VISION BASED REAL–TIME CONTINUOUS 3D HAND GESTURE RECOGNITION INTERFACE FOR GENERIC APPLICATIONS BASED ON INPUT–OUTPUT HIDDEN MARKOV MODELS

by

Cem Keskin

BS., in Computer Engineering, Boğaziçi University, 2003

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University

2006

VISION BASED REAL–TIME CONTINUOUS
3D HAND GESTURE RECOGNITION INTERFACE FOR
GENERIC APPLICATIONS BASED ON
INPUT–OUTPUT HIDDEN MARKOV MODELS

APPROVED BY:

Prof. Lale Akarun .....
(Thesis Supervisor)

Prof. Bülent Sankur .....

PhD. Ali Vahit Şahiner .....

DATE OF APPROVAL: 19.06.2006

ABSTRACT

VISION BASED REAL–TIME CONTINUOUS 3D HAND GESTURE RECOGNITION INTERFACE FOR GENERIC APPLICATIONS BASED ON INPUT–OUTPUT HIDDEN MARKOV MODELS

This study focuses on the application of Input–Output Hidden Markov Models (IOHMM) to the recognition of 3D hand gestures that involve both hand motion and hand posture in a stereo vision–based approach. The proposed system is designed as a real time gestural interface that allows both communicative and manipulative gestures to control target PC applications. The system allows training of new communicative gestures and automatically distinguishes these from manipulative gestures in continuous streams. Uniquely colored gloves with no preset colors are used as markers, which increases the recognition rate and simplifies the hand localization problem. Camera calibration and gesture training tools are provided with the system.

The hand shape is modeled with Hu moments and the angle of the hand. As a novel approach the inclusion of normalized time information as an input to the IOHMMs is proposed, which has the effect of a continuous state variable and is shown to handle temporal information better. Another novel method is proposed for gesture spotting in IOHMM–based frameworks, which distinguishes the meaningful gestures in a continuous input stream in real time using a threshold model with a single hidden state.

The system is tested on two datasets that have 10 gestures with distinct trajectories, and 20 gestures in 10 pairs that share the same trajectories. The proposed system

is able to attain a recognition rate of 97.6% on the former dataset in 2885 trials, and a recognition rate of 94.1% on the latter dataset in 8675 trials. The gesture segmentation test results for the same datasets with the inclusion of the proposed threshold model are 97.1% for the former and 93.8% for the latter dataset.

ÖZET

GİRDİ–ÇIKTI SAKLI MARKOV MODELLERİ İLE GERÇEK ZAMANLI OLARAK ÜÇ BOYUTLU EL HAREKETLERİNİN TANINMASINA DAYANAN GÖRÜ TABANLI ETKILEŞİMLİ ARAYÜZ

Bu çalışmanın amacı Girdi–Çıktı Saklı Markov Modelleri kullanarak hem hareket hem şekil değişimi içeren üç boyutlu el hareketlerini stereo görüntüler üzerinden gerçek zamanlı olarak tanıyacak bir etkileşimli arayüz geliştirmektir. Önerilen sistem kullanıcıların hem manipülatif hem haberleşmeye yönelik doğal el hareketleriyle PC uygulamalarını kontrol etmelerini sağlayacak şekilde tasarlanmıştır. Bu iki tip el hareketi sistem tarafından gerçek zamanlı olarak ayırdedilebilmektedir. Uygulama, etkileşimli hareket eğitimi ve kamera kalibrasyonu için birer yazılım aracı da içerir. Eli ayırdetmek için kullanılan renkli eldivenler her renk olabilir, ve el ayırdetme problemini basitleştirerek tanıma başarısını arttırırlar.

Önerilen sistemde elleri modellemek için el şekli özniteliği olarak Hu momentleri ve elin açısı kullanıldı. Yeni bir yaklaşım olarak düzgelenmiş zaman bilgisinin Girdi-Çıktı Saklı Markov Modellere girdi olarak verilmesinin bu modellerin saklı durumlarının sürekli bir yapı kazanması sağlandı ve bunun el şekli tanımadaki başarı oranını arttırdığı gösterildi. Ayrıca Girdi–Çıktı Saklı Markov Modelleri üzerine kurulu tanıma sistemleri için tanımlı el şekillerini tanımsız hareketlerden ayıracak yeni bir uyarlanır eşik modeli tasarlandı.

Önerilen sistem farklı hareketler içeren on şekilden oluşan ve aynı hareketleri paylaşan on çift el hareketinden oluşan iki ayrı el şekli veritabanı üzerinde test edil-

miştir. Sistem ilk veritabanı üzerinde 2885 denemede 97.6%, ikinci veritabanında ise 8675 denemede 94.1% tanıma başarı oranı göstermiştir. Bu oranlar uyarlanır eşik modeli ile sürekli bir girdi üzerinden tanıma yapıldığında ilk veritabanı için 97.1%, ikinci veritabanı içinse 93.8% çıkmıştır.

