engineering

Supervisor: Prof. Dr. Uğur Halıcı

September 2005, 112 pages

The use of hand gestures provides an attractive alternative to cumbersome interface devices such as keyboard, mouse, joystick, etc. Hand tracking has a great potential as a tool for better human-computer interaction by means of communication in a more natural and articulate way. This has motivated a very active research area concerned with computer vision-based analysis and interpretation of hand gestures and hand tracking.

In this study, a real-time hand tracking system is developed. Mainly, it is image-based hand tracking and based on 2D image information. For separation and identification of finger parts, coloured markers are used. In order to obtain 3D tracking, a stereo vision approach is used where third dimension is obtained by depth information. In order to see results in 3D, a 3D hand model is developed and Java 3D is used as the 3D environment.

Tracking is tested on two different types of camera: a cheap USB web camera and Sony FCB-IX47AP camera, connected to the Matrox Meteor frame grabber with a standard Intel Pentium based personal computer. Coding is done by Borland C++ Builder 6.0 and Intel Image Processing and Open Source Computer Vision (OpenCV) library are used as well. For both camera types, tracking is found to be robust and efficient where hand tracking at ~8 fps could be achieved.

Although the current progress is encouraging, further theoretical as well as computational advances are needed for this highly complex task of hand tracking.

Keywords: Hand tracking, hand gesture recognition, hand modeling, 2D-3D hand model, human-computer interaction (HCI), Camshift algorithm, HSV colour.

ÖZ

VİDEO GÖRÜNTÜLERİNDE 3 BOYUTLU EL İZLEME

Tokatlı, Aykut

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü

Tez Danışmanı: Prof. Dr. Uğur Halıcı

Eylül 2005, 112 sayfa

Klavye, fare, kontrol çubuğu gibi kullanılması hayli zor olan donanımsal aletlere alternatif olarak el işaretlerinin kullanılması, her geçen gün daha cazip hale gelmektedir.Buna bağlı olarak, elin hareketlerinin izlenmesi, bilgisayar kullanımında, daha doğal ve daha anlaşılır olması bakımından büyük bir potansiyel taşımaktadır. Bilgisayar destekli görüntü analizi ve el hareketlerinin bilgisayar sistemlerinde izlenerek yorumlanması çok aktif bir araştırma sahasının ortaya çıkmasına neden olmuştur.

Bu çalışmada, gerçek zamanlı çalışan bir el izleme sistemi gerçekleştirilmiştir. El hareketlerini izleyen bu sistem, görüntü tabanlı yapıda olup, iki boyutlu görüntü bilgisini kullanmaktadır. İzlenilen bir elde, parmakların ve parmaklardaki eklemler arası kısımların hangisi olduğunun bilgisini bulabilmek için birbirinden farklı renkli işaretleyiciler kullanılmıştır. Takibin sonuçlarını, üç boyutlu olarak görebilmek için stereo görüntüleme yaklaşımı kullanılmıştır. Bu yaklaşımda, üçüncü boyut bilgisi derinlik bilgisinden elde edilmiştir. Elin takibinin sonuçlarını 3 boyutta görmek için 3 boyutlu bir el modeli geliştirilmiş ve 3 boyutlu ortam olarak Java 3D aracı kullanılmıştır.

El hareketlerinin izlenmesi, Intel Pentium tabanlı bir bilgisayarda hem usb' den çalışan ucuz bir kamera hem de Matrox Meteor kartına bağlı Sony FCB-IX47AP markalı kamera ile gerçekleştirilmiştir.

Kodlama, açık kaynak kodlu Intel Görüntü İşleme ve bilgisayarlı görüntüleme (OpenCV) kütüphanesi ve Borland C++ Builder 6.0 geliştirme ortamı kullanılarak yapılmıştır. El izleme, saniyede ortalama 8 görüntü hızıyla, kararlı ve etkili bir şekilde yapılmıştır.

Gerçek veriler kullanılarak yapılan denemelerde, el hareketlerinin büyük ölçüde takip edilebildiği görülmüştür. Bu çalışma her ne kadar cesaret verici olsa da, ileriye dönük teorik ve sayısal anlamda ilerlemelere ihtiyaç vardır.

Anahtar Kelimeler: El izleme, el hareketlerinin yorumlanması, elin modellenmesi, iki boyutlu ve üç boyutlu el modeli, insan-makine etkileşimi, Camshift algoritması, HSV renk formatı.

To My Family

ACKNOWLEDGEMENTS

I express my sincere appreciation to my thesis supervisor Prof. Dr. Uğur Halıcı and co-supervisor Ass. Prof. Dr. İlkay Ulusoy for their guidance and insight throughout the research.

I wish to thank to my parents Gülseren and Ahmet Solmaz Tokatlı, and to my brother Aytaç Tokatlı for their support and encouragement throughout the years of my education.

I would like to thank to all of the Computer Vision and Artificial Intelligence Research Group members. Particularly, Erdem Akagündüz, Tolga İnan, Suphi Erden, Mustafa Özuysal, Zafer Arıcan and Mehmet Soner Güler for their invaluable feedback, support, help and motivations.

TABLE OF CONTENTS

LIST OF TABLES

TABLE

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

Since the introduction of the computers, they are so tightly integrated with our daily lives. Everyday, new applications and hardware are constantly introduced. The integration grows up more and more. With the rapid increase of computer usage in this manner, the role of human-computer interaction (HCI) is becoming more important. As for vision-based applications, they provide more natural and intuitive means for interacting with computers. Hence users are allowed for richer and user-friendlier man-machine interaction. This can lead to new interfaces that will allow the deployment of new commands that are not possible with the current input devices. So, the role of these applications is very important. In fact, vision-based applications are very huge and popular research areas and there are lots of things to do. The tools and the data are available but their usability, especially interpretation, must be developed much more.

In our daily lives, we use hundreds of expressive movements as frequently as speaking. Sometimes, while interacting with the physical world, it is better or easier to use those expressive movements instead of speaking. These types of movements are called gestures. The definition of gesture can be extended to eye movements, head movements, body movements and etc but they are out of the scope of this thesis. Human hand gestures are a means of non-verbal interactions among people and range from simple actions such as using our hand to point at or move objects around, to the more complex ones that express our feelings and allow us to communicate with others. The means of communicating with computers at the moment are limited to keyboards, mice, light pens, trackballs, keypads, web cams etc. From those, the most popular ones are keyboards and mice. These devices have grown to be familiar but inherently limit the speed and naturalness with which we interact with the computer.

In this study, we wish to capture hand gestures to form a touch-free computer interface using techniques from computer vision. The main issue of hand gesture capturing is hand tracking which has received significant attention in the last few years, because of its crucial role and help to get a more natural and articulate way for many human computer interaction applications. So, it seems that hand tracking has great potential as a tool for better human-computer interaction.

1.2 The Scope of The Thesis

The aim of the thesis is to develop a system for real time 3D hand tracking based on stereo vision so that a 3D hand model could mimic the movements of the hand in the scene.

For this purpose, first of all, whole hand is segmented from background in input images. For the fingers and the rest of the hand, flesh colour information is used so that the hand regions can be extracted from the input images. Secondly, finger parts are segmented and ordered individually by using coloured markers. Two types of markers are used in our system. One type consists of green markers and the other type includes blue markers. Green markers point the fingers and they are used to index the fingers from thumb to little finger. Blue markers are used to point the joints and to separate individual segments of the fingers. Thirdly, after finding all segments of the fingers, they are ordered (indexed) from palm to the outer finger segments. And finally, these markers are used to calculate all relevant angle information which describes the posture of the hand.

The depth information is extracted using a stereo vision approach. The system uses two cameras where approximate 3D information of the hand can be extracted. At the end of segmentation and indexing processes, all related hand points and also the markers are available in both images of the stereo pair in 2D. Markers and regions obtained from each stereo image are matched between stereo pairs and corresponding pairs are found. By using camera calibration matrices (previously found from the manual stereo calibration of the cameras) and those matched pairs, the 3D information of the markers and finger regions is recovered.

In order to describe the posture of the hand, this 3D information is used. This is done by finding angles between 3D marker and finger region locations. Bu using the 3D information and the relationships between the finger segments with the markers, a 3D hand model's pose is updated for visualization of the hand tracking results. All relationships are in terms of the angles and these angle values are found from calculations in 3D.

Since the problem of 3D hand tracking is very complex, we restricted the movements of the hand and the fingers. Also, we restricted the environmental and viewing conditions such that there is not any occlusion, the lightning is stable and all hand regions are viewed from both of the cameras. Also, some assumptions are done to escape that complexity. Furthermore, a simple 3D hand model in JAVA is used to eliminate the necessity of a detailed human hand model.

1.3 Organization of The Thesis

The organization of the thesis is as follows;

- Chapter 2 gives the necessary information related with 3D hand model. Also the anatomy of a generic hand is explained.
- Chapter 3 presents a literature review on the hand tracking.
- In Chapter 4, colour spaces and their transformations are presented.
- Chapter 5 gives the necessary information related with the 'Camshift' tracking algorithm used in this thesis.
- In Chapter 6, cameras, coordinate systems and their transformations and finally stereo vision are explained.
- Chapter 7 gives the details of implementation with all explanations and pictures.
- In Chapter 8, summary and conclusion and also possible future enhancements are presented.

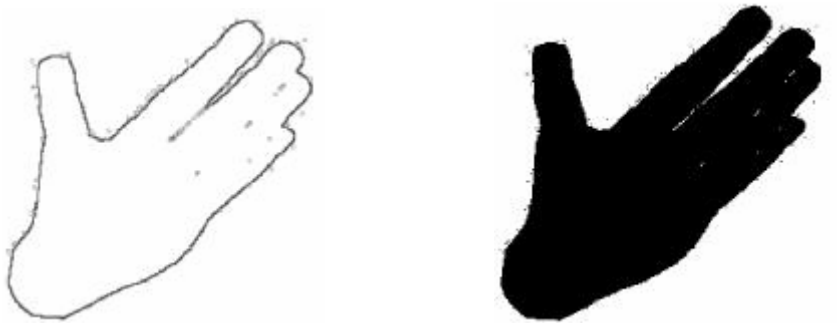
CHAPTER 2

3D HAND MODEL

2.1 Introduction

Automatic input and analysis of hand motion have attracted attention from computer animation and virtual reality researchers. Computer animation applications directed at the human hand must handle about 30 interacting parameters to specify a particular motion. Consequently, researchers have investigated several methods to derive these parameters from real hand motions. Virtual reality research has addressed the removal of the distinctions between the computer system and user environment; that is, the computer system presents a virtual space to the user's sense organs such as eyes, ears, and skin, and the user reacts using gestures and speech. For gesture analysis, the hands obviously play a major role.

Although mechanical gloves can capture hand motion in real time, they are expensive, and the associated measuring equipment inhibits free movement. More advantageous, a camera-based approach requires no mechanical gloves or motion-constraining monitoring equipment. The virtual reality and computer vision communities have targeted capturing hand motion via cameras. Researchers have successfully recognized specific fingers of the hand by silhouette images and distinguished a small set of hand signs by contour features of images. However, the silhouette or contour features recovered from the images do not provide sufficient information to generate 3D hand posture with the fingers positioned properly (Figure 2.1).



(a) (b)

Figure 2.1: (a) Contour (b) Binary silhouette

Also, analysis of hand posture from actual images involves complex problems even when employing conventional, simple models. In addition to the three common difficulties arising in practical computer vision – occlusion, noise, spurious data – the analysis process for the human hand is further complicated by the fact that;

- The human hand is an articulated structure with about 30 degrees of freedom and changes shape in various ways by its joint movements. Contributing to this, hand images change by both finger movements and hand movement as a whole.
- Because four fingers are adjacent to each other, occlusion of the fingers frequently occurs. Many tiny wrinkles existing on the skin also make it difficult to detect meaningful edges of the image. Thus, the conventional strategy of segmenting an image into parts by edge extraction and sequentially tracing the parts along the central axes does not apply.
- The fingers are comparatively short, and the surfaces around the joints are significantly deformed by joint movements, making it very difficult to accurately estimate joint positions in images.

2.2 Hand Anatomy and Description

With the previous studies of the biomechanics of the hand, a 3D skeletal hand model with 27 degree of freedoms (DOF's) is made (Rehg, 1995) (see Figure 2.2 and Figure 2.3) . This model presents a reasonable approximation to the real hand for the purpose of nearly all vision applications. A human hand consists of 14 + 1 joints (including the wrist) and 27 bone segments in between palm and fingers. These bones can be divided into 3 groups:

- carpals (wrist bones = 8),
- metacarpals (palm bones = 5)
- phalanges (finger bones = 14).

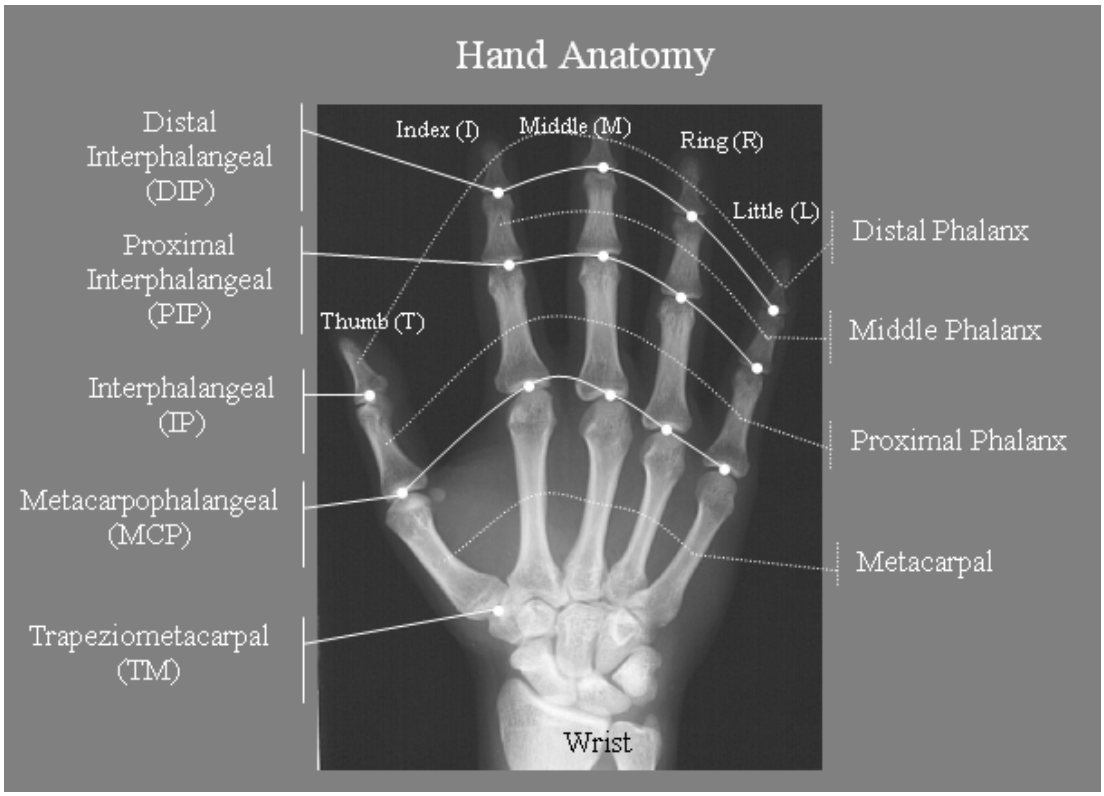


Figure 2.2: The anatomical structure of hand

Joints are named according to their location on the hand as

- metacarpophalangeal (MCP) : (joining fingers to the palm)
- interphalangeal (IP) : (joining finger segments)

In the literature, the 9 IP joints are described as having only 1 DOF, flexion-extension. All 5 MCP points are described as saddle joints with 2 DOF's: abduction-adduction (e.g., spreading fingers apart) in the plane defined by the palm, and flexion-extension. The thumb is more difficult to model. Most of its flexibility lies in the trapeziometacarpal (TM). This is another saddle point with 2 DOF's (same as above). As the last, the wrist's twist movement can be modeled by six DOF's (i.e., 3 DOF's for wrist translation and 3 DOF's for wrist rotation). (see Figure 2.3)

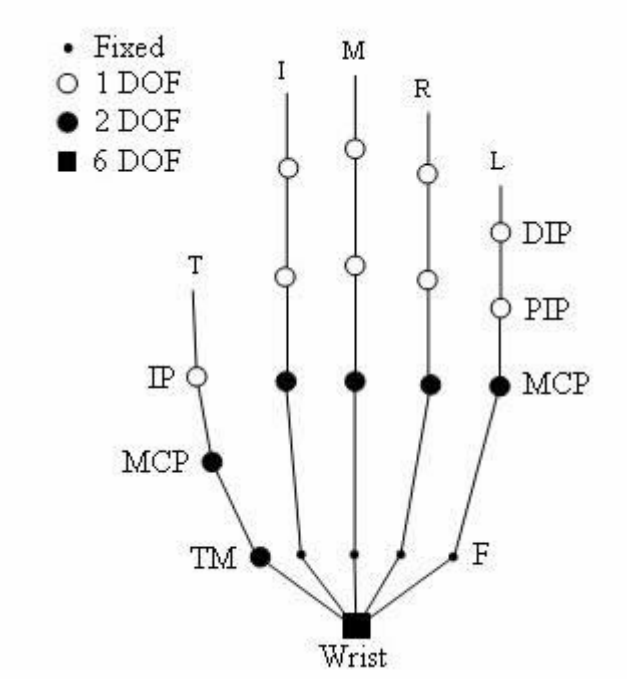


Figure 2.3: 3D Skeletal Model with DOF's

2.3 3D Hand Model

The 3D hand models have often been a choice for hand gesture modeling. They can be classified in two large groups:

- volumetric models
- skeletal models.

2.3.1 Skeletal Hand Model

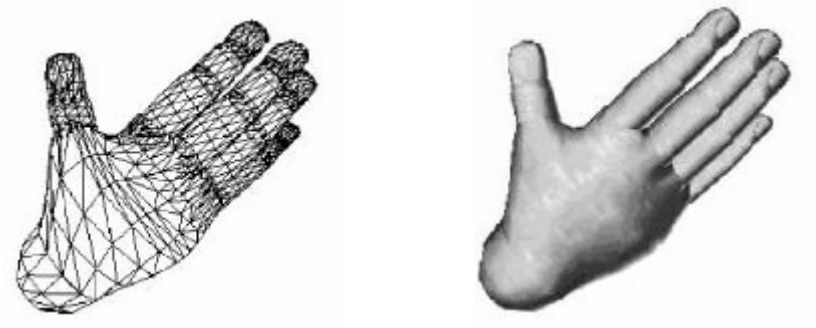
Instead of dealing with all the parameters of a volumetric hand and arm model, models with a reduced set of equivalent joint angle parameters together with segment lengths are often used. Such models are known as skeletal models and they are extensively studied in the human hand morphology and biomechanics (Thompson, 1981), (Tubiana, 1981). (see Figure 2.3)

2.3.2 Volumetric Hand Model

Volumetric models are meant to describe the 3D visual appearance of the human hand and arms. They are commonly found in the field of computer animation (Magnenat-Thalman, 1990), but have recently also been used in computer vision applications. In the field of computer vision, volumetric models of the human body are used for analysis-by-synthesis tracking and recognition of the body's posture (Koch, 1993), (Wren et al., 1996).

Briefly, the idea behind the analysis-by-synthesis approach is to analyze the body's posture by synthesizing the 3D model of the human body in question and then varying its parameters until the model and the real human body appear as the same visual images (Figure 2.4).

Most of the volumetric models used in computer animation are complex 3D surfaces (NURBS or non-uniform rational B-splines) which enclose the parts of the human body they model (Magnenat and Thalman, 1990). Even though such models have become quite realistic, they are too complex to be rendered in real-time.



(a) (b)

Figure 2.4: (a) 3D wireframe volumetric model (b) 3D Textured volumetric model

A more appealing approach, suitable to real-time computer vision, lies in the use of simple 3D geometric structures to model the human body (Rourke and Badler, 1980). Structures like generalized cylinders and super-quadrics which encompass cylinders, spheres, ellipsoids and hyper-rectangles are often used to approximate the shape of simple body parts, like finger links, forearm, or upper arm (Azarbayejani et al, 1996), (Clergue et al, 1995), (Downton and Drouet, 1991), (Etoh et al, 1991), (Gavrila and Davis, 1995). The parameters of such geometric structures are quite simple. For example, a cylindrical model is completely described with only three parameters: height, radius, and colour. The 3D models of more complex body parts, like hands, arms, or legs, are then obtained by connecting together the models of the simpler parts (Kakadiaris et al, 1994). In addition to the parameters of the simple models, these structures contain the information on connections between the basic parts. The information may also include constraints which describe the interaction between the basic parts in the structure.

CHAPTER 3

LITERATURE REVIEW

3.1 Introduction

In recent years, a growing number of computer vision researchers have been working on hand modeling, tracking, gesture recognition and its applications to user interfaces. Capturing hand articulation is a challenging problem, because hand motion exhibits many degrees of freedom. Representing the hand pose by the angles at each joint, the configuration space is approximately 27 dimensional. Additionally, self-occlusions of fingers introduce uncertainty for the occluded parts.

So, two main approaches for hand tracking can be identified:

- image-based (appearance-based) hand tracking
- model-based hand tracking

The first approach estimates hand states from images directly after learning the mapping from the image feature space to the hand configuration space (Feris et al., 2000), (Goncalves et al., 1995), (Faux and Pratt, 1979), (Efros and Leung, 1999). The mapping is highly nonlinear due to the variation of the hand appearances under different views. Other problems are the collection of large training data sets and the accuracy of the pose estimation. However, appearance based methods are usually fast, require a single camera only and have been successfully employed for hand tracking and gesture recognition tasks.

The other approach uses a 2D or 3D hand model (Duda and Hart, 1973). In case of a 3D model the hand pose is estimated by matching the projection of the model and the observed image features. The task is then formulated as a search problem in a high dimensional configuration space. Areas of research for this method include the efficient construction of realistic hand models, dimensionality reduction of the search space and the development of fast and reliable filtering algorithms to estimate the hand configuration.

These approaches are reviewed in Sections 3.2 and 3.3 respectively. In the following, body tracking, face tracking, eye tracking and etc. terms are used in explanations. In fact these concepts are out of this scope. But tracking is widely used in these areas as much as hand tracking. (or may be more than hand tracking). So with the 'tracking' term, the 'hand tracking' is meant. The others will be stated explicitly when used in text. Again, the term 'training' is used. This term is related with the neural networks. But it is also out of scope in this thesis.

3.2 Image-Based Hand Tracking

Image-based approaches are appropriate for applications that can be achieved without the need for a method that recovers high detailed quantitative description of the hand pose and movements relative to other objects in the 3D world. This is usually done by an image-based hand detector used in image based hand trackers. These methods are less suited for accurate estimation of the hand configuration, but are sufficient for classifying different hand postures. A set of training images is collected and features are extracted. All or some of the training images are labeled to be members of particular posture classes. Classifiers are then learned from the training set using a neural network, the Expectation-Maximization (EM) algorithm or other learning algorithms.

Model-free methods entail attaching meaning to clusters of low level image features. In (Quek, 1995), a small "vocabulary" in which there are 15 gestures is recognized by using information provided by clusters of edges in movement (image flow).

Pfinder (Wren et al., 1997) is a well known model-free tracking system which represents the human body as a collection of simple colour blobs. To generate dynamic predictions of the movement of the blobs, a Kalman filter (Bar-Shalom et al., 2001) is used.

Another example of model-free method is that of Berry et al. (Berry et al., 1998) which combines colour and motion cues. This multi-modal intelligent system also uses speech recognition for human-computer interface.

The simplicity of model-free methods is their main advantage, and this allows for low computational cost implementations. On the other hand, the main drawback of these methods is their lack of robustness. If low level local features are not associated with object level information, mismatches can occur if other objects having similar colour or corner features perform motions that are similar to those used to train the hand tracker.

Better segmentation results can be obtained by methods that combine colour and texture information. An example is the work of Flores et al. (Flores et al., 2000), which uses the Beucher-Meyer watershed paradigm with markers detected by a morphological operator based on computational learning. This method can segment and track hands even in the presence of skin coloured background and occlusions. However, its computational cost makes it possible for off-line processing only.

Moving to a slightly higher cognitive level, there are hand detectors/trackers based on shape descriptors (Costa and Cesar-Jr, 2001). Lyons (2002) designed a method for automatic sign language recognition based on search trees. In this method, there is a set of shape descriptors and motion information which is used in a higher level of the search trees. The feature vectors are used to combine the information taken with the set. This method is aided by the use of colour gloves and a uniform background. This leads to virtually error-free segmentation of the hands using the colour classifier and some image processing methods (smoothing and morphological operations). The quality of the segmentation results is a key factor to the success of this method.

The use of an efficient method for hand segmentation is also shown in the work of Von Hardenberg and Berard (2001). Segmentation is done through a method based on background subtraction, and then an algorithm based on fitting circles to the segmented image is used to find the finger tips. Once the fingertips are located, higher level algorithms can estimate the hand posture.

A common feature of methods based on shape descriptors is that their performance depends on the quality of the segmentation. This is a disadvantage of such techniques, as it is intended to avoid the use of intrusive features such as gloves and markers. Background subtraction is also not going to be considered, as there will be

multiple objects in the scene that can move. However, it is possible to segment skin with the only condition that the objects in the background are not skin coloured.