TABLE OF CONTENTS

ABSTRACT iii
ÖZET v
LIST OF FIGURES x
LIST OF TABLES xiii
LIST OF SYMBOLS/ABBREVIATIONS xiv
1. INTRODUCTION 1
1.1. Gesture Taxonomy and Modeling 3
1.2. Hand Shape Modeling 5
1.2.1. Appearance Based Approaches 5
1.2.2. Model–Based Approaches 7
1.3. Feature Extraction 9
1.4. Gesture Recognition 13
1.4.1. Gesture Spotting 17
1.5. Proposed Approach 17
1.6. Outline of the Thesis 20
2. GESTURE RECOGNITION USING HIDDEN MARKOV MODELS 22
2.1. Gesture Recognition using an HMM–Based Framework 22
2.1.1. Recognition 24
2.1.2. Training 26
2.2. Gesture Recognition using an IOHMM–Based Framework 28
2.2.1. Recognition 29
2.2.2. Training 31
3. METHODOLOGY 36
3.1. Hand Tracking 36
3.1.1. Hand Detection 37
3.1.1.1. Marker Registration 39

3.1.2. Hand Region Extraction 44
3.1.2.1. Finding The Hand 45
3.1.2.2. Extracting Hand Features 49
3.1.3. Filtering the Hand Trajectory 52
3.1.4. Estimating the 3D Hand Trajectory 56
3.1.4.1. Camera Calibration 57
3.1.4.2. Stereo Reconstruction 59
3.2. Hand Gesture Recognition 61
3.2.1. Vector Quantization 64
3.2.2. HMM–Based Hand Gesture Recognizer 65
3.2.2.1. Left–Right HMMs 68
3.2.2.2. Automatic Estimation of the Number of States 70
3.2.2.3. Gesture Spotting 70
3.2.3. IOHMM–Based Hand Gesture Recognizer 72
3.2.3.1. Left–Right IOHMMs 76
3.2.3.2. Neural Network Architectures 79
3.2.3.3. Training 80
3.2.3.4. Estimating the Number Of States 83
3.2.3.5. Input Sequence 83
3.2.3.6. Adaptive Threshold Model 85
4. EXPERIMENTS 87
4.1. Datasets 87
4.2. Training 88
4.2.1. Training Parameters 89
4.2.2. Hidden Nodes vs. IOHMM States 90
4.2.3. 5×2 Cross Validation 91
4.3. Test Results 91
4.3.1. Parameter Optimization Tests 91
4.3.2. Recognition Performance Tests 95

4.3.2.1. Hand Signs with Distinct Hand Motions 95
4.3.2.2. Hand Signs with Unrestricted Hand Motions 96
4.4. Reasons for Error 100
4.5. Hand Tracking Computation Time Performance 100
5. CONCLUSIONS 102
5.1. Remarks and Future Work 102
APPENDIX A: APPLICATION 104
A.1. Introduction 104
A.1.1. Requirements 105
A.2. Main Phases 105
A.2.1. System Setup 105
A.2.2. Preparation of a Gesture Database 107
A.2.3. Execution 108
REFERENCES 111

LIST OF FIGURES

Figure 1.1. Hand gestures 2

Figure 1.2. Hand motion vs. hand posture 3

Figure 1.3. Hand gesture taxonomy 4

Figure 1.4. Appearance–based approach 6

Figure 1.5. Geometric hand model 8

Figure 1.6. Hand localization 9

Figure 1.7. Detecting fingertips using a marker 12

Figure 1.8. HMM architecture 14

Figure 1.9. IOHMM architecture 16

Figure 1.10. Gesture spotting 18

Figure 1.11. Flowchart of the system 19

Figure 3.1. Hand tracking 36

Figure 3.2. Significantly different regions estimated using different thresholds . 41

Figure 3.3. Effect of secondary criteria in marker registration 42

Figure 3.4. RGB histogram of a final difference image 43

Figure 3.5. Red, green and blue components of the RGB histogram 44

Figure 3.6. Effect of using very low thresholds for both R_{low} and R_{high} 47

Figure 3.7. Problems with double thresholding using static thresholds 48

Figure 3.8. Results of using double thresholding with dynamic thresholds . . . 48

Figure 3.9. Result of filtering the hand trajectory 55

Figure 3.10. The special object used for the calibration process 59

Figure 3.11. Camera setup and 3D reconstruction 60

Figure 3.12. Left–Right Hidden Markov Model 69

Figure 3.13. Left–Right IOHMM architecture 76

Figure A.1. Camera setup for Hand Gesture Interface 106

Figure A.2. Camera calibration using the internal tool 107

Figure A.3. Training interface 108

Figure A.4. Possible gestures 108

Figure A.5. Training interface 109

Figure A.6. Options interface 110

LIST OF TABLES

Table 3.1. Vector quantization classes 66

Table 4.1. 2D Dataset 87

Table 4.2. 3D Dataset 88

Table 4.3. Optimized training parameters 92

Table 4.4. Best H values for given K' 93

Table 4.5. Best K' values for given H 94

Table 4.6. Optimized architecture parameters 94

Table 4.7. Recognition rate for hand gestures with distinct hand motions with normalized time Γ as Input 96

Table 4.8. Recognition rate for hand gestures with distinct hand motions without the time input 97

Table 4.9. Recognition rate for unrestricted hand gestures with the time input 98

Table 4.10. Recognition rate for unrestricted hand gestures without the time input 99

LIST OF SYMBOLS/ABBREVIATIONS

a_{ij}	Probability of making a transition from the state i to j
$a_{ij}(t)$	Time dependent state transition probability
A	State transition matrix of a hidden Markov model
$A(t)$	State transition matrix of an input–output hidden Markov model
$b_j(m)$	Probability of observing the symbol v_m in state j
$b_{j,t}(m)$	Time dependent observation probability
B	Observation matrix of a hidden Markov model
$B(t)$	Observation matrix of an input–output hidden Markov model
D	Number of input dimensions
E	Matrix formed to solve for the calibration matrix parameters
E'	Matrix formed to solve for the three dimensional hand coordinates
F	State transition matrix for a linear dynamical system
$g_t(i)$	State posterior probability of a input–output hidden Markov model
$h_t(i,j)$	State transition posterior probability of a input–output hidden Markov model
H	Observation matrix for a linear dynamical system
H_N	Number of hidden nodes of state transition networks
H_O	Number of hidden nodes of emission networks
I_i	Hu moments
$I(x,y)$	Grayscale image
K	Kalman gain
K'	Number of distict states of an input–output hidden Markov model
M	Number of distinct observable symbols
M_e	Maximum number of epochs allowed

M_{ij}	Image moments
N	Number of distinct hidden Markov model states
$N_{2D/3D}$	Number of total hand features
N_i	State transition network of state i
$N_{S_{hue/sat}}$	Number or pixels in the bin
o_j	Unnormalized output of a network
O	Observation sequence
O_i	Emission network of state i
O_t	Observation symbol at time t
P	Calibration matrix
P'	Prior error covariance
$P_i^{hue/sat}$	Hue or saturation value of the pixel i
q_t	Value of the state variable at time t
Q	State sequence
Q'	System error covariance
R	Measurement error covariance
$R_{2D/3D}(t)$	Radius of no motion
R_{low}	Low range of pixel values
R_{high}	High range of pixel values
$S_{hue/sat}$	The bin with the maximum number of pixels
S_i	State i of a hidden Markov model
T	Sequence length
Th_{low}	Low threshold
Th_{high}	High threshold
v	Measurement noise variance
v_{jh}	Hidden weights of a network
v_m	Distinct observation symbol
$\vec{V}$	Velocity vector
w	System noise variance

w_{hn}	Weights of a network
$\vec{x}$	Image coordinate vector
$\mathbf{x}$	Input vector of a network
x_t	Value of the input variable at time t
$\bar{x}$	Horizontal component of the centroid
X	Training set
$\vec{X}$	Hidden state vector
$\vec{X}(t)$	Hidden state of a linear dynamical system
$\bar{y}$	Vertical component of the centroid
y_j	Normalized output of a network
$\vec{Y}$	Hand coordinate vector
$\vec{Y}(t)$	Observation in a linear dynamical system
z_h	Value of a hidden node of a network
α	Scalar factor
$\alpha_t(i)$	Forward variable
$\beta_t(i)$	Backward variable
δ	Decay rate
ϵ	Convergence threshold
$\varphi_{ij,t}$	Output of a state transition network
$\gamma_t(i)$	State posterior probability of a hidden Markov model
$\Gamma(t)$	Normalized time input
η_{ij}	Scale invariant central moments
$\eta_{im,t}$	Output of an emission network
λ^+	Increase in learning factor
λ^-	Decrease rate in learning factor
μ	Learning factor
$\mu_{hue/sat}$	Mean of the hue or saturation component
μ_{pq}	Central moments

π_i	Probability of initially being at state i
$\mathbf{\Pi}$	Vector containing the initial state probabilities
θ	Angle of the hand
Θ	Parameter set
ρ_{jk}	Generic weight of a multilayer perceptron
$\sigma_{hue/sat}$	Standard deviation of the hue or saturation component
$\vec{\upsilon}$	Measurement noise
$\vec{\omega}$	System noise
Ω	Auxiliary function used for the generalized expectation maximization procedure
$\xi_t(i,j)$	State transition posterior probability of a hidden Markov model
ψ	Memory size

2D	Two Dimensional
3D	Three Dimensional
ANN	Artificial Neural Network
CPU	Central Processing Unit
DOF	Degree of Freedom
DTW	Dynamic Time Warping
EM	Expectation Maximization
GEM	Generalized Expectation Maximization
GUI	Graphical User Interface
HCI	Human Computer Interaction
HMM	Hidden Markov Model
HSI	Hue Saturation Intensity
IOHMM	Input–Output Hidden Markov Model
LDS	Linear Dynamical System
MHI	Motion History Images
MLP	Multilayer Perceptron

OS	Operating System
RGB	Red Green Blue
RNN	Recurrent Neural Network
SSSM	Switching State Space Model
TDNN	Time–Delay Neural Network
VQ	Vector Quantization

1. INTRODUCTION

In the last decades computers have become an increasingly important part of our daily lives. As each tool gets computerized, new interfaces are designed to allow more natural control of the device. Yet most of the computer applications use keyboards and mice, which inherently limit the speed and naturalness of human computer interaction (HCI).