Active contour-based methods (Blake and Isard, 1998) as well as deformable templates (Baumberg and Hogg, 1995), (Hill et al., 1995), (Bowden and Sarhadi, 2002), (Bowden, 1999) have the advantage of providing more detailed information about the shape of the objects. Furthermore, they are robust to small variations on pose and on shape of the hand. These methods derive or learn a model from what is explicit in the image. Once deformed, contours or templates can be frozen into 2D shape models. This is a simple and successful method if the original viewpoint is maintained, or if characteristic views are available.

Athitsos and Sclaroff (2002) presented an appearance-based framework for 3D hand shape classification and simultaneous camera viewpoint estimation. Given an input image of segmented hand, the most similar matches from a large database of synthetic images are retrieved. A hierarchical retrieval algorithm is used. It combines computationally cheap similarity measures with more accurate and expensive measures. This method seems to be suitable for initialisation of a 3D hand tracker.

Multiple views of hands are used by Hamada et al. (Hamada et al., 2002) to recover the hand shape based on a feature vector extracted from the silhouette contour. This feature vector is projected into an eigenspace for classification. Multiple view data is also employed by Ju et al. (Ju et al., 1996). Three different views are used to design two-dimensional templates (cardboards) that represent projections of the object in each view. These templates are used to track human body while walking from frontal, oblique and side on. They are designed as projections of an articulated human body based on cylinders. However, it is assumed that the relative position and orientation of the person's body does not change in a single video sequence. A similar idea for parsing pictures of people is implemented in the work of Ronfard et al. (Ronfard et al., 2002).

Another image-based method to track an articulated model is the one proposed by MacCormick and Isard (2000). Four degrees of freedom of hands are tracked: planar translation, orientation of the thumb and of the index finger. A combination of an

articulated model with active contours to segment the index finger is employed. The tracking is done through a CONDENSATION (Stochastic Conditional Density Propagation) approach (Isard and Blake, 1998). The obtained results combine speed, robustness, accuracy and simple hardware requirements. Unlike model-free approaches, that system recovers continuous parameters rather than recognising gestures from a discrete “vocabulary".

In general, 2D systems have proved successful for many specific applications, and they can often recover the gross movements of a hand. However, such methods have difficulty on reconstructing more fine-grained gestures and on recovering quantitative description of hand pose. Another major difficulty of 2D tracking methods is that the robustness is affected by the changes in perspective view.

3.3 Model-Based Hand Tracking

For hand tracking, the 3D methods are those that try to recover geometrical parameters of the hand pose described as a 3D model. Such model is usually, but not always, designed using constraints afforded by bone linkage to reduce the time complexity of the system. Approaches for 3D hand tracking have varied in the following features:

- the number of views used:

  (monocular, stereo, trifocal and multiple)

- the type of model used:

  (deformable models, kinematic chain of cylinders, cones, or superquadrics)

- the minimisation and estimation method

The first work studied here is that of Ahmad (1995), which can be seen as a bridge between image based methods and 3D model based methods. It is a monocular vision system that can provide an approximation of 19 DOFs of a hand using neither a 3D representation of the hand nor a calibrated system. The method is based on segmentation of skin colour through a histogram-based classifier that uses the CIE chromatic colour space.

The processing is speeded up by using sub-sampling of the images (using patches rather than pixels). The size of the patches is defined according to the desired frame rate. The mass centre and principal axis of the segmented image are used to determine the position and orientation of the hand in the plane (3 DOFs). A simple idea to estimate the distance between the camera and the hand (Z) was implemented using the fact that the camera acquires top view images of the hand, and that the hand is parallel to the camera plane. Z (1 DOF) is estimated according to the proportion between the hand area (number of skin coloured patches in the image) in the current frame and in the first frame of the sequence. The search in the next frame is constrained by a rectangle which has position and area defined according to the skin colour blob in the current frame.

Fingers are defined as articulated objects with 3 planar joints and they are not allowed to be moved slightly from side to side, hence, the hand configuration has 15 DOFs. The assumption that the user is wearing a full sleeved shirt aids this method on recovering the hand shape. This is done by representing the palm as a circle and the fingers as lines emanating form the centre of the wrist. Fingers are allowed to rotate around the wrist centre for the initialisation of the model. In order to find the palm, the system computes the largest circle around the mass centre of the patches such that the area of all patches within the circle is close to the actual area of the circle.

Every patch that is outside the circle could be part of a finger. The fingers are determined by the 5 highest scores of the Hough transform (Gonzalez and Woods, 2000) from the centre of the wrist. This determines the actual angle of each finger and the fingertip locations are determined by the maximal patch in each direction.

The simplicity of this system made it possible to obtain tracking results at 10 frames per second. A number of applications can be designed using this approach, but this work has several drawbacks. The first is the fact that it does not track rotations in depth and does not handle self-occlusion problems. Another problem is that, since no calibration data is used, this system can not be applied for problems in which it is necessary to obtain accurate data about the 3D position of the hand.

Furthermore, the area of the segmented hand is not an accurate measure to estimate the depth Z, as this area changes when fingers are flexed. The requirement that the user wears a long sleeved shirt is also a disadvantage of this system. Another critical remark is that an image of segmented patches obtained from a top view of a hand can not provide enough information to recover all the DOFs of each finger. This approach could be more efficient and, perhaps, more reliable, if these angles were modelled as linearly dependent variables (Chua et al., 2002).

Rehg (1995) developed the first well-known work that uses a calibrated system to track unadorned, unmarked hand as 3D high DOF kinematic chain in real-time. A highly articulated (27 DOFs) 3D kinematic hand model was introduced. In that model, finger phalanges are modelled as simple cylinders, fingertips as halves of spheres, and the palm as a couple of planes linking two cylinders. Tracking is performed by non-linear minimisation of the model parameters (state of the model's DOFs). Two feature extractors to measure the sum of squared differences (SSD) were presented:

- deformable templates registration and
- point and line features

In template registration, a cost function based on intensity errors is used to measure the geometric mis-alignment between an input image and the image predicted by the projected kinematic model. A set of templates describes the image appearance of each link. The position of each template in the image is given by the kinematic and camera models as a function of the state. A gradient descent minimisation with a SSD cost function was used in these tracking experiments. Templates provide a useful level of generality, and make it possible to exploit arbitrary texture cues. However, Rehg pointed out that for a specific object like the hand, the constraints provided by the template matching can be approximated by purely geometric error functions involving point and line features.

Point and line features tracking are performed by projecting the middle axes of the truncated cylinders onto the image and by searching for edges in directions perpendicular to the projected segments.

Search for edge in the finger tip is also performed. The residual error between the estimated position of the features and the actual located features is minimised using the Gauss-Newton method to estimate the state update through its Jacobian. The significantly lower computational cost of computing point and line features makes on-line tracking possible.

Rehg deals with self-occlusion through a top-down approach using the knowledge of the model to verify the registration of templates. A state space partitioned into regions of fixed visibility order of fingers is used. It is assumed that a finger can not occlude another and be occluded by it at the same time. This is a reasonable assumption, as fingers are modelled as planar kinematic chains and limits on adduction/abduction do not allow such occlusion configuration. A window function attached to each finger segment masks the contribution of segments that are occluded.

Rehg's system is aided by a dedicated hardware, which included a separate frame grabber per camera (two cameras were used) and a separate computer to asynchronously render and display the model using the estimated state. 10Hz tracking frame-rate is achieved on tracking 19 DOFs (the middle fingers are not tracked) and 6.8Hz in the experiments that tracked all the 27 DOFs. This system does not include occlusion handling, which was tested offline.

Rehg and Kanade (1993) introduced the use of a highly articulated 3D hand model for the tracking of a human hand. Their hand tracking system ***DigitEyes*** is able to recover the configuration of a 27 DOF hand model at speeds of up to 10 Hz. For tracking, the axes of the truncated cylinders that are used to model phalanges are projected onto the image, and local edges are found. Finger tip positions are measured through a similar procedure.

The error between the measured joint and tip locations and the locations predicted by the model is minimised by a nonlinear least squares algorithm (Levenberg-Marquard). The hand motion must avoid self-occlusion, i.e. the finger tips have to be visible in each frame because they present important features. Handling self-occlusion leads to a significant drop in system performance.

Some research effort has been put to design 3D models with more realistic appearance where cylinders have been replaced by more complex structures. Gavrila and Davis' model (1996) is based on superquadrics. Their system uses four orthogonal views to track 22 DOFs of people's body. A decomposition approach and best-first technique is used to search through the high dimensional pose parameter space. This system successfully tracked a sequence with two-person close interaction in the Argentine Tango.

Stenger et al. (Stenger et al., 2001) also use a superquadrics-based hand model. Like Rehg's, this has 27 DOFs but it is built from 39 truncated quadrics. The quadrics are projected as set of conics and an efficient algorithm is used to handle self-occlusion. Once the model is projected, an Unscented Kalman Filter is used to update its pose. Experiments on tracking only 3 DOFs (translations and rotation in the plane) are shown in (Stenger et al., 2001). A frame rate of 3Hz was achieved off-line in a 433MHz PC. Initialisation is done manually and there is no clutter in the background. A simple edge detector is used to provide low level data for tracking. It was shown in (Stenger et al, 2001) that this system is easily scalable from single to multiple views, and from rigid to articulated models. However, it was observed that the computational complexity grows linearly with the number of cameras used.

Even more detailed models have been developed with the purpose of tracking based on kinematic chains. Plaenkers and Fua (2002) use a human body model built from metaballs attached to an articulated skeleton. Metaballs are generalised algebraic surfaces that are defined by a summation over n 3D Gaussian density distributions. The metaballs simulate the gross behaviour of bone, muscles and fat tissue. A polygonal skin surface is constructed by ray casting for display purposes. The model is projected into the images and its silhouette is extracted. Tracking is performed in four steps:

- the silhouette of previous frame serves as initialisation for current frame;
- optimise silhouette using active contours on disparity-filtered gradient image;
- fit body model to stereo data constrained by current silhouette estimate;

- optimise silhouette of fitted model using again active contours.

The optimisation steps use a non-linear least squares estimator to minimise the distance between the observations and the model. A variant of the standard Levenberg-Marquardt least-squares solver was implemented. It can handle large number of unknowns using sparse matrices.

The use of 3D deformable models for tracking is a recent approach that has not been widely explored for hand and body tracking. Such approaches require a batch processing with a set of sample images in order to acquire geometric and deformation parameters of the model. Once the model is acquired, tracking can be performed estimating pose and deformation changes.

Heap and Hogg (1996) proposed a 3D deformable Point Distribution Model that is constructed via Principal Component Analysis (PCA) from samples of hand images in various postures. This yields a mean shape and modes of variation, implicitly capturing possible hand motions. However, because the variations are found using linear interpolation between hand shapes, it is not guaranteed to remain within the space of valid hand motions. The model is a deformable surface mesh, from which the positions of expected contours are derived. For tracking, the model is first deformed using a linear combination of eigenvectors, then rotated, scaled and translated so that its contours match edges in the image. The required training information is extracted semi-automatically from 3D Magnetic Resonance Images. 6 DOFs plus deformation could be tracked at 10 frames per second using a single camera. A weighted least squares approach is used to minimise the error. Ambiguous motions and self-occlusions were not very successfully overcome. If the hand moves to a pose that is not in the training set, the tracker can fail.

Another example of method for tracking with 3D deformable models was proposed by Torresani et al. (Torresani et al., 2001). This is a solution for flow-based tracking and 3D reconstruction of deforming objects in monocular image sequences. They showed that it is possible to approximate 3D motion and deformation using a linear combination of 3D basis shapes. A bound is put on the rank of the tracking matrix and this constraint is used to achieve robust and precise low-level optical flow

estimation without prior knowledge of the 3D shape of the object. The flow matrix can be factored to get the 3D pose, configuration coefficients, and 3D basis shapes. This matrix is factored in an iterative manner, looping between solving for pose, configuration, and basis shapes. Experiments with synthetic data demonstrated the accuracy of this method. They also performed experiments with real data.

Cipolla and Hollinghurst (1996) present a hand tracking system using a 2D deformable model. The system uses uncalibrated stereo vision to track the position and pointing of a hand. Using a ground plane constraint, it is possible to find the indicated position with high speed and accuracy.

Delamarre and Faugeras (1998) pursue a stereo-based approach to hand tracking. A stereo-correlation algorithm is employed to obtain a dense 3D scene reconstruction, by solving the common problem of cluttered background, and a 3D articulated model is fitted to this reconstruction. However, the disparity maps are not very accurate and the costs to obtain a 3D depth map are very high.

A two-step algorithm to estimate the hand pose is proposed by Wu and Huang (1999), first estimating the global pose and subsequently finding the configuration of the joints. The problem of determining the hand pose is formulated as a least median of squares problem, and finding local finger motion is solved as an inverse kinematics problem. A genetic algorithm is used to search the state space. However, the algorithm relies on the assumption that all fingertips are visible.

In (Wu et al., 2001), Wu et al. present a model-based method for capturing articulated hand motion. The constraints on the joint configurations are learned from natural hand motions, using a data glove as input device. It is found that natural hand articulation is highly constrained and that the dimensionality of the hand state space can be reduced using PCA without losing too much information. It is suggested that the lower dimensional state space can be characterised by a union of linear manifolds in 7 dimensional space. Using 28 basis configurations, found by cluster analysis, natural hand articulation are approximated by linear interpolation between two basis configurations. A sequential Monte Carlo tracking algorithm, based on importance

sampling, produces good results, but is view-dependent, and does not handle global hand motion.

A vision based drawing system was proposed by Isard and MacCormick (2000). The 2D shape of a hand is modelled with B-splines and partitioned sampling is used to track the contours of a 7 DOF model in real-time. Partitioned sampling is a method to apply a particle filter to tracking where the configuration space is high dimensional. The idea is to partition the state space, effectively decoupling the motions of different fingers.

The works cited in this section seek to obtain better results by using detailed models of the tracked object. The reader may have noticed that no clear quantitative comparison among the methods was presented. In fact, no ground truth comparison or comparisons between different methods using the same image sequence were found in the publications. Researchers claim that a qualitative visual agreement between the back projected models and the image is the most basic requirement of tracking performance. This is usually the basis for experimental validation of their work. Recently, some authors have been leaving videos available in the Internet to demonstrate tracking results showing the original images and the projection of the model superimposed. However, this does not provide quantitative evaluation of the results. It was found that each work on hand tracking was experienced on image sequences acquired specifically for them, i.e., there is no well known benchmark video for comparison between hand tracking methods.

Another fact that puts in question the non-automatic design of highly detailed hand models is that hands have been used as a source of information for biometric identification (Pankanti et al., 2000). This means that the differences between hands of different people are high enough to be used for person identification. Therefore, a manually-constructed highly detailed hand model may be appropriated for a single user but may not be more effective than a simple model if applied for a different user. The use of a hand model that demands high computational power to be built or a long time to be manually designed for a specific person may not always be appropriate.

CHAPTER 4

A SKIN COLOUR DETECTION MODEL

4.1 Introduction

Any markerless visual method designed to detect and track an object without the intervention of a human operator must confront the question such that, how to associate features observed in the image (pixels, edges, corners, etc) with the object itself. An initial automatic segmentation and localisation must be feasible.

For generic objects in generic surroundings, this is usually not feasible, and constraints must be introduced. For example, one might suppose that the object of interest was the only moving object in the scene, or uniquely composed of planar surfaces, or the only one exhibiting regular texture. The definition of a method for locating hands in images seems to be complicated if one considers that hands are articulated objects that can present both high variation in their shape, and in their degree of self-occlusion. However, there is a reasonable uniformity in human skin colour, allowing the development of a localization method based on pixel colour classification.

The next section explains why skin colour detection is quite robust within and between ethnic groupings, where at first sight there are variations in colour. The most common methods for brightness normalisation are described and compared. Section 4.3 describes the classification method used for distinguishing skin and background from their colours.

4.2 Removing Brightness for Skin Detection

Suppose that image entities are described by feature vector in some N-dimensional space, so that they are assumed to be in one class tend to cluster around a particular position in that space. The accuracy of a classifier is related to degree of overlap between clusters belonging to various classes. These classes are formed from the training data set (Duda and Hart, 1973), (Duda et al., 2000) and some noise. To represent objects, compact clusters in the same class are desirable. Also, that is true for low-dimensional and low-correlated feature spaces where the patterns are represented (Jain et al., 2000).

To classify between skin and not-skin based on colour, pixels corresponding to skin which form such a tight cluster in some colour space are required. Since the appearance of skin has significant variations within ethnic groups, and very substantial variations between ethnic groups, it is natural to ask a question of the robustness of any such classifier. However, the colour of skin is determined by the same agents in any person. The spectral variability is mainly dependent on the amount, density, and distribution of melanin pigment, not on its colour (Martinkauppi, 2002). Thus mainly, it is the brightness of the skin that varies, not its colour (Yang et al, 1998). If images are normalised with respect to brightness, the remaining colour description is indeed compact. Another advantage of normalising the brightness of colour images is the increase in robustness to shadows and changes in illumination intensity.

Brightness normalisation usually involves projecting the colour data into a plane of constant brightness. Several methods have been used for this purpose and, in general, the task is facilitated by representing images in a colour space that decouples colour and grey-scale information. A colour space (also called colour model or colour system) is a specification of a coordinate system and a subspace within that system where each colour is represented by a single point (Gonzalez and Woods, 2000).

By representing data in a colour space that decouples grey-scale and colour information, the task of brightness normalisation is simply done by discarding the grey-scale information.

This provides a lower dimensional representation for colour images (2D rather than 3D as in RGB) without leading to a rise in the overlap between clusters from different classes (skin and background). As a consequence, there is a substantial reduction in the amount of training data required as well as in the amount of storage space and in the time for classification.

Among colour spaces that are decoupled, the most commonly used for skin detection are:

- the CIE Chromatic space,
- the perceptual colour spaces, and
- the YUV/YCbCr colour spaces.

In order to discuss the difference between these colour spaces, it is started by describing the RGB colour space (Section 4.2.1) and follow by describing the other above mentioned colour spaces (Sections 4.2.2, 4.2.3, 4.2.4 and 4.2.5).

4.2.1 The RGB Colour Space

Extensive experiments in the human visual system have showed that the cones - sensors in the eye responsible for colour vision - can be divided into three principal sensing categories, corresponding roughly to red, green and blue (Wyszecki and Stiles, 1967). Therefore, colours are seen as combinations of the so-called primary colours red (R), green (G) and blue (B) (Gonzalez and Woods, 2000).

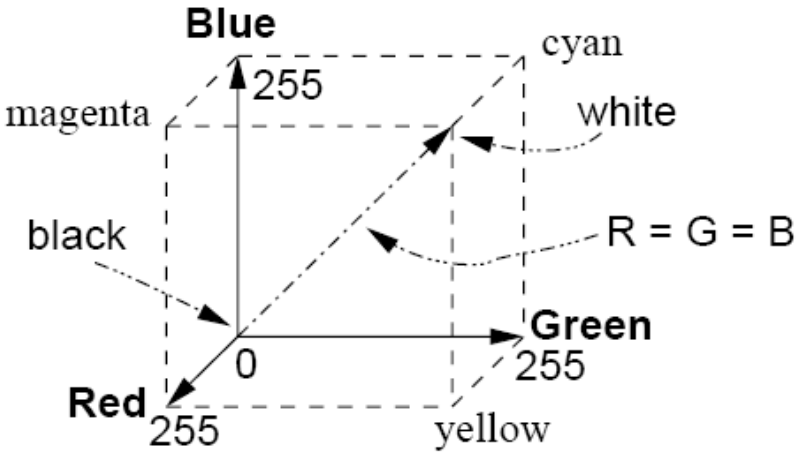


Figure 4.1: RGB colour cube

For this reason, colour displays, monitors, TV receptors and some cameras represent pixels as a triple [R, G, B] of intensities in red, green and blue, respectively, in the RGB colour space. This is also the reason why the RGB space is very commonly used by the computer graphics and image processing community. However, RGB channels are very correlated, as all of them include a representation of brightness. This is illustrated in Figures 4.2 and 4.3, which show that brightness information can be recognised from R, G and B channels shown separately.

Figure 4.2: Sample colour image (a); its gray level version (b)

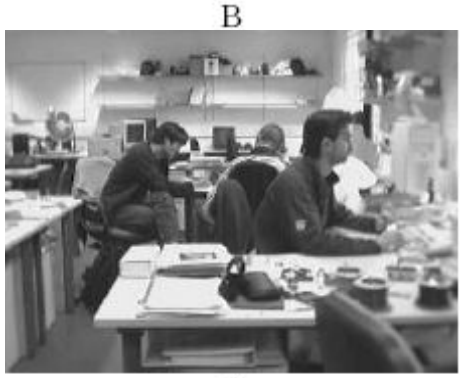
True colour 24 bits RGB images have the triple [R , G , B] represented by 256 discrete values (ranging from 0 to 255) (Jack, 2001). Therefore, the whole range of RGB values forms the RGB colour cube of

$$\left(2^8\right)^3 = 16777216$$

possible values as shown in Figure 4.1.

R

B



G

Figure 4.3: RGB channels of image in Figure 4.2 (a) shown separately

The high correlation between lightness and RGB channels can be noted by the line of the grey values, where R = G = B. In fact, if the corresponding elements in two points, [R1 , G1 , B1] and [R2 ,G2 , B2], are proportional, i.e.,

$$\frac{R_1}{R_2} = \frac{G_1}{G_2} = \frac{B_1}{B_2} \tag{4.1}$$

They have the same colour, but different brightness (Yang and Waibel, 1996).

4.2.2 The CIE Chromatic Space

The CIE chromatic space is a standard proposed in 1931 by the Commission Internationale de l'Eclairage -- the International Commission on Illumination. It has been used in several colour processing tasks (Gonzalez and Woods, 2000) and it is used to define the colour gamut, i.e., the range of possible colour values that a device can represent.

This two dimensional space has the x and y axes respectively defined by the chromatic colours red and green (r; g). Chromatic colours, known as “pure" colours in the absence of brightness, are defined by a normalisation process:

$$r = \frac{R}{R+G+B}$$
$$g = \frac{G}{R+G+B} \tag{4.2}$$

Pure blue (b) is redundant after the normalisation because (r + g + b = 1) (Wyszecki and Stiles, 1967). The use of this colour space for skin detection has became popular specially after the work on face tracking developed at SCS, Carnegie Mellon University (Yang and Waibel, 1996), (Yang et al., 1998).

4.2.3 The Perceptual Colour Space

The perceptual colour spaces were designed by Smith (Smith, 1978) in order to provide a more “intuitive" way of describing colours and lightness. Three quantities are used to define them: hue, saturation and brightness.

- Brightness embodies the achromatic notion of intensity.
- Hue is an attribute associated with the dominant wavelength in a mixture of light waves. It represents colour as perceived by an observer. Thus, when we call an object blue, yellow or red, we are specifying its hue.
- Saturation refers to the relative purity or the amount of white light (or grey of equal intensity) mixed with a hue.

Primary colours (pure red, green and blue) are fully saturated, whereas colours such as pink (red and white) and lavender (violet and white) are less saturated. The degree of saturation is inversely proportional to the amount of white light added (Gonzalez and Woods, 2000).

Basically, there are two distinct perceptual colour spaces: HSL and HSV.

- HSL (hue, saturation, lightness)
- HSV (hue, saturation, value)

Both are defined with polar coordinate systems.

HSV is represented by a hexcone where

- Hue is the angle around the vertical axis,
- S is the distance from the central axis
- V is the distance along the vertical axis.

Primary and secondary pure colours are fully saturated (S = 1).

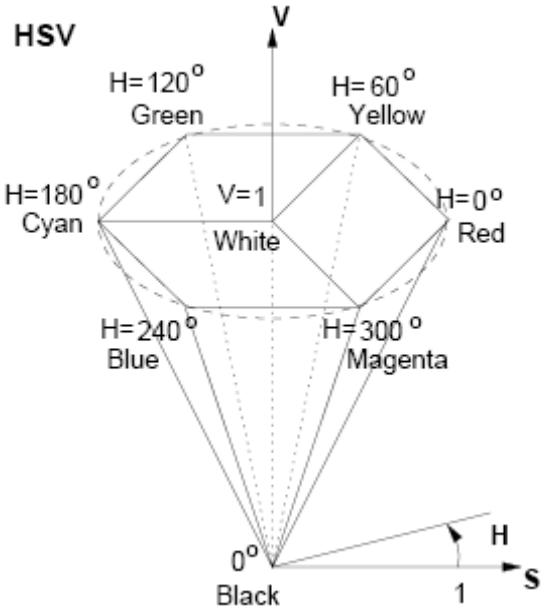


Figure 4.4: HSV colour space

As illustrated in Figure 4.4, starting from H = 0^o (which represents pure red), a secondary or primary colour is located at each 60^o of hue. Complementary colours are 180^o opposite one another measured by H. Colours along the vertical axis have zero saturation, i.e., gray scale values. Note that when S = 0, the value of H is irrelevant (Jack, 2001).

While illustrating how HSV colour space provides an alternative way of describing colours, any colour (defined by H) with V = 1 and S = 1 is assumed as a pure pigment. Adding some amount of white corresponds to decreasing S value and adding black corresponds to decreasing V value. Tones are created by decreasing both S and V (Smith, 1978).