The use of hand gestures is a noteworthy alternative to cumbersome and unnatural interface devices such as keyboards and mice for HCI. The degree of freedom (DOF) provided by such interfaces often presents a bottleneck in applications such as virtual reality, which require manipulation of higher dimensional information. The processing power of today's computers allows development of real time applications that make use of visual recognition and interpretation of hand gestures, which increases the DOF, while providing the ease and naturalness desired for HCI.

Hand gestures are a form of non–verbal communication that makes use of hand trajectories as well as hand postures. The gestures that involve only hand motion or hand posture have no distinctive name in the literature, even though their distinction is significant from the point of view of pattern recognition. Throughout this thesis the gestures that involve only hand motion and no posture information will be called hand trajectories, and hence their recognition process will be called hand trajectory recognition. Similarly recognition problems that only involve hand pose estimation will be called hand posture recognition.

Hand gestures range from simple manipulative gestures that aim to interact with objects, to complex gestures that are used to express feelings or communicate via a sign language. Accordingly, an HCI system may involve simple gestures for 2D



(a) Natural hand gesture (b) Unnatural hand gesture

Figure 1.1. Hand gestures

[1, 2, 3, 4, 5, 6, 7] or 3D virtual environments [8, 9, 10, 11], virtual reality applications [12, 13, 14], robotics applications [15, 16, 17, 18], augmented desktop applications [19, 20], or complex hand gestures for sign language recognition tasks [21, 22, 23, 24, 25, 26, 27, 28]. The aim of most of these applications is to use hand gestures to overcome the restrictions other input devices impose. The most complex of these problems is sign language recognition, where the gestures must be identified within a contextual frame. This problem has recently drawn much attention, since implementation of a real time system that would allow deaf people to communicate through their natural modality, is an increasingly attainable goal.

Interpretation of hand gestures requires that configurations of the human hand to be measurable. Earlier approaches that attacked this problem made use of mechanical devices that directly measure hand joint angles and spatial position. Many researchers used this glove–based approach for robustness and efficiency [29, 30, 31, 32, 33]. This approach requires the user to wear a cumbersome device that hinders the naturalness of the hand gestures. With the development of powerful computers that can interpret hand gestures in real time using computer vision–based methods, these costly and obtrusive devices can be omitted. In vision–based approaches hand gestures are visually interpreted using cameras and computer vision techniques.



(a) Hand trajectory (b) Dynamic hand posture

Figure 1.2. Hand motion vs. hand posture

The approaches differ in the particular aspects of hand gestures they focus on, which can be tracking, trajectory classification, pose estimation or posture classification, i.e. the dynamic and static characteristics of the hand gestures. Important issues related to these approaches are modeling of the gesture, training of the system and recognition of gestures. This chapter will cover the approaches used for these issues, review related works and state the proposed approach of the thesis.

1.1. Gesture Taxonomy and Modeling

For the general hand gesture recognition problem both the trajectory and the hand shape need to be modeled. The choice of the hand model primarily depends on the target application. If the purpose is a natural interaction, a model that allows as many as possible natural gestures to be interpreted by the computer has to be established. Real time HCI applications often prefer to employ simpler models , since speed is an important concern of such applications.

Hand gestures can be defined as "stochastic processes in the gesture model parameter space over a suitably defined time interval" [34]. Therefore each gesture performed is a trajectory in the model parameter space, and training and recognition problems

are learning, classification of parameters of a gesture model.

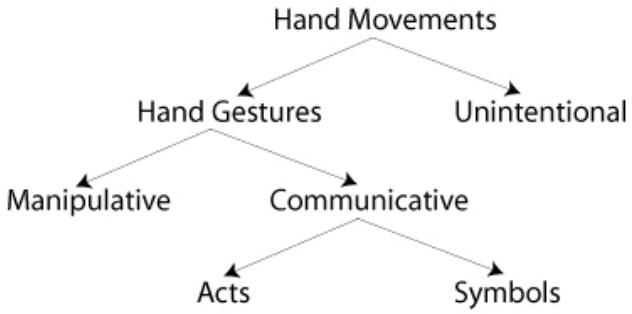


Figure 1.3. Hand gesture taxonomy

A taxonomy of hand gestures is given in Figure 1.3. According to this taxonomy not all of the hand movements are classified as gestures, namely the unintentional hand motions are not considered to be meaningful hand gestures. Hand gestures used in HCI systems are usually either communicative or manipulative. Manipulative hand gestures are used to act on objects that are usually virtual, whereas communicative gestures are either dynamic acts or static symbols, which have communicational purpose.

Usually HCI applications focus on a subset of these classes of gestures and consider only these classes as meaningful. The proposed system in this thesis aims to recognize and classify every possible gesture by filtering the unintentional movements and distinguishing the communicative gestures from manipulative ones. A previous work also aimed to distinguish communicative gestures [8]. In this work the defined gesture models were used to communicate with target generic applications in order to control them, whereas the rest of the movements were used to simulate the mouse click and motion.

The description of gestures inherently involves the characterization of their spatial properties, since hands are flexible objects and gestures are performed in 3D space. The motion and the change in the shape of the hand has different characteristics and hence,

they are often modeled separately. The models that are used to describe hand shapes are reviewed next.

1.2. Hand Shape Modeling

The main purpose of hand shape modeling is to describe the hand in a parameter space that can be used to distinguish hand postures in the recognition phase. Thus the hand pose description may differ according to the intended application domain. A simple HCI application may only be using basic descriptors of the hand, such as openness or closeness of the fingers, whereas a sign language recognition system may need each angle of finger articulations. As reviewed in the first section of this chapter, there are mechanical devices with sensors, which are usually referred to as data gloves that can accurately compute each articulation angle. On the other hand computer vision–based approaches are cheaper and less obtrusive. There are two main approaches for hand shape modeling in computer vision–based systems:

- Appearance–based approaches
- Model–based approaches

1.2.1. Appearance Based Approaches

Appearance–based approaches are not based on prior and explicit knowledge of the hand model, hence the model parameters are not directly derived from the 3D spatial description of the hand. Such methods learn to relate the appearance of a given gesture to a set of predefined gestures. Therefore the classification and discrimination of gestures are based on similarities in image features. Training set may be gathered via data gloves for accuracy [35].

As an appearance–based model the 2D hand image sequences can be used to

describe gestures. Usually monoscopic or stereoscopic image sequences are used to model 2D and 3D gestures, where the model parameters are either images themselves or predefined features extracted from the images. A recent approach is to model gestures by Motion History Images (MHI), which are images formed by accumulation of the motion of every single pixel over some temporal window [36]. The intensity of pixels in such images refer to how much prolonged gestural motion is detected at that pixel.

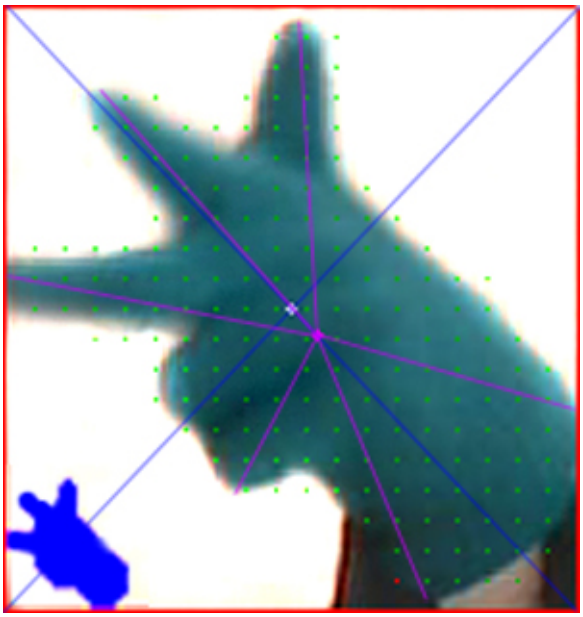


Figure 1.4. Appearance–based approach

The most common method used in appearance–based approaches is using low level features derived from the images to model gestures, which are usually the same parameters used in the recognition phase. These parameters include image moments, image eigenvectors, which can be called eigenhands in this context, contours and edges, orientation histograms and Zernike moments. Contours can be obtained from the result of simple edge detection techniques. When expressed in polar coordinates, they are called "signatures".

Among these methods the eigenspace approach provides an efficient representa-

tion of the feature points of the gesture using a small set of basis vectors. Turk and Pentland proposed using the eigenspace representation for recognition [37]. Following the success of "eigenfaces", the eigenspace approach is commonly applied to hand gestures [2, 38, 39, 40]. Either the hand shape or the contour can be used as the initial feature set in this approach.

In the case of pure hand gesture recognition the exact hand region is not required. Extracting low–level image measurements is fast and robust to noise, which makes it suitable for real time applications. As low–level features, centroid of the hand region, principle axes defining an elliptical bounding region of the hand, the optical flow of the hand region, or coordinates of a finger tip can be used [8].

In [15] the authors make use of a graph based method, namely Elastic Graph Matching to recognize hand shapes. Image moments are also commonly used as shape descriptors, which are easily calculated from hand/arm silhouettes or contours. Starner and Pentland followed this approach in their Hidden Markov Model based sign language recognition system [21].

There are other methods such as deformable 2D templates or simple models that make use of the distance between the palm and the fingertip, which is a coarse approximation to the motions human hand is capable of performing, since human hand is a very flexible object. In pure hand trajectory recognition systems, however, this latter approach may be more suitable, since hand posture information is not used [8]. Also real time HCI applications may consider this approach due to its speed.

1.2.2. Model–Based Approaches

Model–based methods try to minimize the difference between a predefined 3D model and the features extracted from the input image sequence. Primitives such as the

locations of the finger tips, the difference between contours and the silhouette features are used to measure the resemblance between the hand posture and the estimated configuration of the model. The 3D hand models can be classified in two groups:

- volumetric models
- skeletal models



Figure 1.5. Geometric hand model

Volumetric models describe the 3D visual appearance of the hand, whereas skeletal models define a skeletal structure for the hand, which consists of joint angle parameters and segment lengths, and impose restrictions on the motions the hand is capable of. Model–based methods reportedly have high time complexities, therefore they are not suitable for real time HCI applications [41].