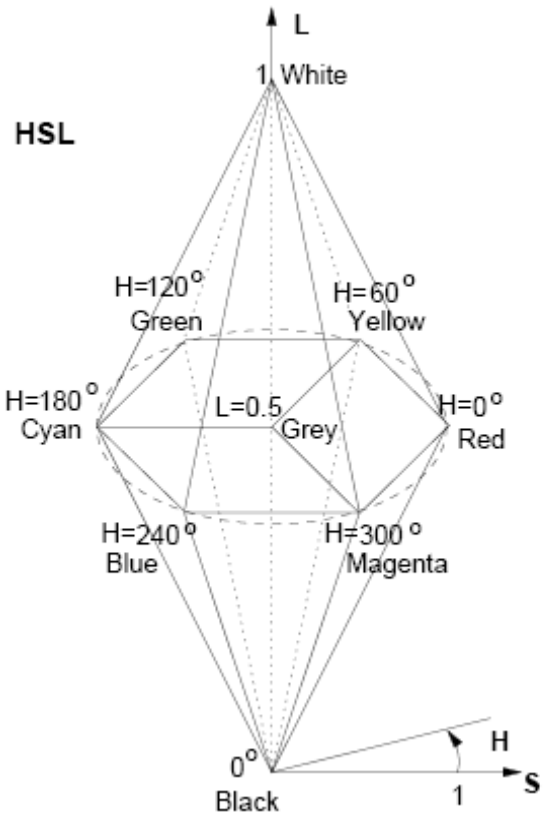


Figure 4.5: HSL colour space

HSL colour space is a double hexcone and can be thought of as a deformation of the HSV space. The distinction between HSV and HSL lies in the representation of brightness information, which determines the distribution and dynamic range of both the brightness (L or V) and saturation (S). In practice, the HSL colour space is best for grey level image processing and also for representing objects in such a way that colour images can be distinguished even in monochrome images (e.g. showing colour cartoons on black-and-white TV receivers), whereas the HSV image space is a better representation for colour processing (Jack, 2001).

As described in (Raja et al., 1998), (Azarbayejani and Pentland, 1996), and (Zhu et al., 2000), on performing skin detection, the brightness channel is discarded and the HS space is used instead. Therefore, there is no significant difference between HSV

and HSL in applications (Bowden, 1999). Figure 4.6 shows the H, S and V channels obtained from image from Figure 4.2(a).

4.2.4 The YUV Colour Space

The YUV colour space was created in order to make colour television broadcasts backwards-compatible with black and white TV receivers. The colour signal also needed to conserve bandwidth because three channels of RGB data would not fit into the limited broadcast signal bandwidth. By combining colour information, YUV uses far less bandwidth than RGB and maintained compatibility with black and white TVs.

- The Y channel describes Luminance, the range of value between light and dark, which is the signal seen by black and white televisions.
- The U and V chrominance channels subtract the Luminance values from Red (U) and Blue (V) to represent the colour only information (without brightness) (Maller, 2002).

H

S

V

Figure 4.6: HSV channels of image in Figure 4.2 (a) shown separately

Then, the basic conversion equation from RGB to YUV is:

$$\begin{aligned} Y &= 0.3R + 0.6G + 0.1B \\ U &= B - Y \\ V &= R - Y \end{aligned} \tag{4.3}$$

The coefficients used to obtain Luminance are the same as those used for the NTSC standard conversion from RGB to grey level images (Poynton, 1996).

It is shown that approximately

- 65% of all the cones in the human eye are sensitive to red light,
- 33% are sensitive to green light
- 2% are sensitive to blue, but the blue cones are the most sensitive (Wyszecki and Stiles, 2000)

As in HSV, brightness normalisation is done by discarding one of the channels. In this case, the UV channels are kept.

4.2.5 The YCbCr Colour Space

The YCbCr colour space was developed as part of ITU-R BT.301 during the development of a world-wide digital component video standard. This colour space was extensively used in the development of the JPEG standard, and was used for skin colour detection by several research projects, including the Pfinder (Wren et al., 1997).

As shown in equation 4.4, YCbCr is a scaled and zero-shifted version of the YUV, so that the chrominance values are always positive (Pennebaker and Mitchell, 1993):

$$Cb = \frac{U}{2} + 0.5$$

$$Cr = \frac{V}{1.6} + 0.5 \tag{4.4}$$

for U and V in the range [0, 1]. For digital values of U and V, a 128 shift is employed, rather than 0.5. So, white colour representation in this colour space is such that Y = 255 , Cb = 128 and Cr = 128.

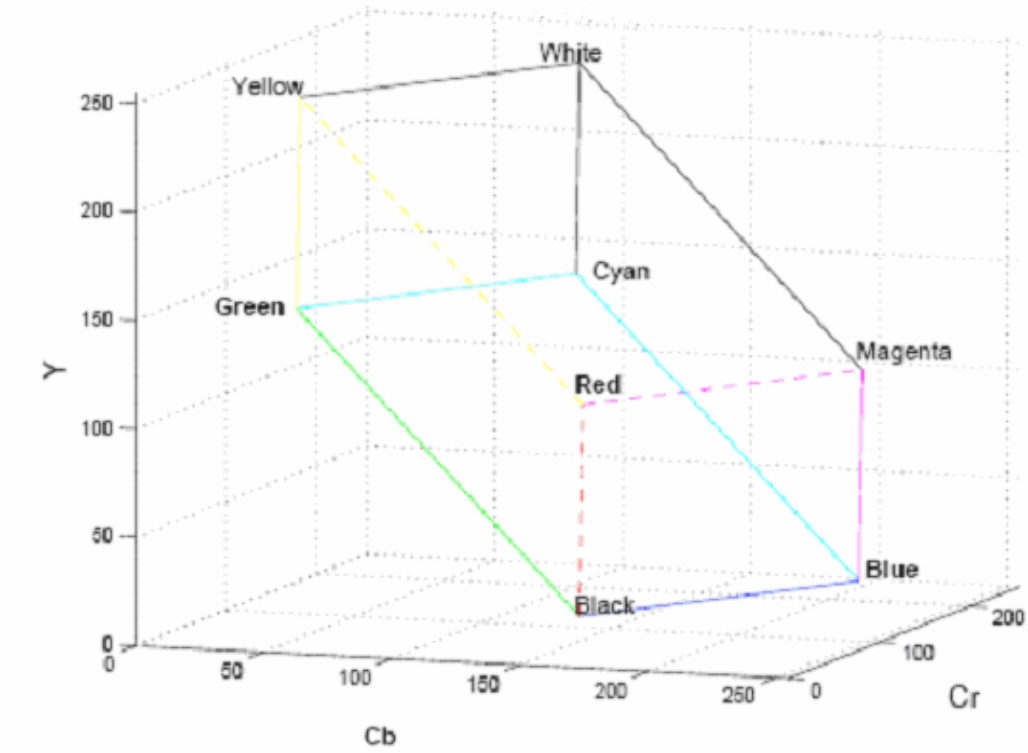


Figure 4.7: RGB colour cube in the YCbCr colour space

Figure 4.7 shows the RGB colour cube in the YCbCr colour space. It shows that not all the possible values in the triple [Y, Cb, Cr] (for each of then ranging from 0 to 255) represent possible RGB colours. Therefore, special care must be taken to about overflow or underflow in YCbCr to RGB conversion implementations. As in HSV, brightness is normalized for skin detection by discarding one of the channels. In this case, the CbCr channels are kept.

4.2.6 Colour Spaces Comparison Results (with Normalisation)

Previous sections have described RGB and the four most common colour spaces that are used for brightness normalisation for skin detection. Here, some images are shown in order to illustrate their properties (colour spaces normalisations for the Figure 4.2(a)).

- Figure 4.8(a) shows the after brightness normalisation by representing its colours using its pure colour representation (r, g and b) as defined in equation 4.2.
- Figure 4.8(b) shows the after brightness normalisation in HSV space The brightness had been set to 127 to reconstruct the normalised images
- Figure 4.8(c) shows the after brightness normalisation in YCbCr space. The brightness had been set to 127 to reconstruct the normalised images

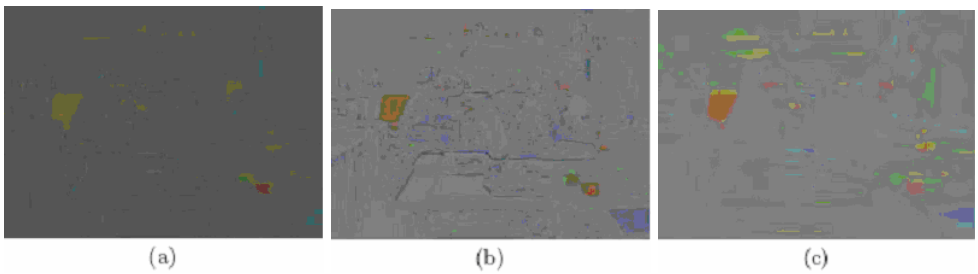


Figure 4.8: After removing brightness, (a) using CIE colour space (b) using HSV colour space (c)YCbCr colour space

4.3 Classifying Pixels for Skin Detection

Given a set of colour images containing projections of the objects which we need to identify, it is possible to train a classifier based on the H and S image space so that we can determine which pixels of a new image belong to which of the known objects. (Training and classifying are the terms related with neural networks. But within this thesis, there is no any neural network approach. So, these terms are used in general meaning.)

Several methods have been applied to this problem (Duda et al., 2000). The simplest methods “manually" define certain range of values in a colour space to be classified as skin colour (Chai and Ngan, 1998). The obvious problem of this approach is the lack of ability to adapt to variations of the data.

4.3.1 Histogram-Based Classifier

For performing classification, the data is represented in the HS of HSV colour space, which is a two dimensional space. In practical applications, each channel of the HS space can have discrete values. These properties of the HS feature space allow the use of a histogram-based classifier. The first step on training this classifier is to build a colour model using a histogram in the feature space for each class. In our case, the feature space is the HS colour space (from HSV). Each channel (or feature) of the space taken separately has its number of possible values represented by a set of intervals called bins.

4.3.2 Training Method

The training process consists of selecting skin regions of images. This task is performed by a human operator, who usually does not know how important skin region being selected is for the training data. A skin patch may be more important and generic for training.

For hand tracking via a skin colour model, skin regions from the users can be sampled by prompting them to center their face or hand in an onscreen box, or

selecting skin related region by a mouse. The hues and saturations derived from the skin pixels in the image are sampled from the H channel and S channel, and then binned into a 1D histograms. When sampling is complete, the histograms are saved for future use.

Clearly, sampling flesh hues from multiple people may make more robust histograms. However, simple skin histograms tend to work quite well with a wide variety of people. A common misconception is that different colour models are needed for different races of people, for example negroids or caucasians. That is simply not true. All human skin is much the same hue. Dark-skinned people simply have more skin colour saturation than light-skinned people, and these differences are largely removed in the HSV colour system.

CHAPTER 5

CONTINUOUSLY ADAPTIVE MEAN SHIFT ALGORITHM

5.1 Introduction

As a first step towards a perceptual user interface, colour tracking algorithms are being developed. The algorithms that are intended to form part of a perceptual user interface must be fast and efficient. They must be able to track in real time yet not absorb a major share of computational resources. For example, other tasks must be able to run while the visual interface is being used. They must deal with

- irregular object motion due to perspective (near objects to the camera seem to move faster than distal objects);
- image noise;
- distracters, such as other faces, hands and etc. in the scene;
- occlusion by hands or other objects;
- lighting variations.

One of them is based on a robust nonparametric technique for climbing density gradients to find the mode (peak) of probability distributions. This is called the Mean shift algorithm (Fukunaga, 1990). In fact, The Mean shift algorithm was not developed intending to be used as a tracking algorithm, but it is quite effective in this role. The Mean shift algorithm operates on probability distributions. To track coloured objects in video frame sequences, the colour image data has to be represented as a probability distribution. Colour distributions derived from video image sequences change over time.

Mean shift algorithm is efficient for handling static colour probability distributions. However in video sequences, this distribution is changing dynamically. Therefore, to deal with dynamically changing colour probability distributions derived from video

frame sequences, the mean shift algorithm has needs to be modified. The modified algorithm is called the Continuously Adaptive Mean Shift (CAMSHIFT) algorithm (Bradski, 1998).

The Camshift algorithm is a generalization of the Mean shift algorithm. Camshift operates on a 2D colour probability distribution image produced from histogram back-projection. When objects in video sequences are being tracked and the object moves, the size and location of the probability distribution changes in time. The Camshift algorithm adjusts the search window size in the course of its operation. For each video frame, the colour probability distribution image derived from that frame is tracked. Within this track the center and the size of the colour object are found with the Camshift algorithm as well. The current size and location of the tracked object are reported and used to set the size and location of the search window in the next video image. The process is then repeated for continuous tracking. Instead of a fixed or externally adapted window size, Camshift relies on the zeroth moment information, extracted by the algorithm itself to continuously adapt its windows size within or over each video frame.

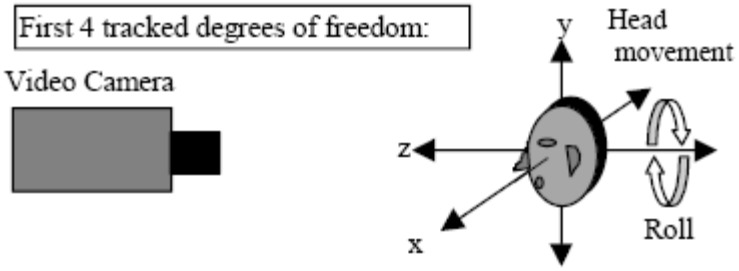


Figure 5.1: First four head tracked degrees of freedom:
X, Y, Z location and head roll

For example if face tracking (for easy explanation and being mostly used area in Camshift applications) is considered, Camshift tracks the X and Y coordinates and the area of the flesh colour probability distribution representing a face. Area is proportional to Z, the distance from the camera. Head roll is also tracked as a further degree of freedom. So, X, Y, Z, and Roll derived from Camshift face tracking can be used as a perceptual user interface for controlling commercial computer games and for exploring 3D graphic virtual worlds (Figure 5.1).

5.2 Colour Probability Distributions

For Camshift, "colour probability distribution" is very important. To be able to track coloured objects in a video scene, a probability distribution image of the desired colour in the video scene must be created. In order to do this, a model of the desired hue, generally, using a colour histogram is created. Hue is a member of Hue Saturation Value (HSV) colour system. (See Chapter 4 for Colour Space Systems' details.)

Figure 5.2: A video image and its flesh probability image. (Bradski, 1998)

The probability distribution image can be thought to be found by the transformation of input image pixels with some weight coefficients. If a pixel belongs to input image matches the tracked object's colour (it is flesh for Figure 5.2) then that pixel is transformed to the second result image (probability distribution image) with multiplying with related coefficient. (The higher relationship results bigger pixel value in gray level). If it does not match, then the coefficient is zero and the effect of that pixel is discarded.

5.3 CAMSHIFT Derivation

The closest existing algorithm to Camshift is known as the Mean shift algorithm. The mean shift algorithm is a non-parametric technique that climbs the gradient of a probability distribution to find the nearest dominant mode (peak). The steps of the Mean shift algorithm are given below;

1. A window W is chosen at size s.
2. The initial search window is centered at data point p_k
3. Compute the mean position within the search window

$$\hat{p}_k(W) = \frac{1}{|W|} \sum_{j \in W} p_j \tag{5.1}$$

4. Center the window at point

$$\hat{p}_k(W) \tag{5.2}$$

5. Repeat Steps 3 and 4 until convergence. Decision for convergence can be made with

$$\hat{p}\ (W) - p_{k} \cong 0 \tag{5.3}$$

For discrete 2D probability distribution image, the mean location (the centroid) within the search window (Steps 3 and 4 above) is found as follows

Find the zeroth moment

$$M_{00} = \sum_x \sum_y I_p(x,y) \tag{5.4}$$

Find the first moment for x and y

$$M_{10} = \sum_x \sum_y x I_p(x,y)$$

$$M_{01} = \sum_x \sum_y y I_p(x,y) \tag{5.5}$$

Then the mean search window location (the centroid) is

$$x_c = \frac{M_{10}}{M_{00}} \qquad y_c = \frac{M_{01}}{M_{00}} \tag{5.6}$$

where $I_p(x,y)$ is the pixel (in probability distribution image) value at position (x,y) in the image, and x and y range over the search window

Unlike the Mean Shift algorithm, which is designed for static distributions, Camshift is designed for dynamically changing distributions. The Camshift algorithm adjusts the search window size in the course of its operation. Initial window size can be set at any reasonable value. Instead of a set or externally adapted window size, Camshift relies on the zeroth moment information, extracted as part of the internal workings of the algorithm, to continuously adapt its window size within or over each video frame.

Continuously Adaptive Mean Shift (Camshift) algorithm is given below;

1. Choose the initial location of the 2D mean shift search window.
2. Calculate the colour probability distribution in the 2D region centered at the search window location in an area (ROI – Region of Interest) slightly larger than the mean shift window size.

3. Run mean shift algorithm to find the search window center. Store the zeroth moment (area or size) and center location (centroid - mean location).
4. For the next video frame, center the search window at the mean location stored in step 3 and set the window's size to a function of the zeroth moment found there. Go to Step 1.

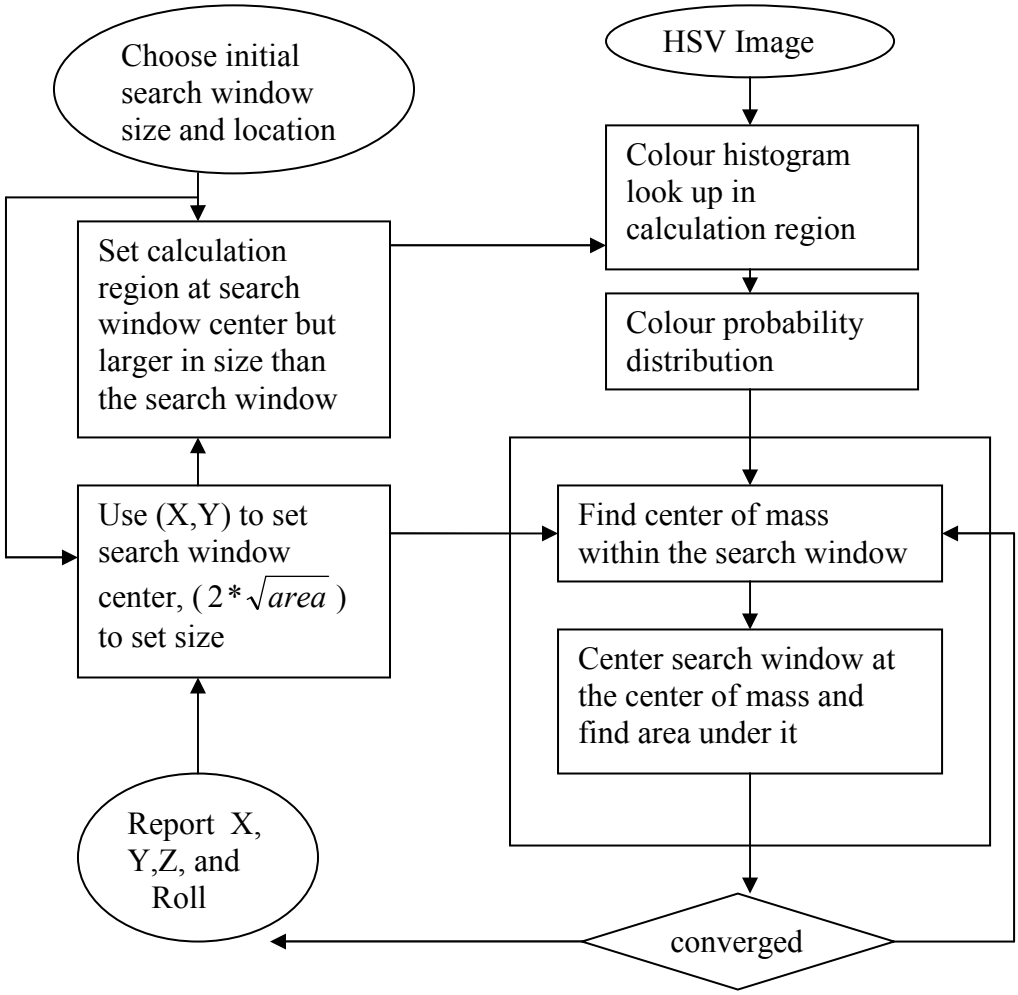


Figure 5.3: Block diagram of colour object tracking

Calculation of 2D Orientation

The 2D orientation of the probability distribution is also easy to obtain by using the second moments during the course of Camshift's operation where (x,y) is the range over the search window, and $I_p(x,y)$ is the pixel value at (x,y):

Find the first moment for x and y

$$M_{20} = \sum_x \sum_y x^2 I_p(x,y)$$

$$M_{02} = \sum_x \sum_y y^2 I_p(x,y) \tag{5.7}$$

Then the object orientation (major axis) is

$$\theta = \frac{1}{2}\arctan\left(\frac{2*\left(\frac{M_{11}}{M_{00}} - x_c y_c\right)}{\left(\frac{M_{20}}{M_{00}} - {x_c}^2\right) - \left(\frac{M_{02}}{M_{00}} - {y_c}^2\right)}\right) \tag{5.8}$$

The first two Eigenvalues (major length and width) of the probability distribution image found by Camshift may be calculated in closed form as follows. Let

$$a = \frac{M_{20}}{M_{00}} - {x_c}^2 \tag{5.9}$$

$$b = 2*\left(\frac{M_{11}}{M_{00}} - x_c y_c\right) \tag{5.10}$$

$$c = \frac{M_{02}}{M_{00}} - {y_c}^2 \tag{5.11}$$

Then length l and width w from the distribution centroid are

$$l = \sqrt{\frac{(a+c)+\sqrt{b^2+(a-c)^2}}{2}} \tag{5.12}$$

$$w = \sqrt{\frac{(a+c)-\sqrt{b^2+(a-c)^2}}{2}} \tag{5.13}$$

When used in face tracking, the above equations give us head roll, length, and width as marked in Figure 5.4.

Figure 5.4: Orientation of the flesh probability distribution marked on the source video image. (Bradski, 1998)

CHAPTER 6

CAMERAS, COORDINATE SYSTEMS AND STEREO VISION

6.1 Introduction

Camera is the basic equipment in vision applications. According to the properties of desired applications, the number of the cameras used can be one, two or more. Whatever it is, the basic and main task is to get frames and provide them to the system that will perform process. Usually, these systems are computer-based systems. In this chapter, the basic notations, descriptions and relevant equations for vision applications with the cameras, computer systems and the other things are explained.

Section 6.2 describes the basic and commonly used geometry entities and their representations. Section 6.3 presents a general view on the camera model used. Section 6.4 presents coordinate systems in general and the relationships between them. The representation of points and their transformations are explained. And finally, the necessary explanations about the camera parameters are introduced. In section 6.5, idea of stereo imaging and 'Epipoler geometry' are explained.

For a more complete and detailed discussion of this topics and more information about these, one can refer to (Özuysal, 2004).

6.2 Projective Geometry

In vision-based applications while the objects in images are represented in 2D, the actual objects belong to 3D. In fact, the objects in images are the projection of objects in 3D to image plane in 2D. In this thesis, homogeneous coordinates will be used for the representation of these geometric entities.

In homogeneous coordinate systems, a 2D point '**x**' and a 3D point '**X**' are denoted as follows.

- $\mathbf{x} = [\, x\ y\ 1 \,]^T$
- $\mathbf{X} = [\, x\ y\ z\ 1 \,]^T$

A projective transformation of 3D is represented by a 4x4 invertible matrix '**T**'.

6.3 Camera Model

3D world points are mapped onto the 2D image plane by cameras. For this mapping, the most commonly used camera model is *pinhole camera model.* This camera model is very simple and basic. So it is widely used for realizing vision systems (Figure 6.1). It will be explained in more detail in the next section.

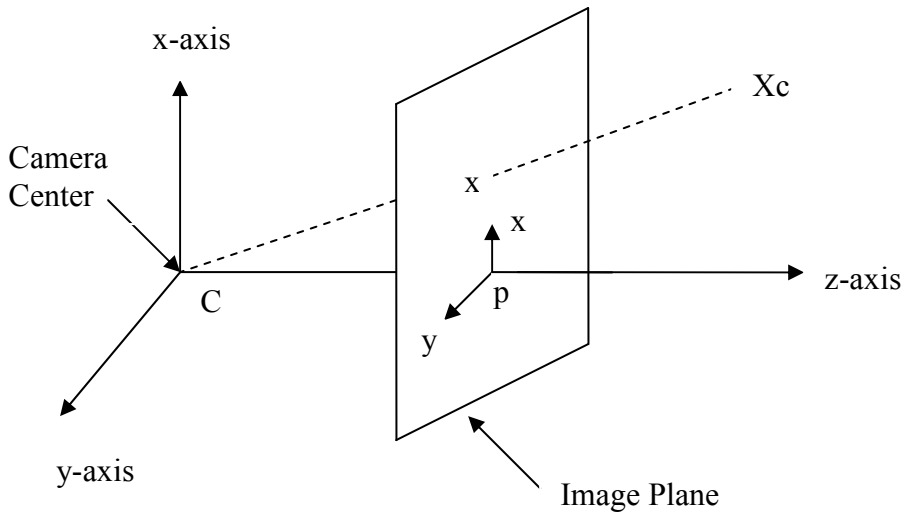


Figure 6.1: Camera coordinate system and projection of a 3D point in real world on the image plane

6.4 Coordinate Systems and Two View Geometry

First of all, coordinate systems should be introduced. This introduction helps to develop clear model of camera geometry.

- World Coordinate System in which the real 3D object takes place.
- Camera Coordinate System is described as its center is camera center. In this coordinate system, the imaging plane is parallel to the 'xy' plane (shown as in Figure 6.1) and placed at a distance '*f*' on the z-axis. '*f*' is called as 'focal length'. The point at which the z-axis intersects the imaging plane is called 'principal point' (shown in Figure 6.1 as '*p*'). The perspective projection equations are valid in this type coordinate system. If point $\mathbf{X}_C$ is a point in this coordinate frame with homogenous coordinates(X_C , Y_C , Z_C , 1) then its projection can be written as

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X_C \\ Y_C \\ Z_C \\ 1 \end{pmatrix} \tag{6.1}$$

  where the projected point **x** is in 2D homogeneous coordinate system in the imaging plane with the principal point as its origin.

- Pixel Coordinate System is used in the computer systems. It is used to represent the points in '*pixe*l' measurement. So, a transformation from the imaging plane coordinates to pixel coordinates is required. The resulting projection matrix can be written as

$$\mathbf{P} = \begin{pmatrix} f_x & s & p_x & 0 \\ 0 & f_y & p_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{6.2}$$

Aspect ratio is equal to f_y / f_x .

S is called as *skew* and represents a tilted imaging plane.

p_x and p_y represents an amount of shift in the origin of the coordinate system.

The projection matrix '**P**' is called the '*Camera Matrix*' and can be seen as

a matrix composed by two types of calibration parameters.

- The Internal parameters are those that specify optical properties of the camera. These parameters are

  '*focal length (f)*' ,

  '*principal point coordinates* (p_x, p_y)',

  '*aspect ratio*' and '*skew*'

- The external parameters are those that specify the pose of the camera in the world coordinate system (Figure 6.2). These parameters are '*Rotation Matrix*' and '*Translation Vector*'.

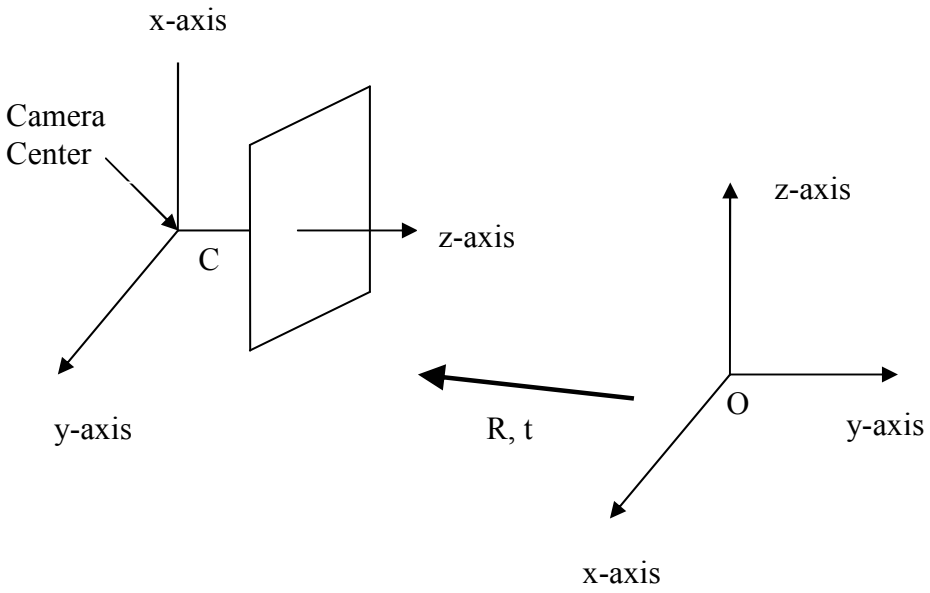


Figure 6.2: The coordinate transformation between the world and camera coordinate frames

With those explanations, camera matrix can be expressed as given below.

$$\mathbf{P} = \begin{pmatrix} f_x & s & p_x & 0 \\ 0 & f_y & p_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{6.3}$$

$\mathbf{P} = \mathbf{K}\left[\mathrm{I}_{3x3} \mid 0\right]$ where K is called '*the internal calibration of the camera*'.

In order to do measurements in '*world coordinate system*' some transformations should be performed. Because up to now, the measurements for 3D points are assumed to be taken in the camera coordinate system. This transformation can be made by

$$\mathbf{X}_C = \begin{pmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix} \mathbf{X}_W \tag{6.4}$$

where $\mathbf{R}_{3x3}$ is the rotation matrix, $\mathbf{t}_{3x1}$ is the translation vector,

$\mathbf{X}_C$ is the point defined in the camera coordinate system and

$\mathbf{X}_W$ is the same point defined in the world camera system.

So the resulting camera matrix is

$$\mathbf{P} = \begin{pmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix} \tag{6.5}$$

Hence the projection of a point in 3D on the image plane can be expressed as follows

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{R}^T & -\mathbf{R}^T\mathbf{t} \\ \mathbf{0}^T & 1 \end{pmatrix} \begin{pmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{pmatrix} \tag{6.6}$$

6.5 Stereo Vision

The camera model and coordinate systems presented in the previous chapters are used to analyze the projection of the 3D points and the resulting image. In fact we have 2D information about the objects in computer systems. So, the inverse of that problem should be investigated. This new problem is to get 3D information from two or more 2D images.

Let's assume that, there are two images. These images are taken by two cameras and are of the same scene. Let the related camera matrices be P and P'. To get the depth information of a point X in 3D, the pixel coordinates of its projections, x and x' should be found. If these pixel coordinates are found, point X lies at the intersection of the back projected lines of them. By the way, these points are called 'corresponding points'.

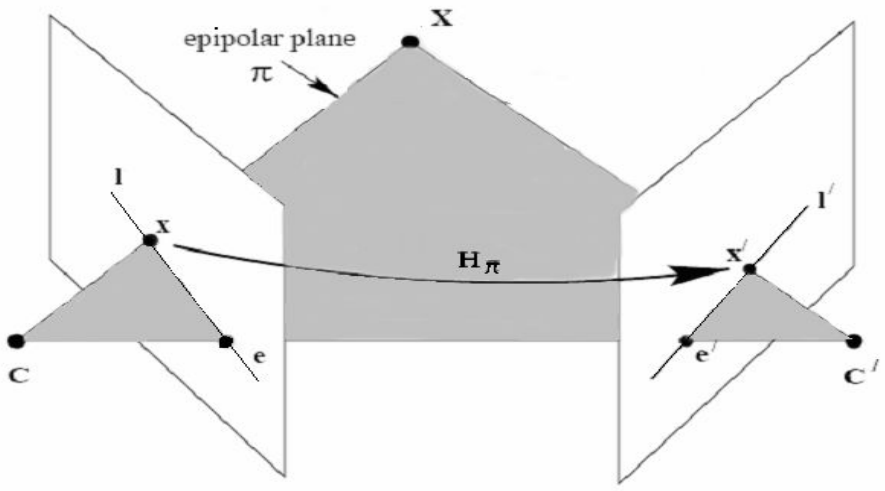


Figure 6.3: A point **x** in the left image and its corresponding point **x'** in the right image can be used to reconstruct point **X.**

In Figure 6.3, some points are shown. Briefly,

- The two cameras are indicated by their centers C and C' and image planes. The camera centers, X point in 3D and its images x and x' lie in a common plane 'π' called '*epipolar plane'*.
- The '*epipole (e) and (e')'* are the points corresponding to intersection of the line joining the camera centers (the baseline) with the image plane.
- An '*epipolar line (l')*' is the intersection of an epipolar plane with the image plane. All epipolar lines intersect at the epipole. An epipolar plane intersects the left and right image planes in epipolar lines, and defines the correspondence between the lines.

Finally, this geometry is usually motivated by considering the search for corresponding points in stereo matching. In terms of a stereo correspondence algorithm the benefit is that the search for the point corresponding to x need not cover the entire image plane but can be restricted to the line *l'*.

CHAPTER 7

EXPERIMENTAL IMPLEMENTATION AND DETAILS

In this thesis, a stereo-based hand tracking in 3D is studied. For this purpose, a stereo-camera system is used. The system was designed in the Computer Vision and Artificial Intelligence Laboratory of the Electrical and Electronics Engineering Department of the Middle East Technical University. The analog cameras are attached to a planar platform almost in parallel form. The video signals from the cameras are then connected to the frame grabber card installed on a personal computer. The cameras are assumed to be working continuously and the time difference between 2 frames is minimum to preserve synchronization as well.

Figure 7.1: Hand image with markers

In order to get hand regions such as fingers and individual finger parts, some coloured markers are needed to be used. In this study, two different coloured markers are used. One of them (green marker) is used to index fingers from thumb to little finger. The other one (blue marker) is used to index individual finger parts from palm to the outer segments (Figure 7.1).

Although, it is very well known that brightness normalisation can improve the distinction between skin and background, it is not clear which colour space is the best. In fact, several works have been done on comparing colour spaces for skin detection (Martinkauppi, 2002). However, no definitive conclusion can be drawn, because each work was done using different illumination conditions. Since the aim of this thesis is to implement a system that works on-line, it is important to minimise the processing time for each frame grabbed by the cameras. So the HSV colour space is chosen since it is obtained directly by performing transformation from RGB space.

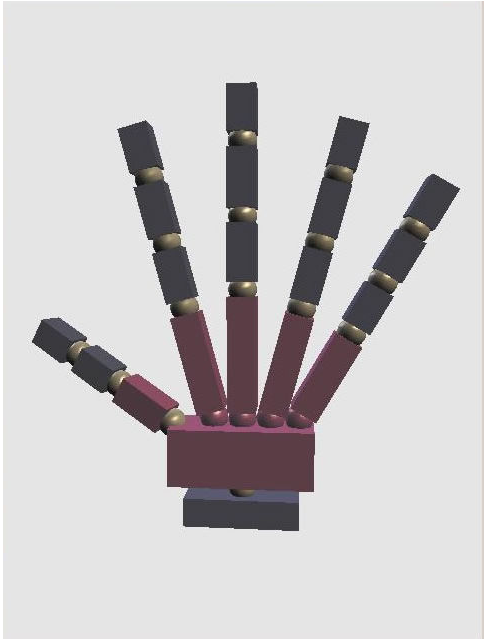


Figure 7.2: 3D hand model generated in JAVA

Visualization is done by JAVA. A required 3D model is defined in JAVA (Figure 7.2). That model consists of ellipsoids and prisms. Ellipsoids are used as joints between finger parts. Prisms are used to represent palm and the fingers as well. After a frame is processed and completely analysed, information for the tracked object's movement is obtained.

This information consists of all points and hand regions' coordinate knowledge (both in 2D and 3D) for the fingers, angles between the fingers and the palm, angles between the fingers and angles between finger parts and the joints. According to the tracking result, the 3D coordinate information and joint angles are sent to the JAVA program and 3D hand model is updated with this new data.

So, a module-based coding is done and top down approach is followed. In the following lines, all the process is summarised with flowcharts. The details and the related explanations are presented. Table 7.1 shows the hardware and software setup used in this work. Figure 7.1 shows a hand image with markers.

Table 7.1: Hardware and Software setup used

System	**Properties**
PC Desktop System	Intel Pentium 4 2.66GHZ 512 MB Ram Windows XP Professional
Analog Camera	Sony FCB-IX47AP
Frame Grabber Card	Matrox Meteor II/Standart
IDE Platform	Borland C++ Builder 6.0 Enterprise (Main) Microsoft Visual C++ 2003 Borland Java Builder 7.0
Software Libraries	Intel OpenCV Library 4.0 Matrox MIL-Light 7.5

Flowchart of the Process

The overall process is summarized in the chart given in Figure 7.3 and the details are explained in the following.

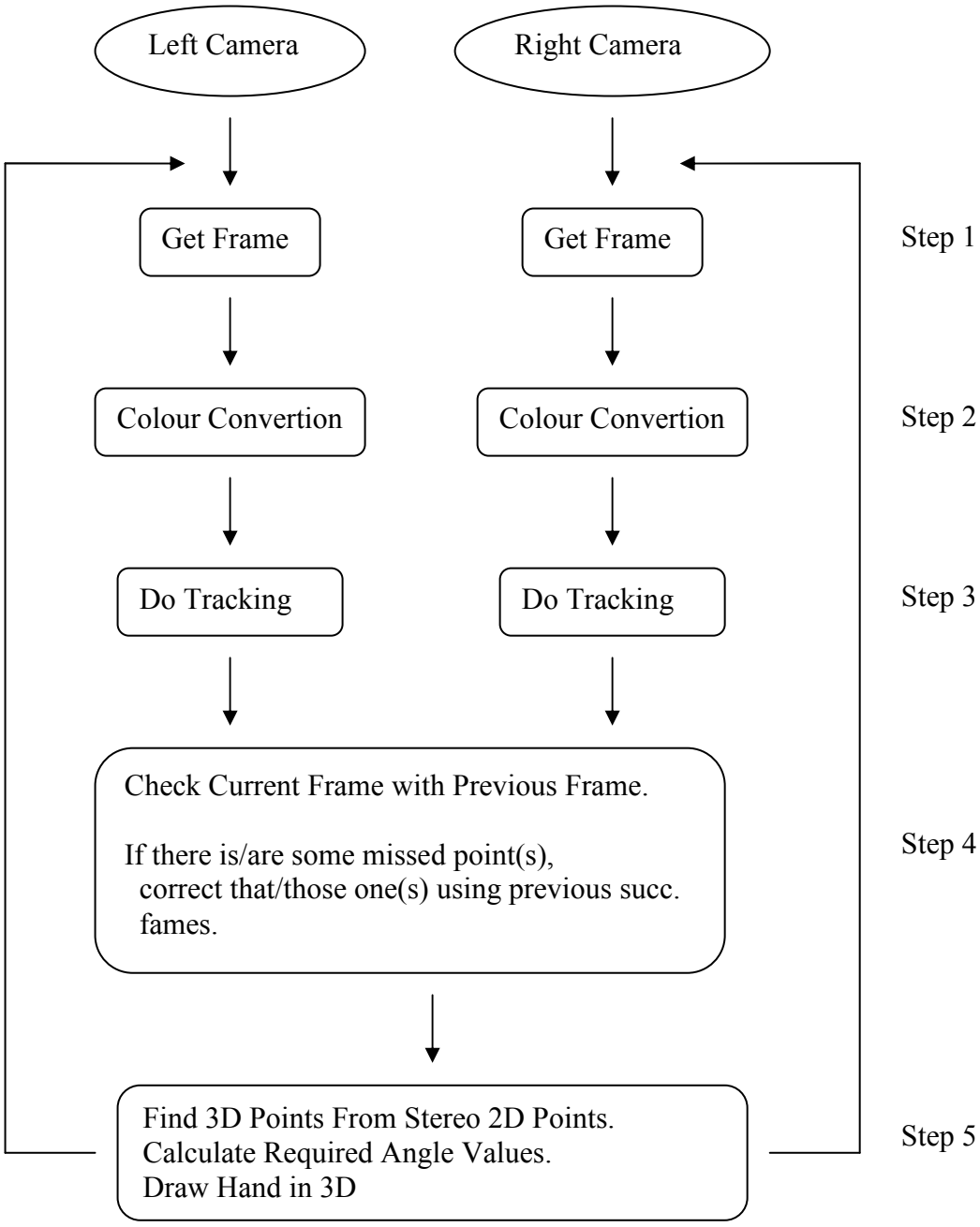


Figure 7.3: Outer loop of the whole process.

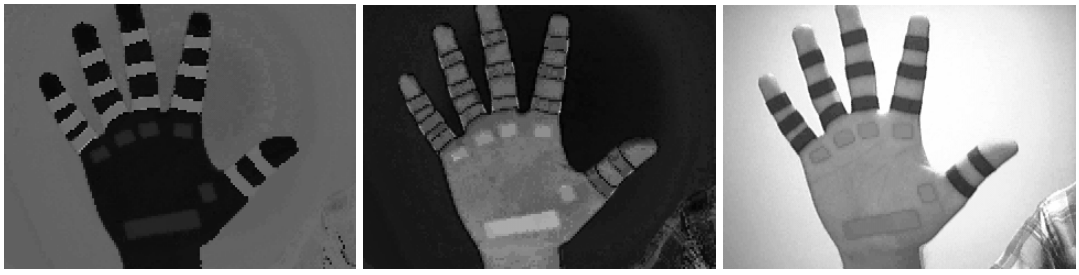
Step 1) Get Frame: By choosing appropriate parameters, overall system can be achieved to run on-line mode or off-line mode. But current system is more suitable for off-line mode. Off-line mode means working with pre-recorded video files. These video files are stereo pairs. Figure 7.4 shows an image taken at this step which is the first and main image.

Figure 7.4: Input image taken from the camera

Figure 7.5: Gray mode of input image

Step 2) Colour Convertion: After getting frames which are in RGB colour space, they are converted to the 'gray mode' and 'HSV colour space'. The rest of the process uses these converted images. (see Figure 7.5, Figure 7.6 and Figure 7.7)

Figure 7.6: Input image in HSV Colour Space



(H) (S) (V)

Figure 7.7: HSV channels of the input image

Step 3) Do Tracking: This is the heart of the whole process and all necessary steps for tracking are being done in this step. A detailed explanation will be given is given in the next page.

Step 4) Check Current Frame with …: If everything goes all right, all necessary points of the hand will be available. But unfortunately, there are so many miss-placed or lost points. To recover these points, it is necessary to check the current frame and its information with some previous successful frames. If a point is found correctly or by assumption then that frame is a successful frame for that point. When that point is lost later, the last information related to this point which was saved in the memory will be used.

Step 5) Find 3D Points From…: After getting frames from cameras, it is necessary to find the corresponding point in 3D from stereo images. Initially the camera calibration is assumed to be done. By using this calibration information, a point in one camera frame with its corresponding point in the other camera can be projected to 3D.

Calculate Required Angle…: In order to fit the resulted 3D points to the 3D hand model, some calculations are required. The relationship among these points is adequate. The relationship consists of some angle values and lengths.

Draw Hand…: At the final step, in order to see what is going on, the tracked hand must be drawn on to the screen.

Step 3) Do Tracking

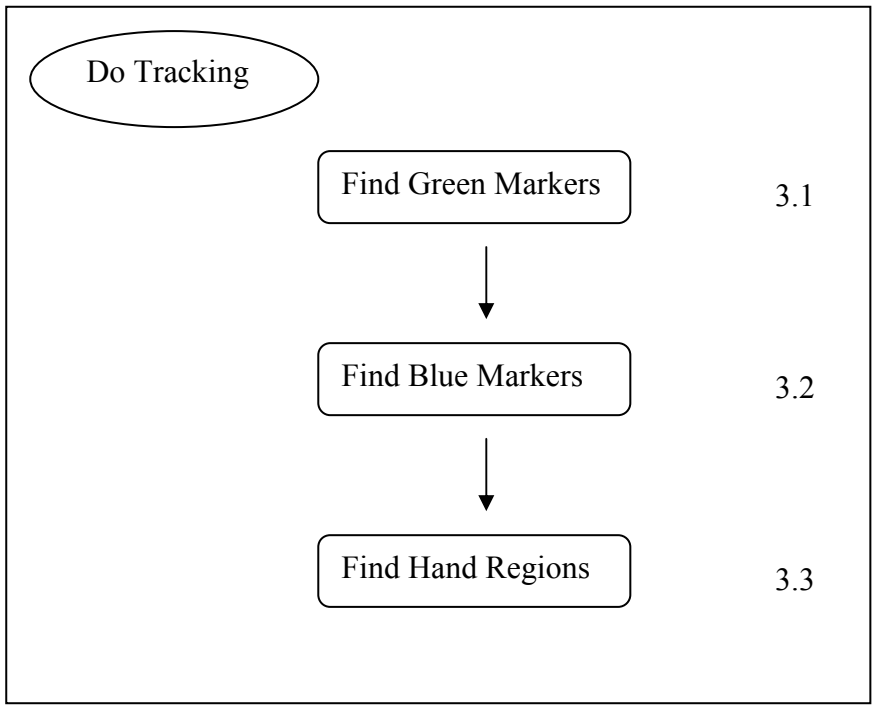


Figure 7.8: Steps for markers and hand regions during the tracking

As said before, all necessary steps for the tracking are being done in this step (Figure 7.8). For segmentation of fingers and finger-parts, colour markers are used. First of all, these markers are to be found separately. The process starts with the green markers and goes on with the blue ones. Finally, hand regions are processed. Here, green markers are used to identify and index the fingers. Hands and fingers are highly deformable objects. So, this task is highly difficult. Green markers are placed at the palm as close as possible to the fingers. Indexing those green markers means indexing fingers. This step gives only finger information. But a finger has two or more deformable parts. To separate those parts from each other and to index them in a finger, blue markers are used. Those markers are placed at the joints. After finding those markers and their positions, the related hand regions can be found. There is an important constraint about these overall steps.

In order to get a successful result (finding all points exactly with their information), the hand must be seen as clear as seen as clear as possible by the cameras. The markers (particularly blue markers) and hand regions should not be overlapped. If that kind of occlusion occurs, wrong results can be obtained.

Step 3.1) Find Green Markers

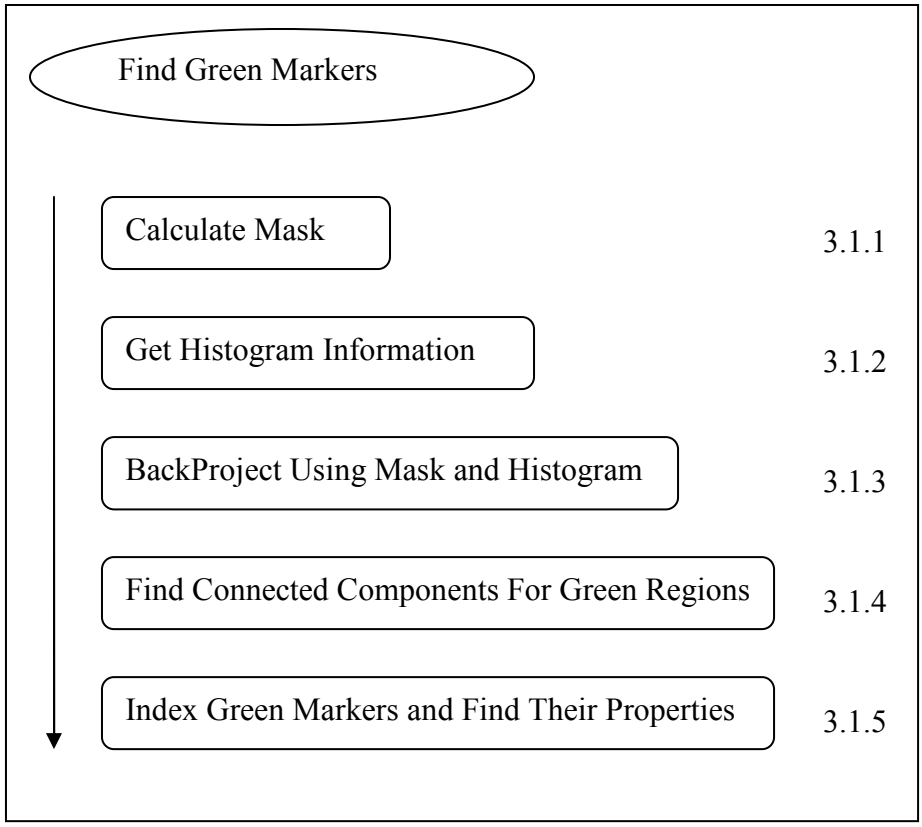


Figure 7.9: Steps for green markers during the tracking

Step 3.1.1) Calculate Mask: Both cameras get frames in RGB mode. By 'Colour Convertion' those frames are converted to the HSV mode. After that process, 'Mask' image can be made by filtering some values of the HSV image. These values are 'saturation' and 'value'. In the experiments, they are used as $S_{\min} = 30$, $V_{\min} = 10$ and $V_{\max} = 255$.

Step 3.1.2) Get Histogram Information: In order to track an object, first of all, it must be separated from the whole image (a kind of background subtraction). For this purpose, collection of some information about the object to be tracked is required. This information describes what to be searched in the input image. In this step, there are 2 possibilities. One of them is that, the required information (mostly stored in binary file) is available at the system and there is no need to update this information. The other possibility is to update this information during processing. Filtering is performed based on this information which is colour in our case. The colour values may be defined previously or can be defined during the processing. This colour information can be assumed to be the colour ranges bounded with upper and lower limits. Since the frame is in HSV, the information can be in the form of $H_{\min} < H < H_{\max}$. So in real time, in order to provide the required information about the object to track (shown in Figure 7.10-(2)), the user selects true colour information from the screen with a mouse as shown in Figure 7.10-(1)

(1) (2)

Figure 7.10: According to the selection (1) shown with blue region, necessary information about the green marker is taken (2).

Step 3.1.3) BackProject Using Mask …: The process of separation of tracked object from background takes place in this step. By using 'Histogram' information with the 'Mask' information, tracked object is separated in HSV colour space and the result is shown in gray mode (Figure 7.11-(1)).

(1) (2)

Figure 7.11: At the end of the filtering Figure 7.10-(1), the gray result image is found (1). By accepting (1) as input, the connected components are found in (2).

Step 3.1.4) Find Connected Components …: After filtering, the desired object(s) are extracted from the input image where the result is given as binary image. By accepting this binary image as the new input image, a second region-separation process is needed. As shown in Figure 7.10-(1), the green points are the desired objects and at the end of the filtering, only they are available (Figure 7.11-(1)). There are six well separated regions. To segment the separated regions, connectivity is used. With connectivity, while two regions are being processed, if there is a link in any shape between them then these regions are assumed to be together and accepted as the same region. If there is no link between, then they are decided to be different regions from each other. There is an important assumption in here which states that the regions of interest are well separated with markers. Thin markers, capturing frames at a very long distance from the cameras, bad lighting conditions and other reasons may cause a decrease in the performance of this process. At the end of this process, the filtered image (Figure 7.11-(1)) is separated individually and the necessary parameters of those parts are found as shown in Figure 7.11-(2), which are the center points (x,y) of those regions at pixel coordinates and the individual areas of those regions and can be thought as details.