1.3. Feature Extraction

The goal of feature extraction is to estimate the parameters of the gesture model from the video images of the hand. Feature extraction implicitly involves the task of detecting and localizing the hand in the input images. Localization is the extraction of the hand image from the input image. Usually either the color or motion information is used in the localization process.

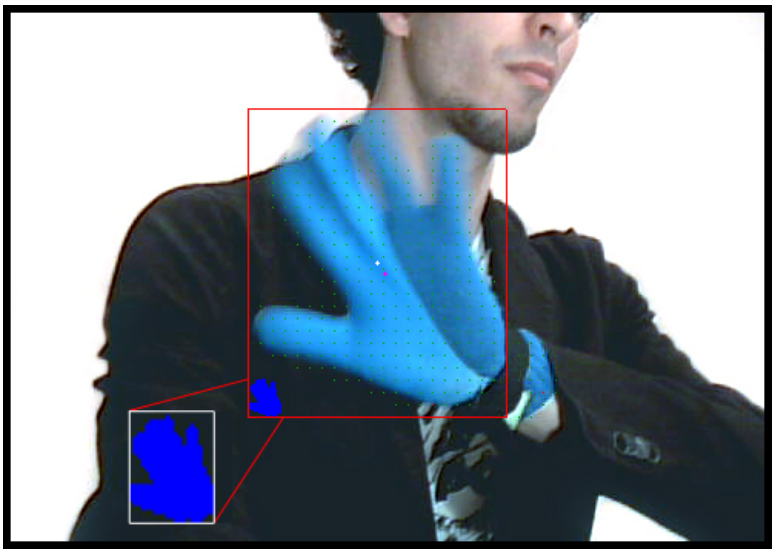


Figure 1.6. Hand localization

Mean shift procedure is proposed as a method for detection and clustering of the skin color. Mean shift is a kernel–based density estimation technique that has been used in many applications including data clustering, color image segmentation, and object tracking [42].

Color spaces that are best suited for hand localization have been thoroughly researched. Zarit *et al.* compared CIE L*a*b*, Hue Saturation Intensity (HSI), rg–chromaticities, YCbCr, and the log–opponent space [43]. They report that the HSI and the log–opponent space perform best using lookup tables, whereas for the Bayesian approach the choice of the color space had no influence on the result. Martinkauppi *et*

al. conclude in their paper that other color spaces do not produce better results than NCC Red Green Blue (RGB), and that the behavior of skin chromaticity is dependent on the selected camera [44]. Many other researchers reported that no single color space has a distinct advantage over the others. Nevertheless the HSI color space is commonly used in tracking applications due to its ability to handle changes in illumination in a simple and robust manner.

Skin color based algorithms fail to distinguish the hand in input images if there are other body parts in the scene. Therefore either advanced tracking methods such as particle filtering [45], or uniquely colored gloves are used [8, 13, 46]. The use of colored gloves makes it possible to localize the hand efficiently in real–time, but imposes the a restriction on the user and the interface setup. Without these restrictions, however, color–based localization techniques are usually computationally intensive and currently hard to implement in real–time.

The motion cue is also used to localize the hand, using certain assumptions such as:

- there is one gesturer;
- the gesturer is stationary;
- only the hand and the arm are moving.

Therefore, it is assumed that the main component of motion in the image is the motion of the hand, which can be used for localization. This localization approach is used in [8, 47, 48]. Specifically in [8] the motion cue is used for marker registration, which allows the system to learn the color parameters of the marker. This color is then used to localize the marker in each image.

The image features employed to extract the recognition parameters are usually

very similar in different approaches. Most of the appearance–based approaches make use of the extracted hand region image or the fingertip. This is computationally an easy task and is suitable for real time applications.

Hand silhouettes are among the simplest and most frequently used features, and they are used both for appearance–based and model–based methods [41]. Silhouettes are easily extracted using color or motion cues as reviewed. The disadvantage is that using these features usually results in lack of information that may hinder the 3D pose estimation performance of the system.

Fingertip locations are frequently used as features, since they can be used to obtain parameters of both the 3D hand models and the 2D appearance–based models. The previous work on trajectory recognition used fingertip locations as the features to detect and track [8]. The fingertip detection problem is usually simplified by using marked gloves or colored gloves. Pattern matching techniques can also be used to detect fingertips.

Once the hand is localized, the model parameters should be extracted, which is usually followed by gesture recognition. The feature extraction phase depends on the model parameters and the features that have been selected. The estimation of the 3D model of the hand is a complex task [34, 41]. Therefore, model–based methods are currently not suitable for real time applications. A more suitable approach for real time HCI applications is using appearance–based parameters.

Appearance–based approaches usually use compact descriptions of the image sequences. The parameters can be selected as sets of gesture images describing the gesture [49]. Eigenspace approaches can be used to describe images with respect to the mean image, which is a common method in eigenface methods. The MHI images can be used for gestures that involve significant global motion, which makes them harder to use for

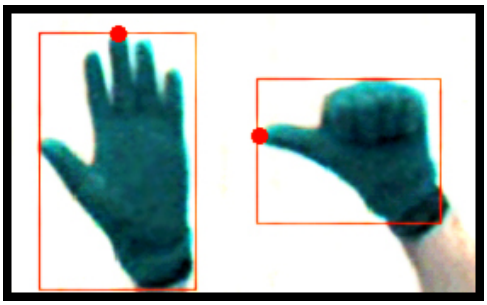


Figure 1.7. Detecting fingertips using a marker

manipulative gestures. The most commonly used feature is once again the silhouettes of the hands, which is a relatively simpler and faster method. The geometric moment description of hand shapes are usually extracted from the silhouettes, which are also commonly used [21, 50].

Some other techniques use Zernike or Hu moments [51], whose magnitudes are invariant to rotation, thus allowing for rotation invariant shape classification.

A major disadvantage of these models is their sensitivity to noise. Therefore the exact hand region needs to be known at all times during the hand motion [50], and the HCI system should employ means of motion prediction, which are usually based on simple Newtonian dynamics models and Kalman–based predictors.

Kalman filtering is widely applied to model the dynamics of the hand for tracking purposes [52]. Kalman Filter views the hand motion as a linear dynamical system (LDS), which is a partially observed stochastic process with linear dynamics and linear observations, both subject to Gaussian noise. Kalman Filter has been extensively used in computer vision applications.

1.4. Gesture Recognition

The final phase of the hand gesture recognition problem is the recognition and classification of extracted hand feature sequences. To choose a suitable model for recognition of general hand gestures, the specific properties of hand gestures should be discussed.

The approaches to gesture classification problem can be categorized as:

- rule–based methods
- machine learning–based methods

Rule–based methods aim to encode gestures as a set of rules, which relies on others' abilities to encode gestures as rules and does not allow training of the system for new gestures, which seriously limits the capabilities of an HCI application. An example of such an approach is given in [53].

Machine learning methods try to model the gestures with flexible structures that allow training from a set of positive examples. This is a suitable approach for HCI applications, since it simplifies introduction of new gestures to the system. Moreover machine learning methods can be designed to be adaptive to specific conditions.

Dynamic hand gestures are sequential data that involve both temporal and spatial context. Due to their temporal context, gestures are time scale and time instance invariant. Therefore the time needed to perform the same gesture is allowed to differ, and the specific time a gesture starts does not affect the recognition. This is called the temporal invariability of the gestures. Furthermore, the absolute spatial coordinate a gesture starts from, as well as the spatial size of the gesture, may not be important for an HCI application. The existence of both types of requirements is called the

spatio–temporal invariability of hand gestures. The chosen recognition model should be able to handle such sequential data in order to accurately handle the hand gesture recognition problem.

A common approach is to treat the gesture as the output of a partially observable stochastic process. It is assumed that the conditional probability density of the current event depends only on recent events, which is called the Markov property. This architecture can be viewed as a probabilistic network with hidden and observable states. Hence this architecture is called a Hidden Markov Model (HMM), which is the most common method used in the literature for classifying gestures due to its tremendous success in modeling spatio–temporal time series [26, 27, 54, 55]. The HMM structure is explained in detail in Chapter 2.

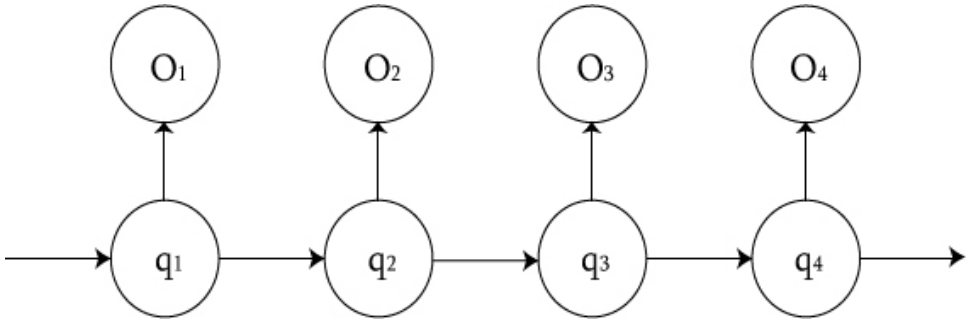


Figure 1.8. HMM architecture

At each time instance the HMM is in one of its hidden states. Transitions between the hidden states and the observations are governed by probabilistic rules. The hidden state space is discrete, whereas the observations are allowed to be continuous or discrete. In the case of continuous observations, the emission probabilities of states can be represented with mixture of Gaussian probability density functions.