Step 3.1.5) Index Green Markers ...: After finding connected components for green markers, they must be ordered for the further steps of tracking. First of all, finding reference green marker which is placed in the palm is required. To find that reference easily, it is adjusted to be the biggest in terms of the area. After finding that one, the next step is to find some relationships between this reference point and the other green markers. This is done by finding line equations between 2 points. One of the points is always that reference point (the biggest one) and the other one is selected among the others. This goes on until all the line equations for pairs of points are found and ordered. Ordering is done by comparing angle values of these lines with the 'x' axes. As the final step, these angles are reordered in one direction. These final ordered points are saved to an array. Figure 7.12 shows the result of this step. All green markers are processed and found. Also they are indexed as shown in Figure 7.12 respectively. At the end of this step, a great amount of information is collected. This information covers the coordinate knowledge, area information, predefined order (like sorting) and probably finger number that belong to it. Ordering is starting from the thumb as seen in Figure 7.12(2). In order to find the green point related to the thumb, that green marker is placed to be near to the reference green point. Hence by comparing the distances of green points to the reference one, the minimum distance gives us the thumb-green point. Later on, ordering (sorting) is done with respect to the thumb.

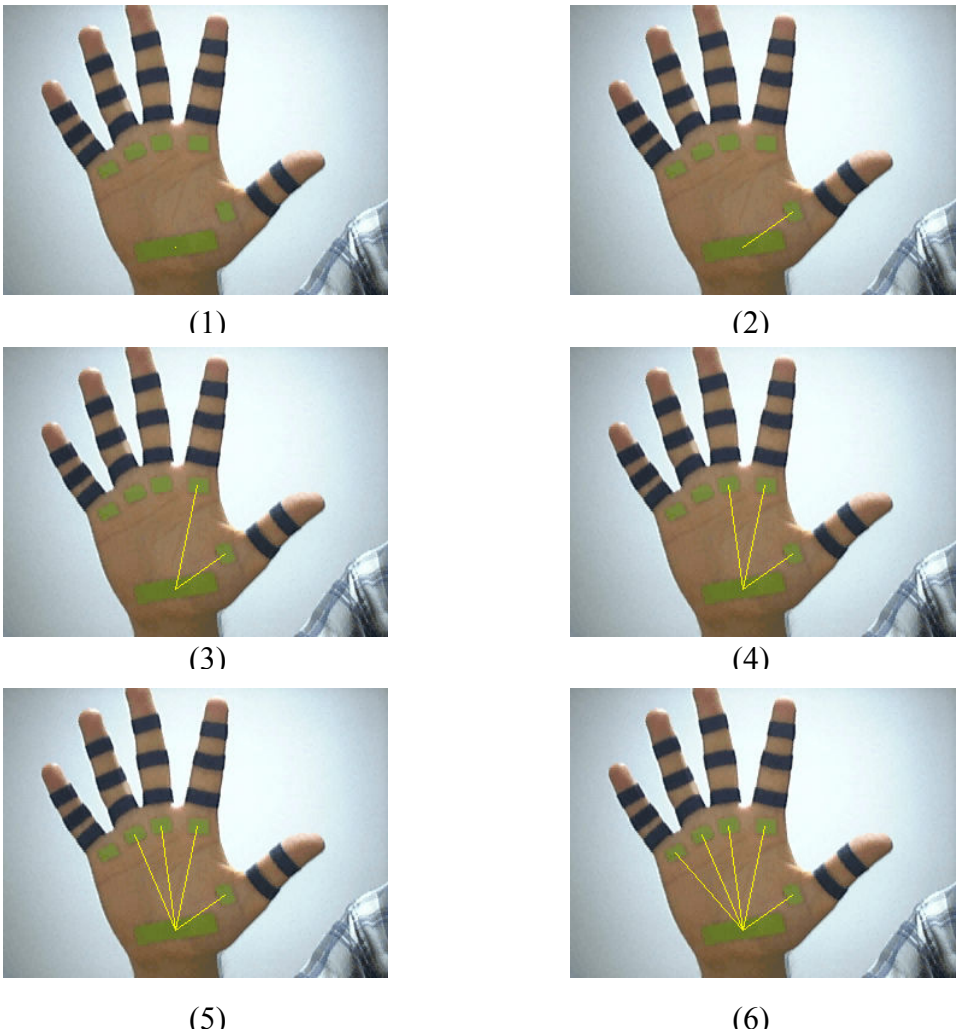


Figure 7.12: Green markers are found and labeled in order

Step 3.2) Find Blue Markers

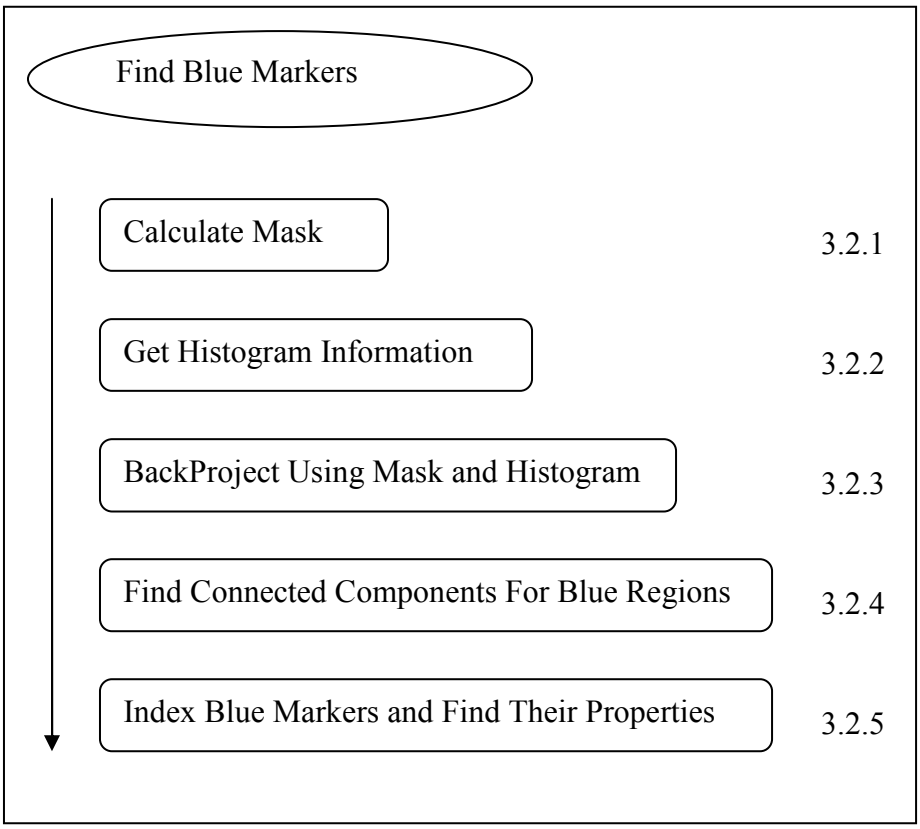


Figure 7.13: Steps for blue markers during the tracking

Step 3.2.1) Calculate Mask: Details are almost the same as in Step 3.1.1

Step 3.2.2) Get Histogram Information: Details are almost the same as in Step 3.1.2. Figure 7.14-(1) and (2) shows a region selected on the blue marker and the resulting information of the colour respectively.

(1)

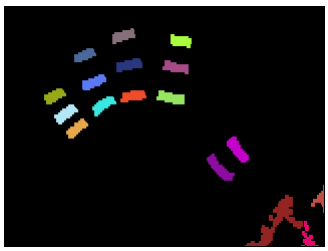
(2)

Figure 7.14: According to the selection (1), necessary information about the blue marker is taken (2).

Step 3.2.3) BackProject Using Mask …: Details are almost the same as in Step 3.1.3. For images see Figure 7.15-(1)

Step 3.2.4) Find Connected Components …: Details are almost the same as in Step 3.1.4

(1)



(2)

Figure 7.15: At the end of the filtering Figure 7.14-(1), the gray result image is found (1). By accepting (1) as input, the connected components are found in (2).

Step 3.2.5) Index Blue Markers ...: The same types of explanations are also valid for this step. Additionally, after finding all connected components for blue markers, firstly, all line equations are found according to the green reference point with other blue marker points (Figure 7.16). Let's call these equations as 'EQU 1'. With these line equations, in fact, some other useful data (as detail) becomes available. The slopes, lengths and the corresponding angle values for the lines are obtained. Hence, all lines (and also all blue points) can be sorted by using these angles. After that, the next step is to assign every blue marker point to its correct finger and to find its correct position in the finger. For that reason, a second find-line equations process is required (Figure 7.17). Again, the same approach is valid with only one difference. Now, every other green marker is taken as reference point and according to that new point, all line equations of blue points are found. This group of lines can be called as 'EQU 2'. So, we have all blue points (in terms of their line equations with respect to main reference green point and individual reference green points). The aim is to get the green marker points and blue marker points probably belong to the same finger together. So, these two kinds of equations are used to decide which point belongs to which finger. The next step is to match those equations to each other (First ones with the second ones). For this reason, the length and the slope of a line can be used. There is a common axis (x-axis) and all the equations have a slope with respect to this axis. By sorting these angles (from slopes), points can be grouped and assigned to their fingers. In the experiments, to group these green and blue marker points as if they belong to the same finger, it is adequate that the difference between two angle values to be equal or less than 10 degrees. By looking at these differences, the blue points with the appropriate green point are found. With this process, all related points (blue and green) are collected together. For a finger, there are only one green point and at least two blue points. After finding all points belonging to a finger, the points must be ordered according to the palm or more accurately, to the reference point(s). While finding the line equations, the length information also becomes available. According to this length information, a sorting process is adequate to order those points in a finger. All related lines are found in memory with a lot. For simplicity, in Figure 7.16 the arranged lines are drawn. In Figure 7.17, all of them are shown in their final ordered forms.

Figure 7.16: Blue markers are found and labeled in order (according to the reference green point)

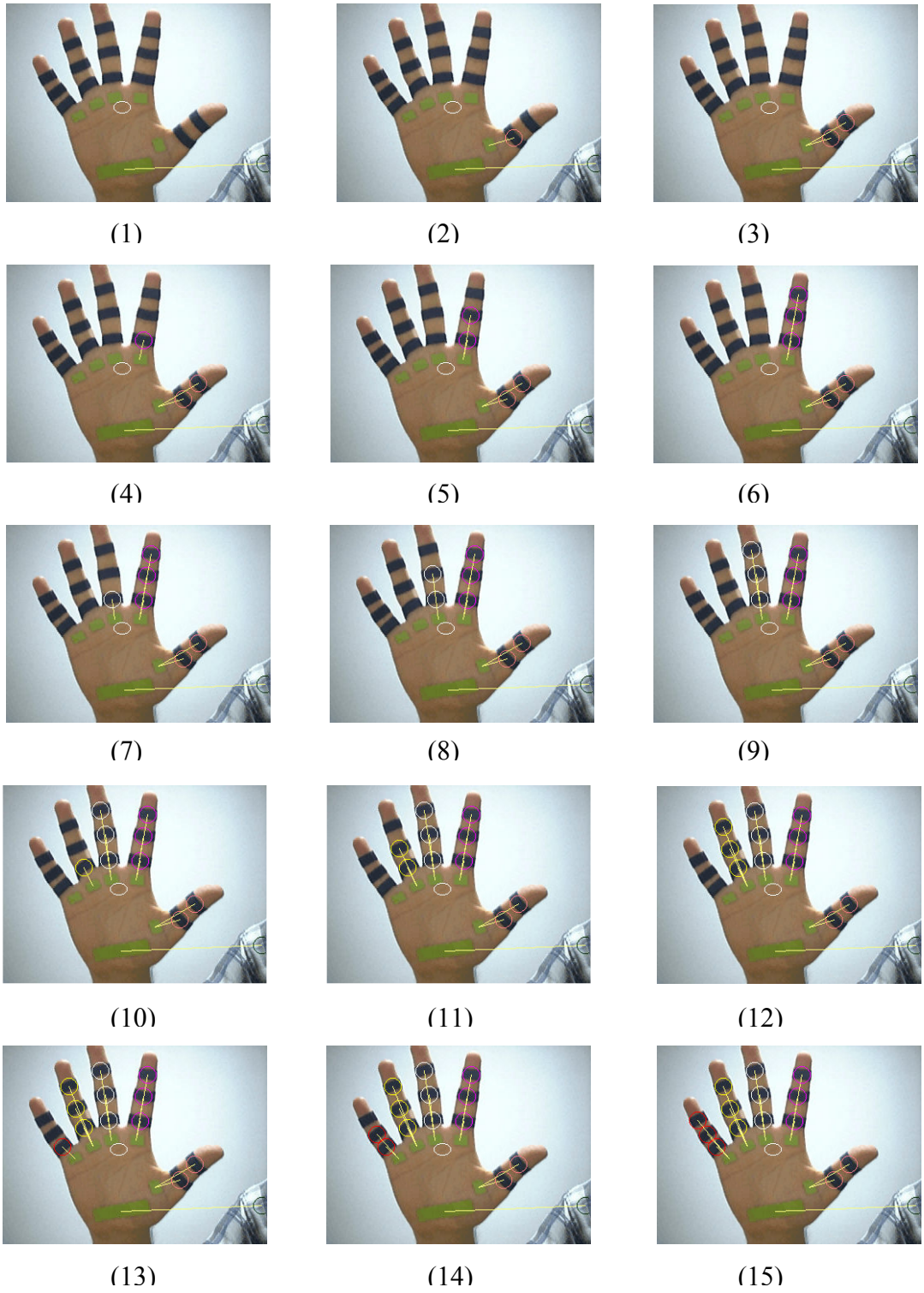


Figure 7.17: Blue markers are found and labeled in order (according to their individual reference green point).

Step 3.3) Find Hand Regions

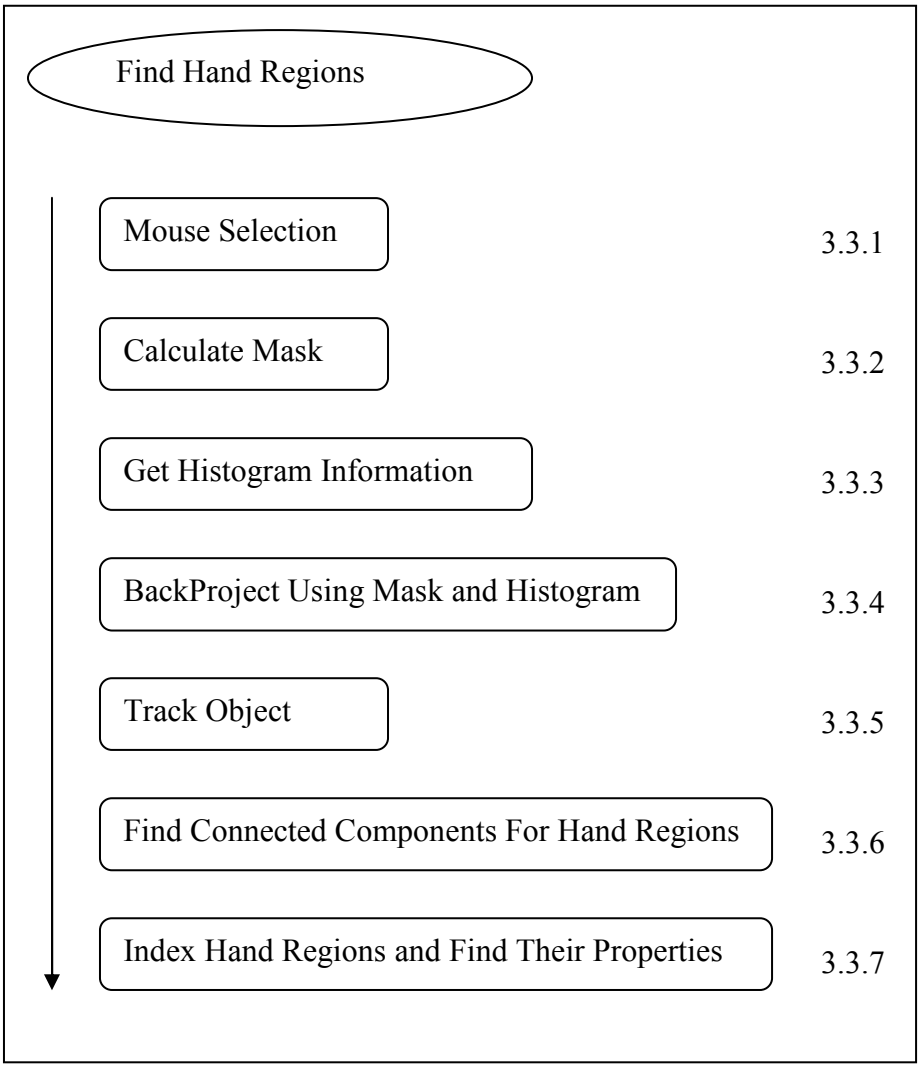


Figure 7.18: Steps for hand regions during the tracking

Step 3.3.1) Mouse Selection: In this part, user can select the object that he/she wants to track. By mouse, the user can specify the coordinates of the selection area. According to this selection, the data needed to be used in the 'Get Histogram Information' step is collected. Also when the selection is in progress, it triggers to update all histogram information for markers and hand regions as well.

Step 3.3.2) Calculate Mask: Details are almost the same as in Step 3.1.1

Step 3.3.3) Get Histogram Information: Details are almost the same as in Step 3.1.2. Figure 7.19-(1) and (2) shows an update process for hand region and the result information of that process respectively.

(1) (2)

Figure 7.19: According to the selection (1), necessary information about the hand region is taken (2).

Step 3.3.4) BackProject Using Mask …: Details are almost the same as in Step 3.1.3 and for images see Figure 7.20-(1)

(1) (2)

Figure 7.20: At the end of the filtering Figure 7.19-(1), the gray result image is found (1). By accepting (1) as input, the connected components are found in (2).

Step 3.3.5) Track Object: Object tracking is achieved by 'Camshift Algorithm'. If the user does 'mouse selection', track window's coordinates are updated with the selection ones. Tracking goes on with this new coordinates. If some problem occurs and track window is lost then the whole image is set as track window and according to the histogram values, the track object in the picture is tried to be found. If 'mouse selection' is not done then the related data values are read from files and the whole image is set as search window. Then, process goes on like this.

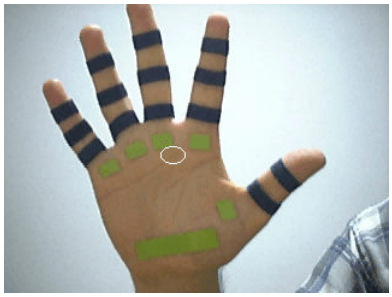


Figure 7.21: At the end of Camshift, this raw image is found. This image with its track information is inputted to the system in the internal processings.

Step 3.3.6) Find Connected Components …: Details are almost the same as in Step 3.1.4

Step 3.3.7) Index Hand Regions ...: The same explanations are also valid for this step as said before. It resembles to 'Index Blue Markers and Find Their Properties'. Because at every finger, there are blue marker points as much as (separated) hand regions. So, everything told for blue points are valid for this part as well. There is only one difference. By means of hand regions, the 'Find Connected Component' process finds the whole hand region as a component. This is true. But it is not useful. So this component must be deleted. By searching the component that has the biggest area and coordinate information is near to the track window center (white circle region) is deleted. After this deletion, every hand region can be thought as a corresponding point of related blue points. In Figure 7.22 and 7.23, the hand regions process can be seen.

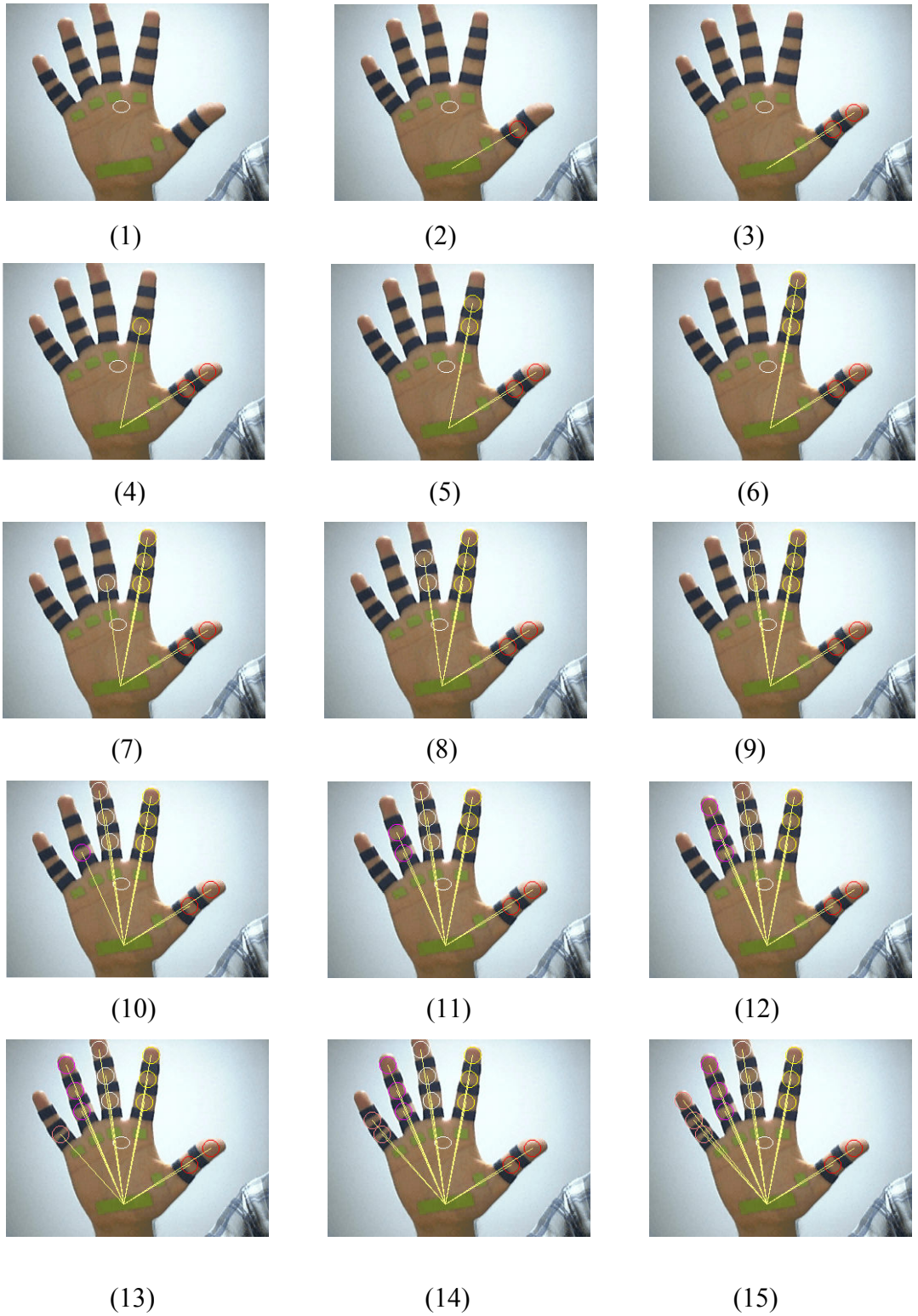


Figure 7.22: Hand regions are found and labeled in order (according to the reference green point)

Figure 7.23: Hand regions are found and labeled in order (according to their individual reference green point).

Step 4) Check Current Frame with Previous Frame

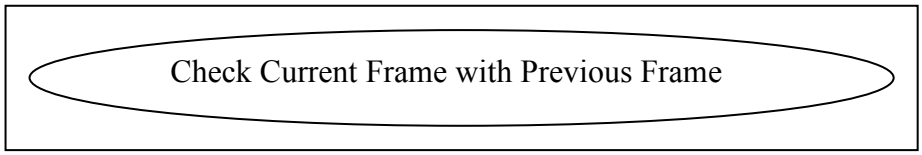


Figure 7.24: Comparison and correction of current frame with previous frame in terms of points.

Step 4) Check Current Frame…: As said before, if everything goes all right, all necessary points of the hand and markers will be available. But unfortunately, there may be so many miss-placed or lost points. To recover these points, it is necessary to check the current frame with the previous successful frames. This check is finger-based. For example, for the index finger, there are 3 finger segments and 3 blue markers which are used to separate these segments and a green marker at the bottom of this finger on the palm. During the process, if these points are found successfully then they are recorded. If an error with a point in a finger is occurred, it is searched and recovered from previous recorded data. For example, for a randomly chosen point, if n^{th} frame is a successful frame for this point but in the $(n+1)^{th}$ frame that point could not be found then its n^{th} information will be used as the $(n+1)^{th}$ information. Until the first following frame which is error-free, this will go on like this. But whenever an error-free frame for that point is caught, then for that point the recorded data is updated and will be used later. By doing this, both point(s)-based recovery and finger(s)-based recovery is possible. In Figure 7.25, an example can be seen.

(1) (2)

Figure 7.25: All the points related with the last finger are lost in (1). By using previous successful frames for each point in that finger, they are recovered (2).

Step 5) Find 3D Points From Stereo 2D Points. Calculate Required Angles and Draw Hand in 3D

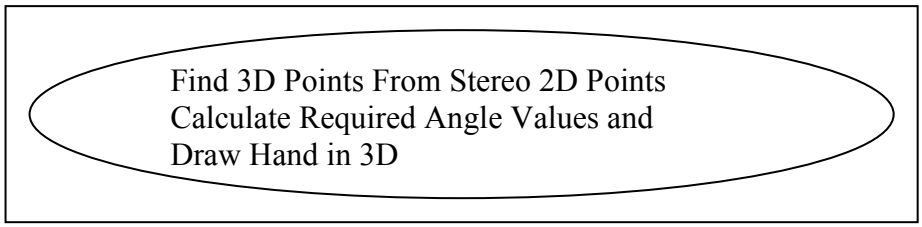


Figure 7.26: Reconstruction of 3D Point from stereo 2D Points and 3D Hand Drawing

Step 5) Find 3D Points ...: In this part, in order to see the tracked hand in 3D, stereo images must be used. To get the depth information, a second camera is required. If those cameras are called 'Left Camera' and 'Right Camera', firstly Left Camera image is processed and the required points and related data are obtained. And then, Right Camera image is processed in the same way as well. Using 2D information obtained from each camera separately, all found points are matched between stereo pairs and using camera projection matrices 3D information for each point is obtained (Özuysal, 2004).

Calculate Required Angle ...: In this part, the hand which is being tracked is visualized in 3D. For this purpose, a function is used which is written by Erdem Akagündüz in Java. For drawing (for example index finger), some angle values are required. For the index finger, we have a green marker point, 3 blue marker points and 3 hand region points. Also, since the fingers have some amount of degrees of freedom (DOF), their positions are given as angle values. After finding all related points in 3D from the corresponding 2D points, the required angle values are calculated and the resulting 3D points and angles are transferred to JAVA for visualizing the position and the movement of the whole hand in 3D.

In order to update the pose of the 3D Java hand model, total 22 angles are defined. Some of these angles describe the orientation of the hand and some of them describe the relative position of the finger segments to each other.

If A and B are two different points in 3D, then these points can be represented in terms of vectors and their notations are shown such that vector $\overrightarrow{\mathbf{A}}$ and $\overrightarrow{\mathbf{B}}$

Additionally, vector $\overrightarrow{\mathbf{AB}}$ is defined as

$$\overrightarrow{\mathbf{AB}} = \vec{\mathbf{B}} - \vec{\mathbf{A}} \tag{7.1}$$

If C and D are another two different points in 3D, then vector $\overrightarrow{\mathbf{CD}}$ is defined

$$\overrightarrow{\mathbf{CD}} = \vec{\mathbf{D}} - \vec{\mathbf{C}} \tag{7.2}$$

Then, dot product of these two vectors is defined as the following.

$$\overrightarrow{\mathbf{AB}}.\overrightarrow{\mathbf{CD}} = \left|\overrightarrow{\mathbf{AB}}\right|\left|\overrightarrow{\mathbf{CD}}\right|\cos\theta \tag{7.3}$$

And angle between two vectors can be found with the following equation.

$$\theta = \arccos\left(\frac{\overrightarrow{\mathbf{AB}}.\overrightarrow{\mathbf{CD}}}{\left|\overrightarrow{\mathbf{AB}}\right|\left|\overrightarrow{\mathbf{CD}}\right|}\right) \tag{7.4}$$

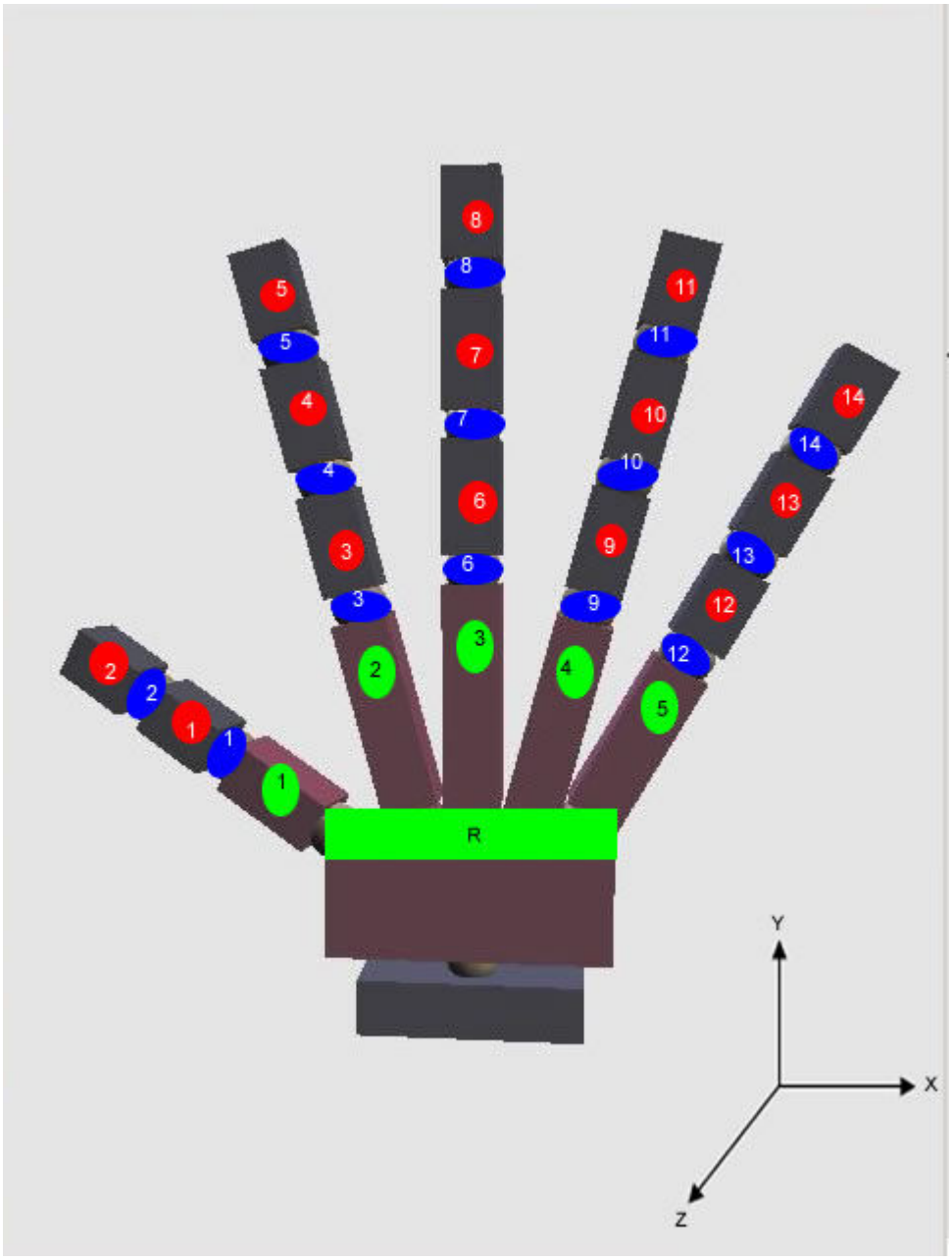


Figure 7.27: All fingers and finger segments are indexed. The posture of the Java 3D hand model is updated by angle calculations on Java 3D hand model

(R means reference green marker)

(G means green marker) (B means blue marker) (H means hand region)

1. First of all, a normal vector ($\vec{\mathbf{N}}$) of the palm is found. It is in (+z) direction.

$$\vec{\mathbf{N}} = \overrightarrow{\mathbf{G_1R}} \mathrm{x} \overrightarrow{\mathbf{RG}_5} \tag{7.5}$$

By using this normal vector, the direction of the palm (and hand as well) in 3D can be found. In fact, in our system, we use this obtained normal vector to calculate the amount of deviation from initial z direction. As seen in Figure 7.27, the hand model is initialized and its direction is assumed to be in z direction with zero deviation. For every frame, this vector is calculated and during the process, it is used to calculate the necessary angles to update the posture of the hand model.

2. If the obtained normal vector is defined as (n_x, n_y, n_z), then

- Rotation on the y axis is

$$\arctan\left(\frac{n_x}{n_z}\right) \tag{7.6}$$

- Rotation on the x axis is

$$\arccos\left(sqrt\left(\frac{{n_x}^2 + {n_z}^2}{{n_x}^2 + {n_y}^2 + {n_z}^2}\right)\right) \tag{7.7}$$

In fact, these calculated angles are used to compare the current tracked hand posture with the initial hand posture in terms of rotation. So, these angles are not used directly. Instead, their differences with the related angles of the initial hand posture are used. Hence, the model is rotated in related directions by amount of those differences.

3. For the thumb finger, total 4 angles are calculated. Since its DOF is 2, one of 4 angles is related with the 'Abduction and Adduction' and the other 3 angles are related with the 'Flexion and Extension'.

   - Abduction and Adduction

$$\overrightarrow{\mathbf{RG}_2}.\overrightarrow{\mathbf{RB}_1} \tag{7.8}$$

   - Flexion and Extension

$$\overrightarrow{\mathbf{RG}_1}.\vec{\mathbf{N}} \tag{7.9}$$

$$\overrightarrow{\mathbf{G}_1\mathbf{B}_1}.\overrightarrow{\mathbf{B}_1\mathbf{H}_1} \tag{7.10}$$

$$\overrightarrow{\mathbf{H}_1\mathbf{B}_2}.\overrightarrow{\mathbf{B}_2\mathbf{H}_2} \tag{7.11}$$

Initialy, equations 7.8, 7.9, 7.10 and 7.11 are calculated and saved in memory. These initial angles describe the initial posture of the thumb finger of our 3D hand model as seen in Figure 7.27. During the overall process, when these angles are calculated, by looking at the differences of new calculated angles with the corresponding initial angles, the necessary information for the thumb is sent to the Java hand model. Hence, the pose of the thumb can be updated.

Equation 7.9 describes the movement of the finger into the palm.

Equation 7.10 and Equation 7.11 describes the movements of the joints labelled B_1 and B_2 respectively.

4. For the index finger, total 4 angles are calculated and all explanations previously told for the thumb finger are also valid.

   - Abduction and Adduction

$$\overrightarrow{\mathbf{G}_2\mathbf{G}_3}.\overrightarrow{\mathbf{G}_2\mathbf{H}_3} \tag{7.12}$$

- Flexion and Extension

$$\overrightarrow{\mathbf{G}_2\mathbf{B}_3}.\overrightarrow{\mathbf{B}_3\mathbf{H}_3} \tag{7.13}$$

$$\overrightarrow{\mathbf{H}_3\mathbf{B}_4}.\overrightarrow{\mathbf{B}_4\mathbf{H}_4} \tag{7.14}$$

$$\overrightarrow{\mathbf{H}_4\mathbf{B}_5}.\overrightarrow{\mathbf{B}_5\mathbf{H}_5} \tag{7.15}$$

In initial pose of the 3D Java hand model (Figure 7.27), from equations 7.12, 7.13, 7.14 and 7.15, initial angles for the index finger are calculated and saved to be used later during the comparisons. During the overall process, when these angles are calculated, by looking at the differences of new calculated angles with the corresponding initial angles, the necessary information for the index finger is sent to the Java hand model.

5. The procedure for the middle finger is also similar to the previous fingers. The related equations are given below.

   - Abduction and Adduction

$$\overrightarrow{\mathbf{G}_3\mathbf{G}_4}.\overrightarrow{\mathbf{G}_3\mathbf{H}_6} \tag{7.16}$$

   - Flexion and Extension

$$\overrightarrow{\mathbf{G}_3\mathbf{B}_6}.\overrightarrow{\mathbf{B}_6\mathbf{H}_6} \tag{7.17}$$

$$\overrightarrow{\mathbf{H}_6\mathbf{B}_7}.\overrightarrow{\mathbf{B}_7\mathbf{H}_7} \tag{7.18}$$

$$\overrightarrow{\mathbf{H}_7\mathbf{B}_8}.\overrightarrow{\mathbf{B}_8\mathbf{H}_8} \tag{7.19}$$

6. The procedure for the ring finger is also similar to the previous fingers. The related equations are given below.

   - Abduction and Adduction

$$\overrightarrow{\mathbf{G}_4\mathbf{G}_5}.\overrightarrow{\mathbf{G}_4\mathbf{H}_9} \tag{7.20}$$

   - Flexion and Extension

$$\overrightarrow{\mathbf{G}_4\mathbf{B}_9}.\overrightarrow{\mathbf{B}_9\mathbf{H}_9} \tag{7.21}$$

$$\overrightarrow{\mathbf{H}_9\mathbf{B}_{10}}.\overrightarrow{\mathbf{B}_{10}\mathbf{H}_{10}} \tag{7.22}$$

$$\overrightarrow{\mathbf{H}_{10}\mathbf{B}_{11}}.\overrightarrow{\mathbf{B}_{11}\mathbf{H}_{11}} \tag{7.23}$$

7. The procedure for the little finger is also similar to the previous fingers. The related equations are given below.

   - Abduction and Adduction

$$\overrightarrow{\mathbf{G}_5\mathbf{G}_4}.\overrightarrow{\mathbf{G}_5\mathbf{H}_{12}} \tag{7.24}$$

   - Flexion and Extension

$$\overrightarrow{\mathbf{G}_5\mathbf{B}_{12}}.\overrightarrow{\mathbf{B}_{12}\mathbf{H}_{12}} \tag{7.25}$$

$$\overrightarrow{\mathbf{H}_{12}\mathbf{B}_{13}}.\overrightarrow{\mathbf{B}_{13}\mathbf{H}_{13}} \tag{7.26}$$

$$\overrightarrow{\mathbf{H}_{13}\mathbf{B}_{14}}.\overrightarrow{\mathbf{B}_{14}\mathbf{H}_{14}} \tag{7.27}$$

In order to provide a visualization of region, segment and marker detection, all points (green points, blue points and hand regions) are shown with different colours on a black screen in (Figure 7.28, Figure 7.29 and Figure 7.30)

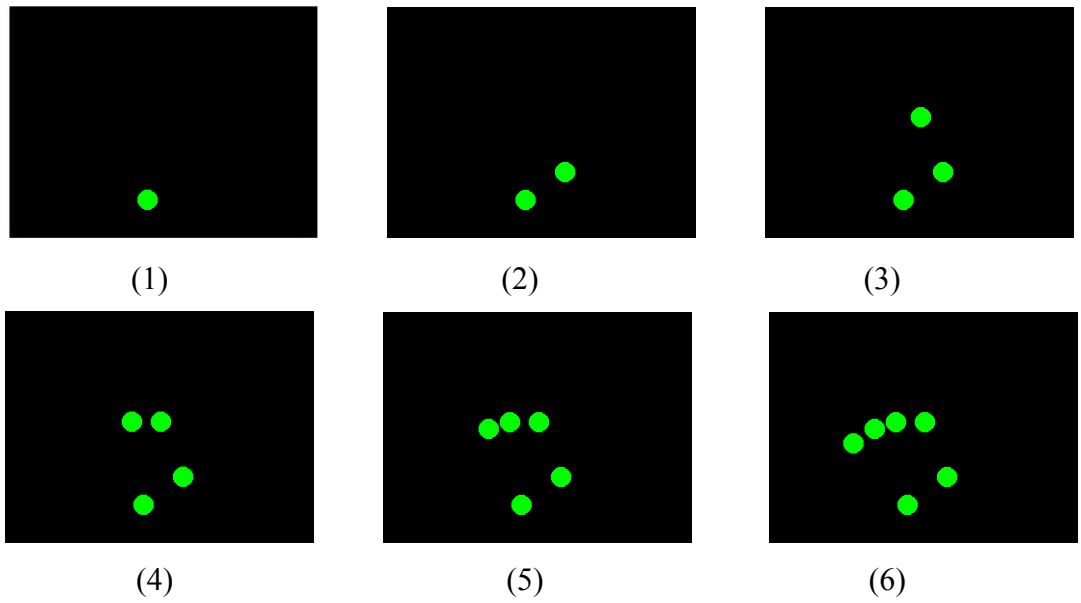


Figure 7.28: The green points in tracked image. These points can be thought as if they are palm points. They are ordered and known in detail.

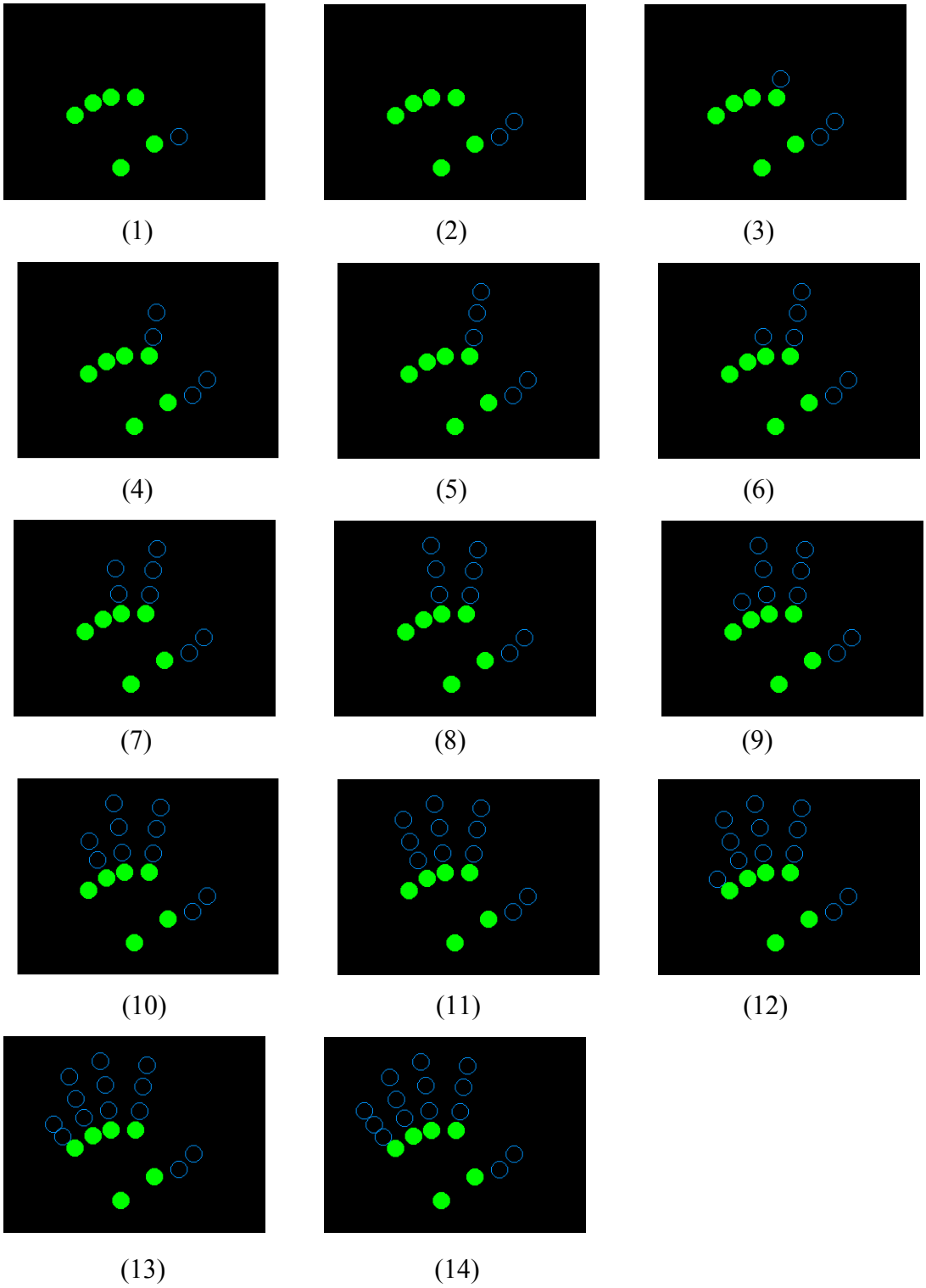


Figure 7.29: The blue and green points together in tracked image. The blue points can be thought as if they are our seperators in the fingers. They are ordered and known in detail.

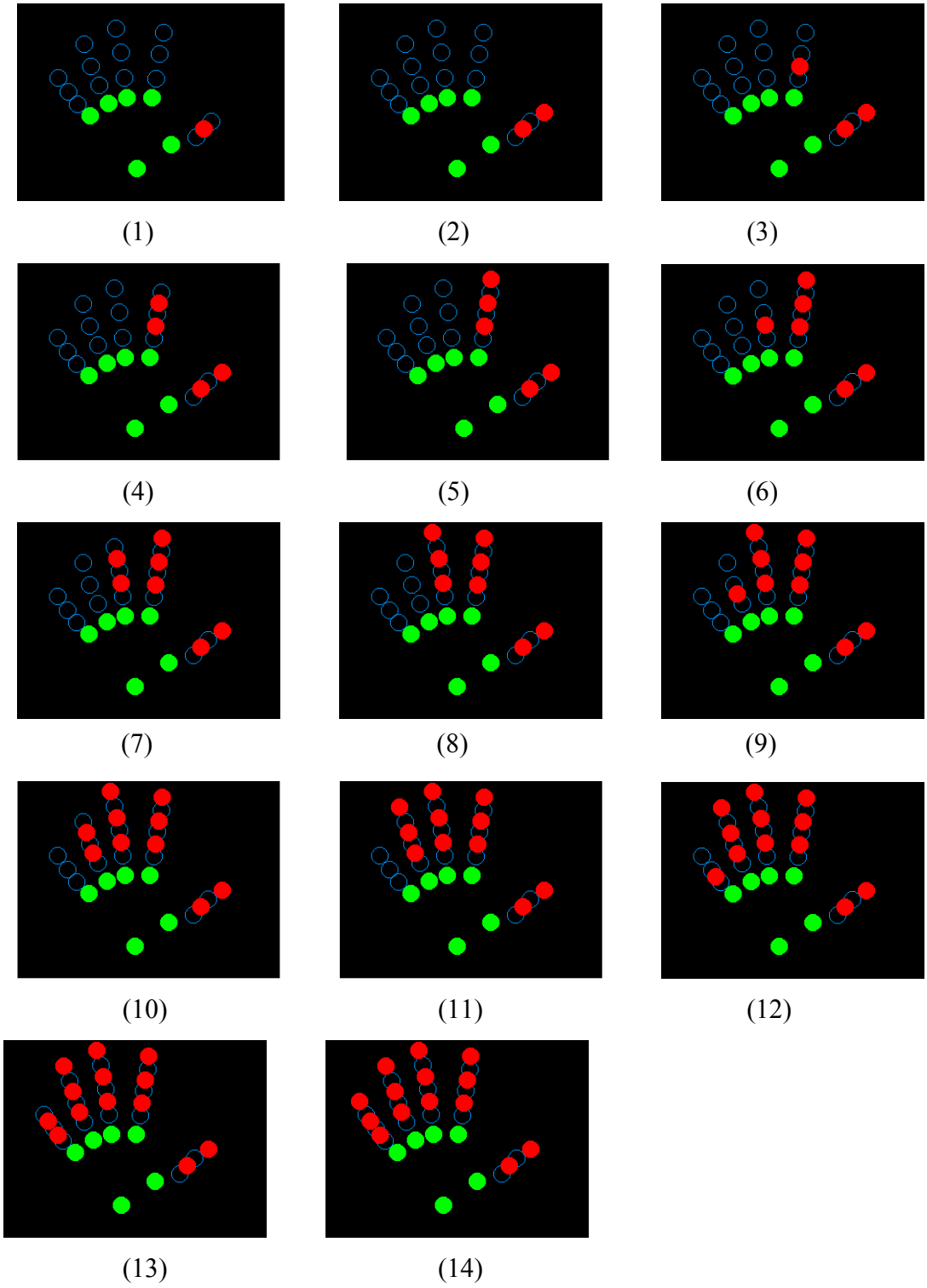


Figure 7.30: All points together in tracked image. The red points can be thought as if they are our hand regions separated with blue markers in the fingers. They are ordered and known in detail.

In the following (Figure 7.31), the input images of the hand and the resulted Java hand images are presented. The upper-left one belongs to the “Left image” and the lower left is the “Right image”. In the input images, background is eliminated (seen as black in the images). 3D information obtained from these stereo images is visualized on the 3D model given on the right of each figure. (See Appendix A)

(1) (2)

(3) (4)

(5) (6)

Figure 7.31: Hand tracking results in 3D Java hand model

CHAPTER 8

CONCLUSION AND FUTURE WORKS

In this thesis, we aimed to develop a hand tracker system. This system should work in real-time as well as off-line mode. It should not require any specific hardware (means a standard PC Desktop system should be enough).It should track finger movements as well as hand movements. It should be free on external devices like gloves, sensors etc. and tracking should be achieved in 3D.

In the experiments, the frame rate is found as ~8 frames per second. So with this result, our system can be said to be working on real-time.

System consists of the equipments summarized in the Table 7.1. Cameras seem to be special but when the system is worked with the cheap web-cams, the tracking process is achieved but the performance is worse than when special cameras are used since it is tightly correlated with the quality of image pictures.

Our hand tracking system is based on skin colour and coloured-markers. So any object which has the similar colour as the hand can be accepted as hand region. This is a handicap for the tracking system and may cause it to fail. The same handicaps are valid for the markers, also. Because, after finding hand in the picture, the segmentation of the finger regions is done using those coloured markers. Any additional objects which have similar colours with those markers drop the performance of correct segmentation or may it be wrong totally. So during the tracking of hand and fingers, there must be no object other than the object and its markers with colours similar to the flesh colours and marker colours. In other words, to get a better result for this segmentation, the background should be clear and empty as much as possible. Additionally, background colour should be far away from colours of the tracked object and its markers.

Under these conditions, our system can track hand and finger movements. When they are compared in terms of robustness, the hand tracking is much more robust than the fingers. Hand region is the overall of the all finger and non-finger areas. Besides, the fingers have much more degrees of freedom and more deformable than a hand.

Our system does not handle finger-finger occlusion. When that kind of a situation happens, the tracking of the occluded fingers fail.

In terms of lighting, there must be soft-lightened and stable environment. The lightning conditions must be as stable as possible. When the lightning conditions are changed, this affects the overall system and probably it drops the performance of the tracking or at the worst, it may cause the system to fail totally. When it fails, until the next frames, the system does nothing.

Our system is based on an image-based hand tracking method. Because, high detailed quantitative description of the hand pose and movements are not used. But stereo vision approach is used to achieve the tracking in 3D. First of all, the images taken from the stereo cameras are processed in 2D. The hand regions are segmented from the background by using their colour information. The fingers are indexed and segmented into individual parts by using the markers. Up to this point, all processes take place in 2D. In order to reconstruct the tracked object in 3D, its feature points (both defined previously and found at the end of the segmentation) are used. To recover 3D information of a point, its projection points information (the corresponding points on the left and right image) and camera projection matrices (derived from the stereo calibration of the cameras) are used. When all the corresponding feature points are back-projected to the 3D environment, the positions of the feature points in 3D are obtained. By using these 3D points, the 3D hand model's pose is updated.

This update process includes some angle calculations, vector calculations and orientation information. After all feature points are found and translated into the 3D coordinate system, palm orientation and finger positions are decided. For this purpose, first of all, the direction of the palm is found by using the information related to palm feature points (for detail explanation of this paragraph, see Chapter

7). Here, palm points are the green reference points. After that, the finger points, which are the blue markers and hand regions between those blue markers, are extracted and their positions with respect to each other (by means of angle) are calculated. At the same time, the finger locations according to the palm (in 3D) are found in terms of angles.

The results are obtained without doing any big and mainly important optimization. For image processing, OpenCV library and for 3D hand model, Java 3D virtual engine, which is based on OpenGL, is used.

In this study, hand and fingers are tracked in video sequences using markers placed on hand and fingers. The main purpose to achieve is to track hand using only camera and pure human hand without any markers. Because, human hands and fingers have high degree of freedom, in this initiative study in our Computer Vision and Intelligent Systems Research Laboratory, we used markers to simplify the problem. This breaks the naturalness and in the future, the requirements of these markers are to be eliminated. From just hand images, without using any devices or markers on the hands, the tracking should be done.

Another important point is the reduction of dependency on the lightning conditions. Although HSV colour space, which is more robust than RGB, is used, the dependency on the lightning is very high and the changes on the lightning (to be very shiny or opposite) drops the performance of the tracking.

One of the biggest problems met during this study is the indexing of the fingers and individual finger parts. Most of time, hand regions and markers are found separately and exactly. The most important situations which break that operation (finding and indexing) are the poor lightning conditions, the occlusion problem and the closeness of the points of interest to each others. Especially, for the fingers there is high degree of freedom and deformability and this results the occlusion problem frequently. Once, the segmentation of hand from the background and the individual finger parts within a finger is achieved, the next step is to find which point belongs to which finger and at which position it must be. Indexing is done with finding some linear equations and the relationship between them. Line equations are extracted between

the reference points, which are the green and blue markers, and the hand regions. According to a green marker, the other blue points and hand regions which are found to be at the same direction from that green point are tried to be grouped and assigned that corresponding finger. Although all the related points are found at the end of segmentation, during the indexing process some of them are lost. The method used for indexing seems to be fine and while the hand seems to be clear on the screen in terms of occlusion, that indexing process works correctly. However, the high degrees of freedom of fingers and their deformable structure make indexing process difficult to deal with them. Some orientation of the hand or some finger movements within the hand bring the loose of some points while indexing. In order to overcome those kinds of lost, the indexing process must be reviewed and developed.

In order to simplify the hand tracking problem, we restricted the movement of the hand so that it is close to frontal view in some degree of freedom. We used cylindrical stripes around the fingers as markers and we have taken the center of the marker at each image as the corresponding point. Although this assumption works well for the frontal view of hand when fingers are open, it causes problem when the hand is rotated and fingers are bended. In future study, special care should be paid on finding corresponding points on the fingers when considering more general orientation and posture of the hand.

The reconstruction is done with the camera matrices and the corresponding 2D information of the points. Both the left and the right images are processed separately. After finding all related points with their index information, the points with the same index number (in terms of finger number and position in a finger) are assumed to be the corresponding points. In traditional stereo vision, the corresponding points in image pairs are decided on the properties of the feature points and camera calibration data. But in our system, they are found separately and matched according to their index numbers. This is a reasonable assumption when indexing is correct. The performance of the reconstruction process is also related with the camera matrices. With the camera calibration process, the most suitable camera matrices parameters are generated. But always there are some errors within the tolerance. So, the correctness of the reconstruction is concerned with the correctness of the calibration

parameters and correctness of matching procedure. The camera calibration process and its results should be improved for a better performance.

Finally, we have used a simple 3D hand model made of cylinders and spherical joints. For a better representation of the hand in 3D, a more detailed model close to appearances of the hand will improve the visual quality.

REFERENCES

[1] (Ahmad, 1995)
S. Ahmad. A usable real-time 3d hand tracker. In Proceedings 28th Asilomar Conference on Signals, Systems and Computers, pages 1257-261. IEEE Computer Society Press, 1995.

[2] (Azarbayejani et al., 1996)
A. Azarbayejani, C. Wren, and A. Pentland , "Real-Time 3D Tracking of the Human Body," Proc. IMAGE'COM 96, Bordeaux, France, 1996

[3] (Azarbayejani and Pentland, 1996)
A. Azarbayejani and A. Pentland. Real-time self-calibrating stereo person tracking using 3d shape estimation from blob features. In 13th International Conference on Pattern Recognition, pages 627- 632, volume C, Vienna, Austria, August 1996. IEEE Computer Society Press

[4] (Athitsos and Sclaroff, 2002)
V. Athitsos and S. Sclaroff. An appearence-based framework for 3d hand shape classification and camera viewpoint estimation. In 5th Conference on Face and Gesture Recognition. IEEE, IEEE Computer Society Press, May 2002.

[5] (Bar-Shalom et al., 2001)
Yaakov Bar-Shalom, Xiao Rong Li, and Thiagalingam Kirubarajan. Estimation with Applications to Tracking and Navigation: Theory, Algorithms, and Software. John Wiley & Sons, Inc., 2001.

[6] (Baumberg and Hogg, 1995)
A. Baumberg and D. Hogg. An adaptive eigenshape model. In D. Pycock, editor, 6th British Machine Vision Conference, volume 1, pages 87-96, Birmingham, July 1995. BMVA

[7] (Berry et al., 1998)
G. A. Berry, V. Pavlovic, and T. S. Huang. Battleview: A multimodal hci research application. In Workshop on Perceptual User Interfaces, San Francisco, November 1998.

[8] (Blake and Isard, 1998)
A. Blake and M. Isard. Active Contours. Springer Verlag, April 1998

[9] (Bowden, 1999)
R. Bowden. Learning Non-linear Models of Shape Motion. PhD thesis, Department of Sustems Engineering, Brunel University, October 1999

[10] (Bowden and Sarhadi, 2002)
R. Bowden and M. Sarhadi. A non-linear model of shape and motion for tracking finger spelt american sign language. Image and Vision Computing, Elsevier Science, 20(9-10):597-607, August 2002.

[11] (Bradski, 1998)
Gary R. Bradski . Computer Vision Face Tracking For Use in a Perceptual User Interface. Microcomputer Research Lab, Santa Clara, CA, Intel Corporation

[12] (Chai and Ngan, 1998)
D. Chai and K. n. Ngan. Locating facial region of a head- and-shouders colour image. In 3rd International Conference on Automatic Face and Gesture Recognition, pages 124-129, Nara, Japan, 1998. IEEE, Computer Society Press

[13] (Chua et al., 2002)
C-S Chua, H. Guan, and Y-K Ho. Model-based 3d hand posture estimation from a single 2d image. Image and Vision Computing - Elsevier Science B.V., 20:191-202, 2002.

[14] (Cipolla and Hollinghurst, 1996)
R. Cipolla and N. J. Hollinghurst. Human-robot interface by pointing with Uncalibrated stereo vision. Image and Vision Computing, 14(3):171–178, April 1996.

[15] (Clergue et al., 1995)
E. Clergue, M. Goldberg, N. Madrane, and B. Merialdo, "Automatic Face and Gestural Recognition for Video Indexing," Proc. Intel Workshop on Automatic Face and Gesture Recognition, Zurich, Switzerland, pp. 110-115, June 1995.

[16] (Costa and Cesar-Jr, 2001)
L. da Fontoura Costa and R. M. Cesar-Jr. Shape Analysis and Classification – Theory and Practice. Image Processing. CRC Press, 2001

[17] (Delamarre and Faugeras, 1998)
Q. Delamarre and O. D. Faugeras. Finding pose of hand in video images: a stereobased approach. In Proc. 3rd Int. Con. on Automatic Face and Gesture Recogntion, pages 585–590, Nara, Japan, April 1998.

[18] (Downton and Drouet, 1991)
A. C. Downton and H. Drouet, "Image Analysis for Model-Based Sign Language Coding," Progress in Image Analysis and Processing II: Proc. Sixth Intel Conf. Image Analysis and Processing, pp. 637- 644, 1991

[19] (Duda and Hart, 1973)
R. O. Duda and P. E. Hart. Pattern Classification and Scene Analysis. Wiley-Interscience, USA, 1st edition, 1973

[20] (Duda et al., 2000)
R. O. Duda, P. E. Hart, and D. G. Stork. Pattern Classification. Wiley-Interscience, New York, USA, second edition, 2000

[21] (Efros and Leung, 1999)
A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In International Conference on Computer Vision, Corfu, Greece, September 1999. IEEE, IEEE

[22] (Etoh et al., 1991)
M. Etoh, A. Tomono, and F. Kishino, "Stereo-Based Description by Generalized Cylinder Complexes From Occluding Contours," Systems and Computers in Japan, vol. 22, no. 12, pp. 79- 89, 1991.

[23] (Faux and Pratt, 1979)
Faux and Pratt. Computational geometry for design and manufacture. Ellis Horwood, 1979.

[24] (Feris et al., 2000)
R. S. Feris, T. E. Campos, and R. M. Cesar-Jr. Detection and tracking of facial features in video sequences. In Lecture Notes in Artificial Intelligence, volume 1973, pages 129-137, Acapulco, Mexico, April 2000. Springer-Verlag Press

[25] (Flores et al., 2000)
F. C. Flores, R. Hirata Jr.,J. Barrera, R. A. Lotufo, and F. Meyer. Morphological Operators for Segmentation of Color Sequences. In Proceedings of SIBGRAPI'2000, pages 300-307, Gramado, Brazil, October 2000. IEEE Computer Society Press

[26] (Fukunaga, 1990)
K. Fukunaga, "Introduction to Statistical Pattern Recognition," Academic Press, Boston, 1990

[27] (Gavrila and Davis, 1995)
D.M. Gavrila and L.S. Davis, “Towards 3D Model-Based Tracking and Recognition of Human Movement: A Multi-View Approach,” Proc. Intel Workshop on Automatic Face and Gesture Recognition, Zurich,Switzerland, pp. 272-277, June 1995.

[28] (Gavrila and Davis, 1996)
D. M. Gavrila and L. S. Davis. 3-d model-based tracking of humans in action: a multi-view approach. In Computer Vision and Pattern Recognition, San Francisco, CA, USA, 1996. IEEE

[29] (Goncalves et al., 1995)
L. Goncalves, E. Bernardo, E. Ursella, and P. Perona. Monocular tracking of the human arm in 3d. In 5th International Conference on Computer Vision, pages 764-770, Los Alamitos, CA, USA, 1995. IEEE Computer Society Press

[30] (Gonzalez and Woods, 2000)
R. C. Gonzalez and R. E. Woods. Digital Image Processing. Prentice Hall, Pearson Education International, New Jersey, 2nd –international edition, 2000

[31] (Hamada et al., 2002)
Y. Hamada, N. Shimada, and Y. Shirai. Hand shape estimation using sequence of multi-ocular images based on transition network. In 15th International Conference on Vision Interface, pages 362-368, Calgary, Alberta, Canada, May 2002. CIPPRS and IAPR

[32] (Heap and Hogg, 1996)
T. Heap and D. Hogg. Towards 3d hand tracking using a deformable model. In 2th Conference on Face and Gesture Recognition, Killington, Vermont, USA, October 1996. IEEE, IEEE Computer Society Press

[33] (Hill et al., 1995)
A. Hill, T. F. Cootes, and C. J. Taylor. Active shape models and the shape approximation problem. In D. Pycock, editor, 6th British Machine Vision Conference, volume 1, pages 157-166, Birmingham, July 1995. BMVA

[34] (Isard and Blake, 1998)
M. Isard and A. Blake. Condensation: Conditional density propagation for visual tracking. International Journal of Computer Vision, 29(1):5-28, 1998.

[35] (Isard and MacCormick, 2000)
M. Isard and J. MacCormick. Partitioned sampling, articulated objects, and interface-quality hand tracking. In Proc. 6th European Conf. on Computer Vision, volume 2, pages 3–19, Dublin, Ireland, June 2000.

[36] (Jack, 2001)
K. Jack. Video Demystified. LLH Technology Publishing, third edition, 2001

[37] (Jain et al., 2000)
A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: A review. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(1):4-37, January 2000

[38] (Ju et al., 1996)
S. Ju, M. Black, and Y. Yacoob. Cardboard people: a parameterized model of articulated motion. In Conference on Automatic Face And Gesture Recognition, pages 38-44, Killington, 1996. IEEE

[39] (Kakadiaris et al., 1994)
I.A. Kakadiaris, D. Metaxas, and R. Bajcsy, “Active Part Decomposition, Shape and Motion Estimation of Articulated Objects: A Physics-Based Approach,” Proc. IEEE C.S. Conf. Computer Vision and Pattern Recognition, pp. 980-984, 1994

[40] (Koch, 1993)
R. Koch, “Dynamic 3D Scene Analysis Through Synthetic Feedback Control,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 15, no. 6, pp. 556-568, 1993

[41] Lyons (2002)
D. E. Lyons. A qualitative approach to computer sign language recognition. Master's thesis, Department of Engineering Science, University of Oxford, Oxford, UK, August 2002. Supervised by M. Brady and I. Reid.

[42] (MacCormick and Isard, 2000)
J. MacCormick and M. Isard. Partitioned sampling, articulated objects, and interface-quality hand tracking. In D. Vernon, editor, 6th European Conference on Computer Vision, number 1843 in Lecture Notes in Computer Science, pages 3-19, Dublin, Ireland, June/July 2000. Springer Part II.

[43] (Magnenat-Thalman, 1990)
N. Magnenat-Thalmann and D. Thalman, Computer Animation: Theory and Practice. New York: Springer-Verlag, 2nd rev. ed., 1990.

[43] (Maller, 2002)
J. Maller. Joe's Filters 3.0 for Final Cut Pro, FXScript Reference on RGB and YUV Color.

[44] (Martinkauppi, 2002)
Birgitta Martinkauppi. Face colour under varying illumination - analysis and applications. PhD thesis, Department of Electrical and Information Engineering, University of Oulu, Oulu, Finland, August 2002.

[45] (OpenCV)
Intel Open Source Computer Vision.
www.intel.com/research/mrl/research/opencv
Last Date Accessed : 07.10.2005
Author : Intel
Title : Intel Open Source Computer Vision
Update Date : 02.05.2005

[46] (Özuysal, 2004)
Mustafa Özuysal. Manual and Auto Calibration of Stereo Camera Systems. Master's thesis, Middle East Technical University, 2004

[47] (Pankanti et al., 2000)
S. Pankanti, R. M. Bolle, and A. Jain. Biometrics: The future of identification. Computer, pages 46-49, February 2000

[48] (Pennebaker and Mitchell, 1993)
W.B. Pennebaker and J. L. Mitchell. JPEG Still Image Data Compression Standard. Van Nostrand Reinhold, New York, 1993

[49] (Plaenkers and Fua, 2002)
Ralf Plaenkers and Pascal Fua. Model-based silhouette extraction for accurate people tracking. In A. Heyden et al., editor, European Conference on Computer Vision, number 2351 in Lecture Notes in Computer Science, pages 325-339, Copenhagen, Denmark, May 2002. Springer-Verlag Berlin Heidelberg

[50] (Poynton, 1996)
Charles A. Poynton. A Technical Introduction to Digital Video. John Wiley & Sons, 1996

[51] (Quek, 1995)
F. Quek. Eyes in the interface. Image and Vision Computing, Elsevier Science, 13(6):511-525, 1995.

[52] (Raja et al., 1998)
Y. Raja, S. McKenna, and S. Gong. Segmentation and tracking using colour mixture models. Lecture Notes in Computer Science, I(1351):607-614, January 1998

[53] Rehg (1995)
J. Rehg. Visual Analysis of Articulated Objects with Applications to Hand Tracking. PhD thesis, School of Computer Science, Carnegie Mellon University, April 1995

[54] (Rehg and Kanade, 1993)
J. M. Rehg and T. Kanade. Visual tracking of high DOF articulated structures: an application to human hand tracking. In J.-O. Eklundh, editor, Proc. 3rd European Conf. on Computer Vision, volume II of Lecture Notes in Computer Science 801, pages 35–46. Springer-Verlag, May 1993.

[55] (Ronfard et al., 2002)
R. Ronfard, C. Schmid, and B. Triggs. Learning to parse pictures of people. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, 7th European Conference on Computer Vision, Vol. IV, number 2353 in Lecture Notes in Computer Science, pages 700{714,Copenhagen, Denmark, May 2002. Spring.

[56] (Rourke and Badler, 1980)
O'Rourke and N.L. Badler, "Model-Based Image Analysis of Human Motion Using Constraint Propagation," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 2, pp. 522-536, 1980.

[57] (Smith, 1978)
Alvy Ray Smith. Color gamut transform pairs. In 5th International Conference on Computer Graphics and Interactive Techniques, pages 12-19, New York, USA, 1978. Siggraph - ACM Special Interest Group on Computer Graphics and Interactive Techniques, ACM Press

[58] (Stenger et al., 2001)
B. Stenger, P. R. S. Mendonça, and R. Cipolla. Model-based hand tracking using an unscented kalman filter. In Tim Cootes and Chris Taylor, editors, British Machine Vision Conference, volume 1, pages 63-72, The University of Manchester, UK, September 2001. BMVA

[59] (Stenger et al, 2001)
B. Stenger, P. R. S. Mendonça, and R. Cipolla. Model-based 3D tracking of an articulated hand. In Proc. Conference on Computer Vision and Pattern Recognition (CVPR), Kauaii, USA, December 2001

[60] (Thompson, 1981)
D. Thompson, "Biomechanics of the Hand," Perspectives in Computing, vol.1, pp. 12-19, Oct. 1981

[61] (Torresani et al., 2001)
L. Torresani, D. B. Yang, E. J. Alexander, and C. Bregler. Tracking and modeling non-rigid objects with rank constraints. In Conference on Computer Vision and Pattern Recognition. IEEE, IEEE Computer Society Press, 2001. Best Student Paper Prize

[62] (Tubiana, 1981)
R. Tubiana, ed., The Hand, vol. 1. Philadelphia, Penn.: Sanders, 1981

[63] (Wren et al., 1996)
C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, “Pfinder: Real-Time Tracking of the Human Body,” Proc. Intel Conf. Automatic Face and Gesture Recognition, Killington, Vt., pp. 51-56, Oct. 1996

[64] (Wren et al., 1997)
C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland. Pfinder: Real-time tracking of the human body. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(7):780-785, 1997

[65] (Wu and Huang, 1999)
Y. Wu and T. S. Huang. Capturing articulated human hand motion: A divide-and conquer approach. In Proc. 7th Int. Conf. on Computer Vision, volume I, pages 606–611, Corfu, Greece, September 1999.

[66] (Wu et al., 2001)
Y. Wu, Lin J. Y., and T. S. Huang. Capturing natural hand articulation. In Proc. 8th Int. Conf. on Computer Vision, volume II, pages 426–432, Vancouver, Canada, July 2001

[67] (Wyszecki and Stiles, 1967)
G. Wyszecki and W. S. Stiles. Color Science: (Wiley, New York, 1967), p.578

[68] (Wyszecki and Stiles, 2000)
G. Wyszecki and W. S. Stiles. Color Science: Concepts and Methods, Quantitative Data and Formulae.Wiley Classics Library. Wiley Inter-Science, 2nd edition, August 2000. First edition published in 1967

[69] (Von Hardenberg and Berard, 2001)
C. von Hardenberg and F. Berard. Bare-hand human-computer interaction. In Workshop on Perceptive User Interfaces, Orlando, FL, USA, November 2001. ACM.

[70] (Yang and Waibel, 1996)
J. Yang and A. Waibel. A real-time face tracker. In Third Workshop on Applications of Computer Vision, pages 142-147. IEEE, IEEE Computer Society Press, 1996

[71] (Yang et al., 1998) J. Yang, W. Lu, and A. Waibel. Skin-color modeling and adaptation. In Proceedings of ACCV, volume II, pages 687-694, Hong Kong, 1998. http

[72] (Zhu et al., 2000)
X. Zhu, J. Yang, and A. Waibel. Segmenting hands of arbitrary color. In Fourth International Conference on Automatic Face and Gesture Recognition, Grenoble, France, March 2000. IEEE, IEEE Computer Society Press

APPENDIX A

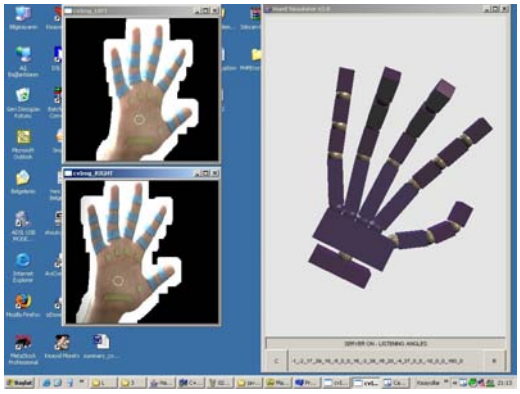
HAND TRACKING RESULTS IN 3D JAVA HAND MODEL

(1) (2)

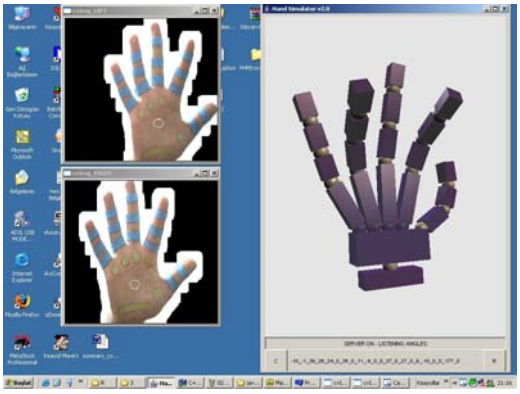
(3) (4)

(5) (6)

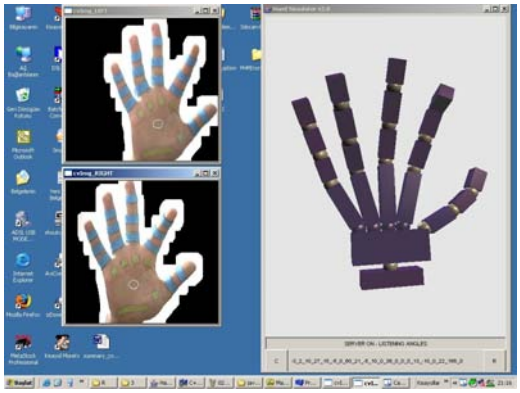
Figure A.1: Hand tracking results in 3D Java hand model



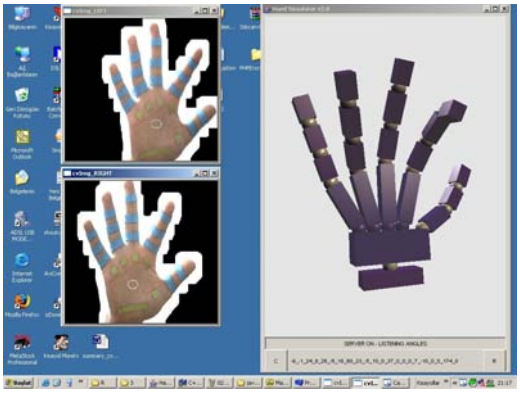
(7)



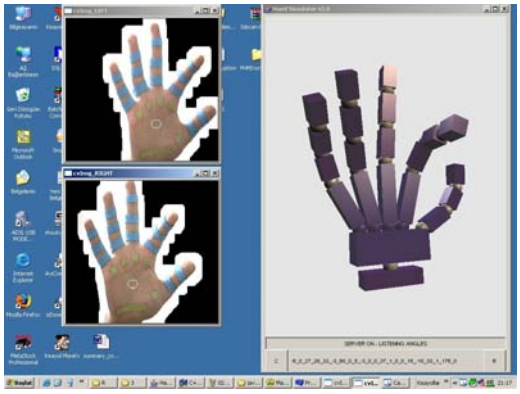
(8)



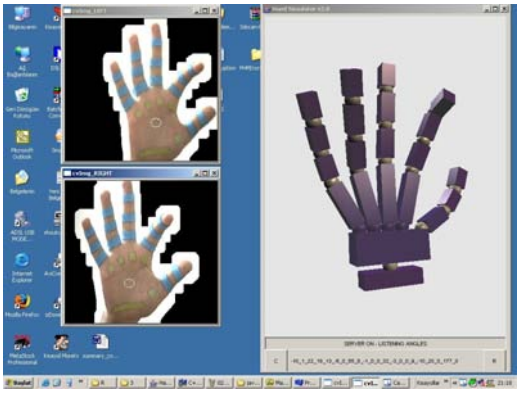
(9)



(10)

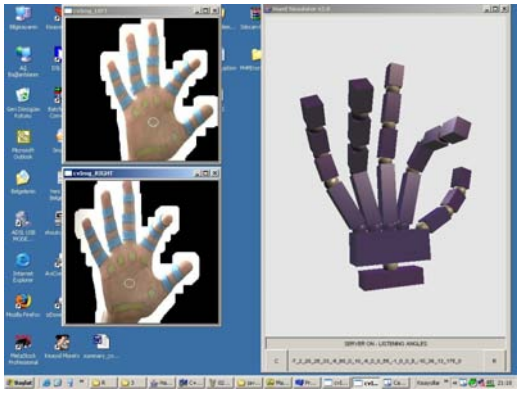


(11)

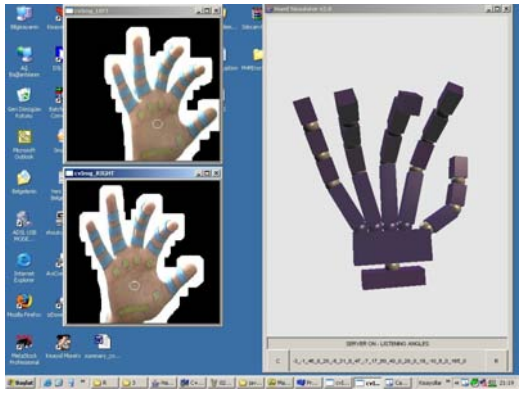


(12)

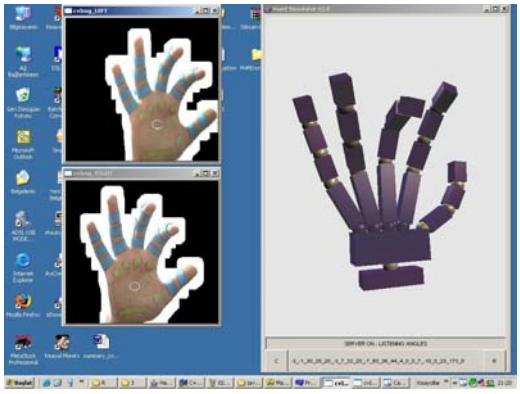
Figure A.2: Hand tracking results in 3D Java hand model



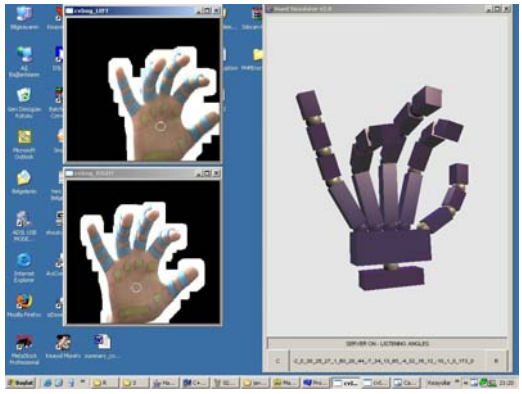
(13)



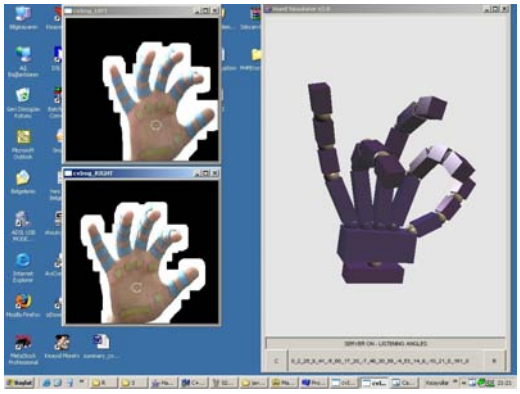
(14)



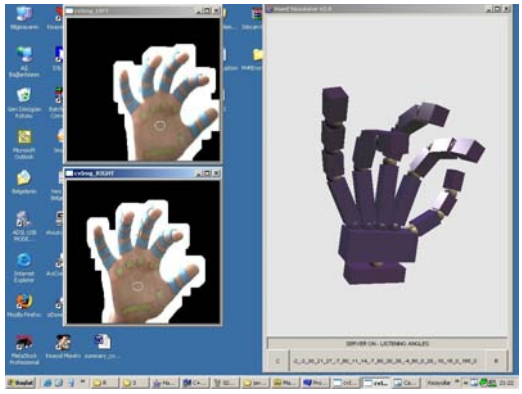
(15)



(16)

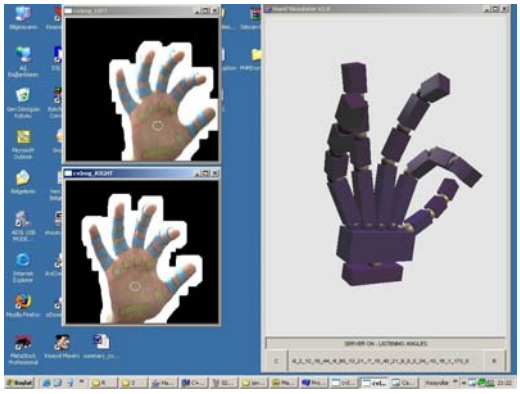


(17)

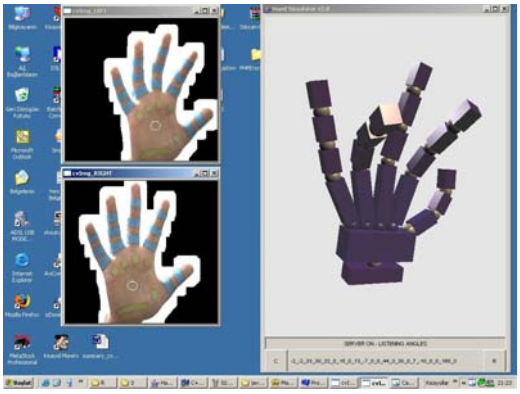


(18)

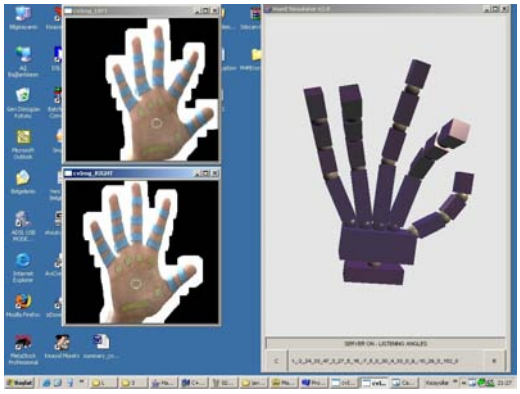
Figure A.3: Hand tracking results in 3D Java hand model



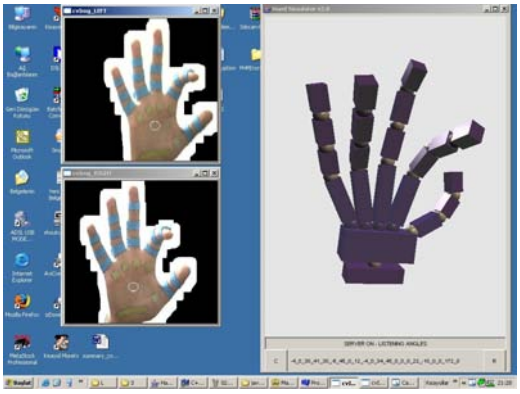
(19)



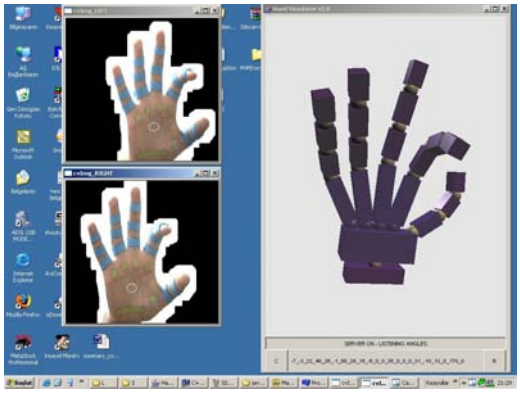
(20)



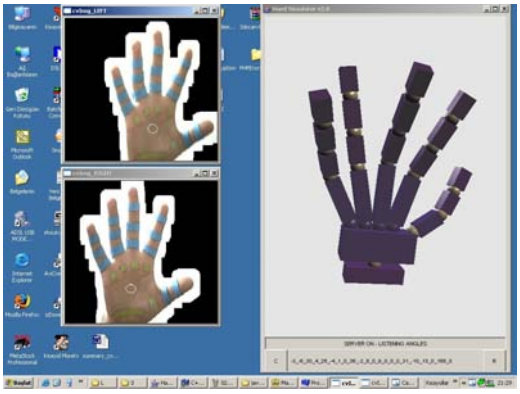
(21)



(22)

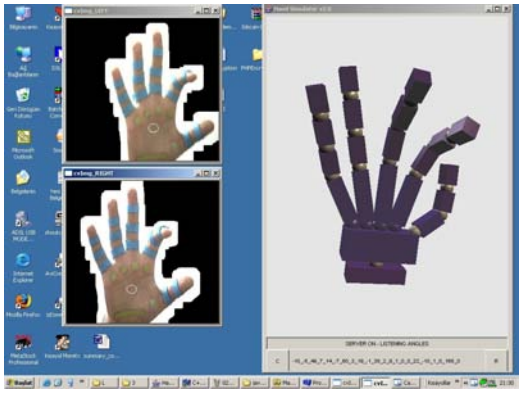


(23)

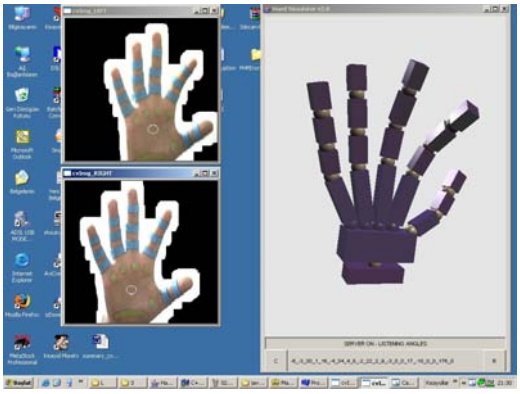


(24)

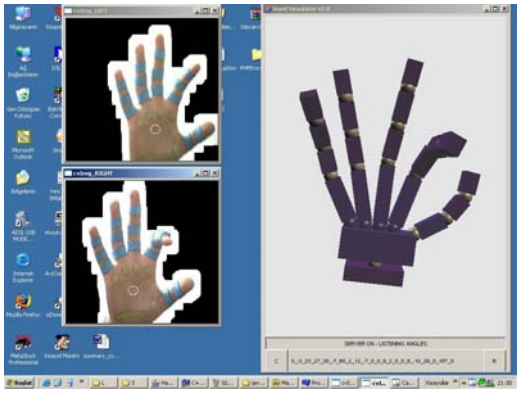
Figure A.4: Hand tracking results in 3D Java hand model



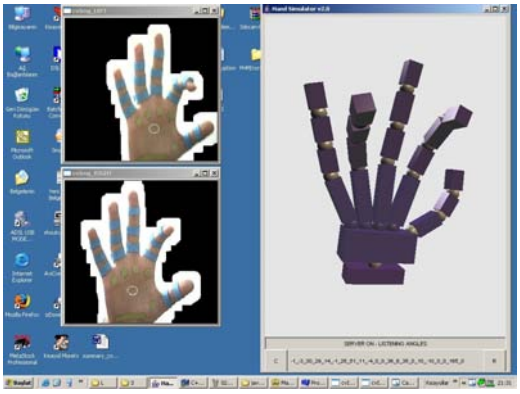
(25)



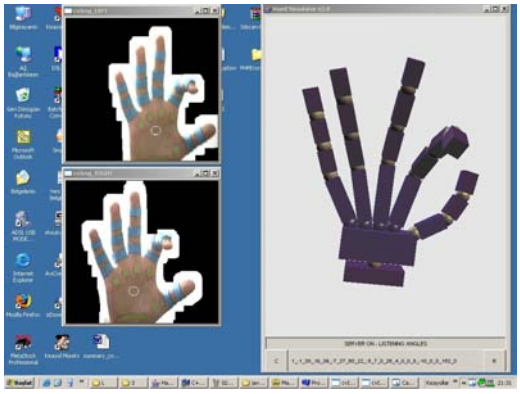
(26)



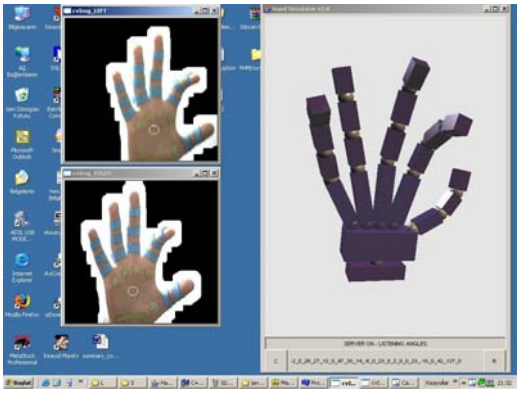
(27)



(28)

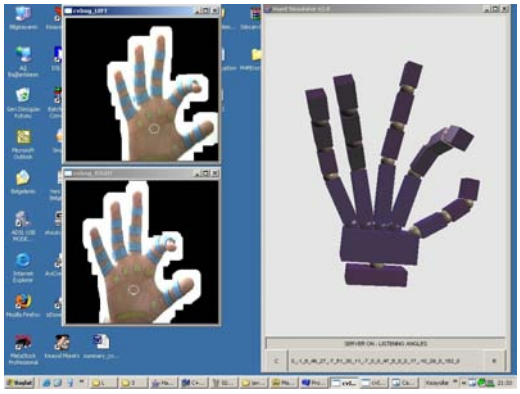


(29)

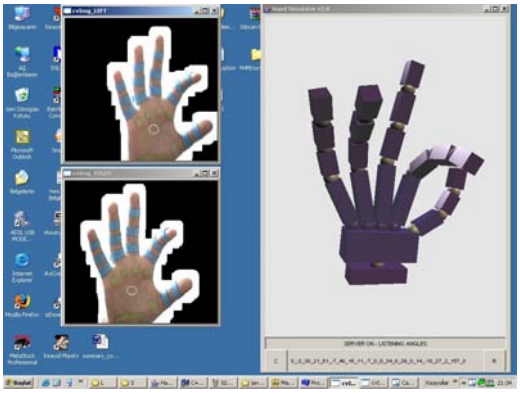


(30)

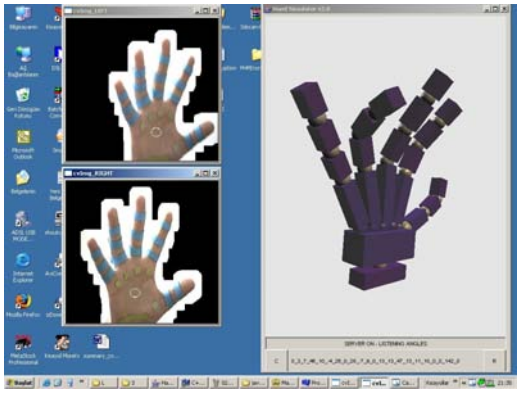
Figure A.5: Hand tracking results in 3D Java hand model



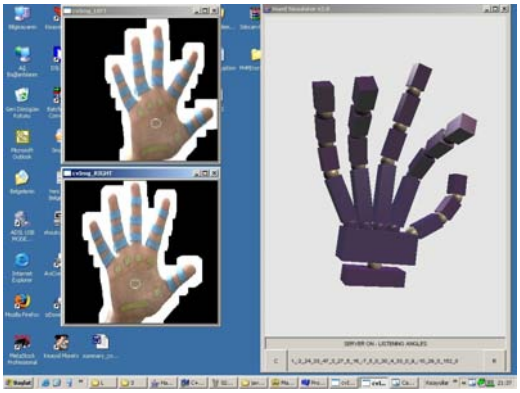
(31)



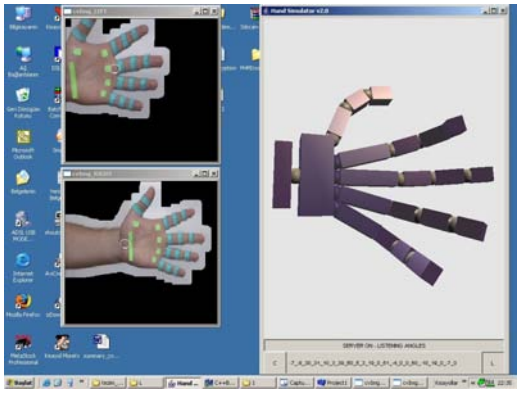
(32)



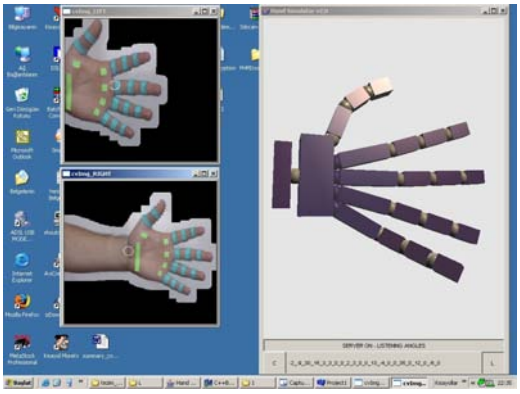
(33)



(34)

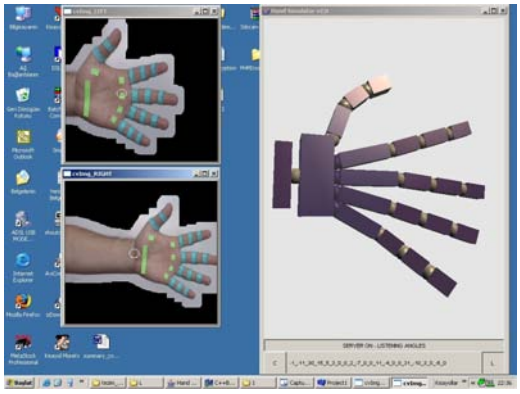


(35)

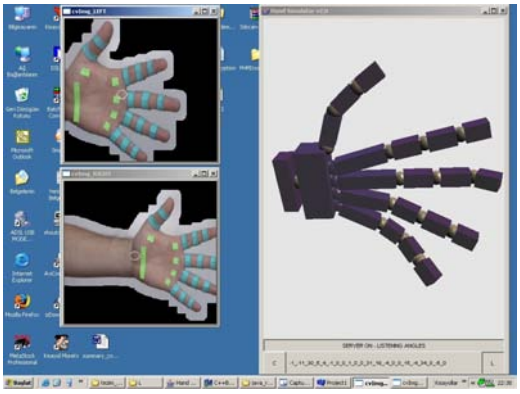


(36)

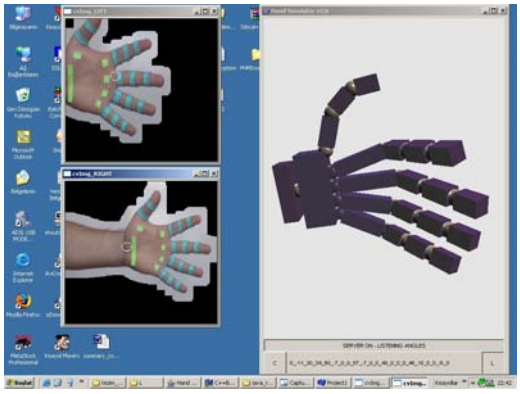
Figure A.6: Hand tracking results in 3D Java hand model



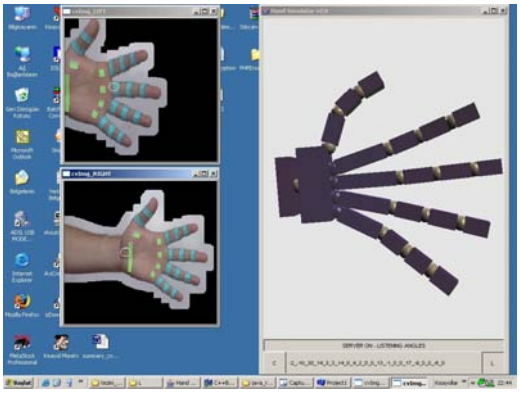
(37)



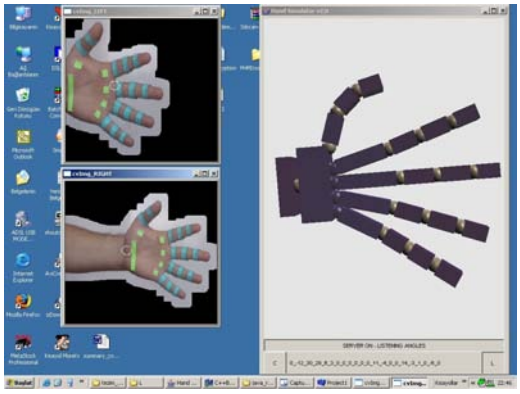
(38)



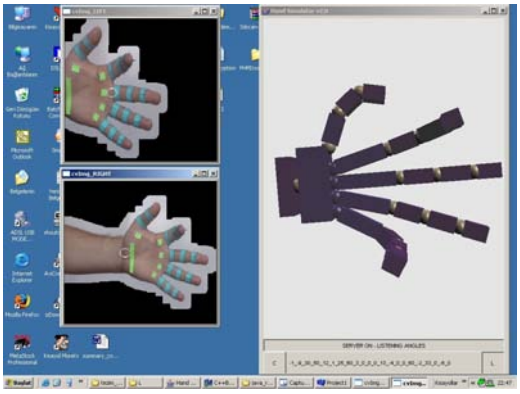
(39)



(40)

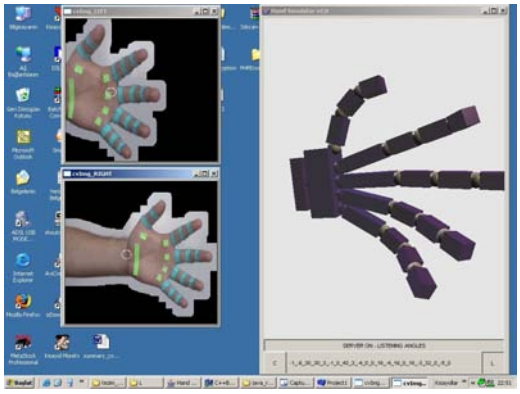


(41)

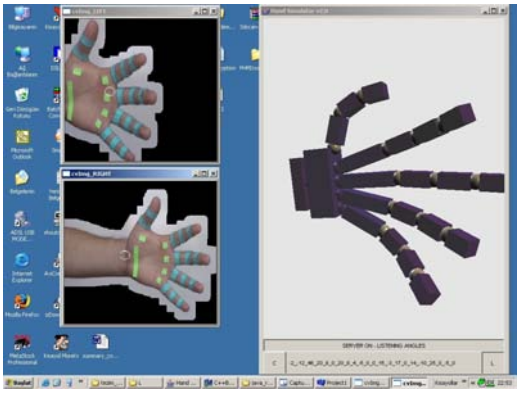


(42)

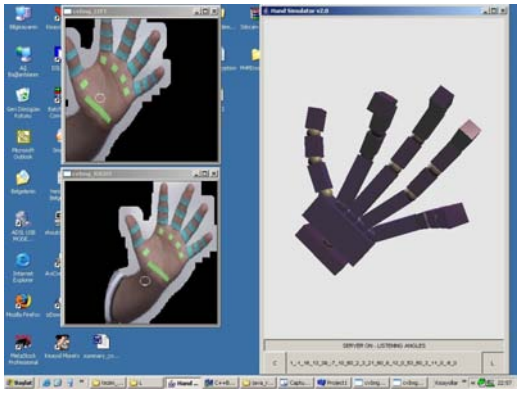
Figure A.7: Hand tracking results in 3D Java hand model



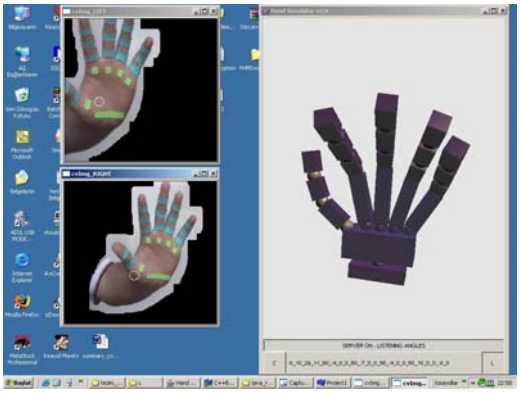
(43)



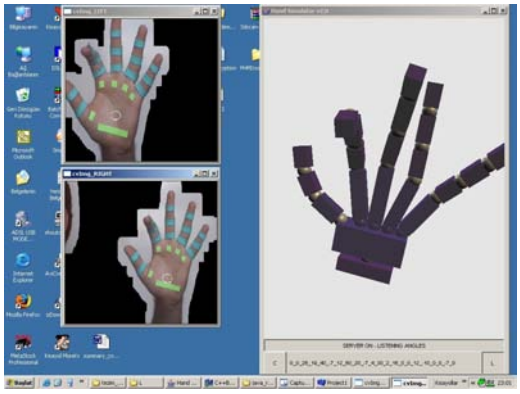
(44)



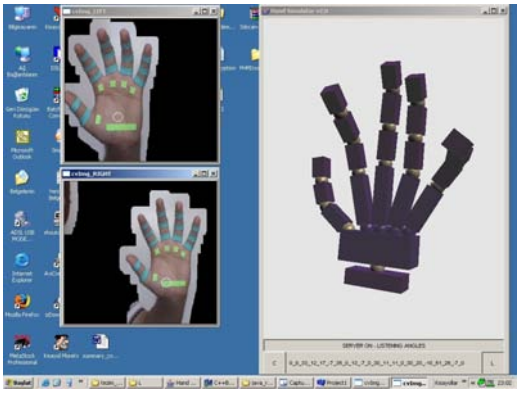
(45)



(46)



(47)



(48)

Figure A.8: Hand tracking results in 3D Java hand model