The observations of hand gestures are the extracted features from the images, which can be image moments, eigenimage coefficients or spatial coordinates, which can be 2D or 3D. The training process is, therefore, the modification of model parameters

such that the model describes a set of observation sequences, i.e. the training set, as good as possible. This is achieved by maximizing the likelihood of the training set, which usually involves an expectation maximization (EM) procedure like the Baum–Welch algorithm [56]. Such training procedures involve a step based on dynamic programming that in turn has a dynamic time warping (DTW) property, which accounts for the variability in duration of the training samples. Basically this is achieved by hidden states allowing transitions to themselves.

In the recognition process, a given trajectory in the model parameter space of the gestures is tested over the set of trained HMMs in the system. The model that gives the highest likelihood is taken to be the gesture class the trajectory belongs to. The likelihoods are calculated using the Viterbi algorithm.

Possible drawbacks of HMMS are:

- Markov property may not hold for every system,
- higher order HMMs are not as computationally efficient,
- the observation functions are linear,
- HMMs are homogeneous, i.e. the observation and state transition probabilities do not vary in time.

Several HMM variants attack these problems:

- Coupled HMMs
- Asynchronous HMMs
- Input–Output HMMs

Input–Output HMM, which is proposed by Bengio *et al.* in [57],a neural network HMM hybrid, attacks the problems of HMM by representing the local models, i.e. the hidden

states with possibly nonlinear and flexible complex structures such as Multi Layer Perceptrons (MLP).

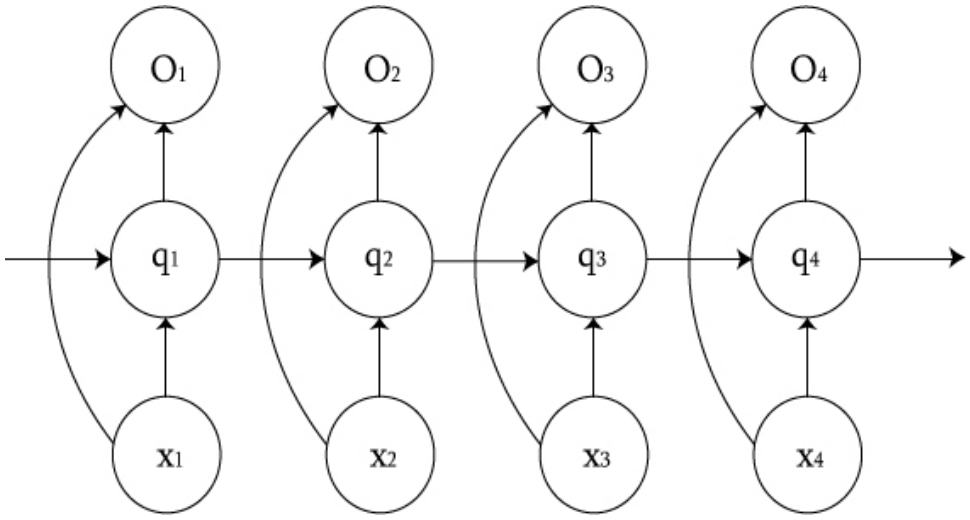


Figure 1.9. IOHMM architecture

IOHMMs condition the stochastic architecture of HMMs on an input sequence that has no independence assumptions. Hence the architecture can be conditioned on any function of observable or system parameters. Therefore the stochastic network parameters of an IOHMM is not homogeneous. Due to their inhomogeneity, IOHMMs are expected to learn long time dependencies better than HMMs. Despite their promising characteristics, IOHMMs have not yet drawn much attention. They are used in [58] by Just *et al.* to distinguish deictic and symbolic hand gestures.

Artificial Neural Networks (ANN) are used by many researchers to model gestures [29, 33, 59]. Fels *et al.* use ANNs for their "Glove–Talk" interface, which maps input coming from a data glove to speech synthesizer commands. Recurrent Neural Networks (RNN) are also used for the gesture recognition task [60].

Other models used for gesture classification include Time–Delay Neural Networks (TDNN) that are temporal extensions to ANNs, which are also commonly used to learn mappings between training sets and gestures [29, 61]. The major problem of TDNNs

is that they cannot learn the long time dependencies as well as HMM variants.

1.4.1. Gesture Spotting

In certain HCI applications an important phase of the recognition problem is spotting the meaningful gestures. An application may want to filter out the unintentional movements, distinguish manipulative and communicative gestures, or it may not restrict the possible hand gesture and try to spot gestures from continuous input images, that conform to any of the gesture models in the system [8]. This is a complex task, unless users are required to perform specific gestures that indicate start and end of gestures, which affects the naturalness of the HCI application. The method proposed by Lee and Kim in [54] is an effective method that deals with gesture spotting problem for HMM–based frameworks.

Lee and Kim propose a method for online gesture spotting using HMMs that can distinguish meaningless gestures from defined ones. Their approach proposes creation of an ergodic HMM from the hidden states of the defined gestures, which is called an adaptive threshold model. This model will consist of the hidden states of the defined gestures and hence, give a high likelihood for every gesture that has similarities to any of the defined gestures in subsequence level. Nevertheless the dedicated model is expected to give higher likelihood than the threshold model, since it does not make use of the temporal information. Figure 1.10 shows how gesture spotting detects a circular gesture from a continuous input stream, where the blue and red dots indicate the raw and filtered hand coordinates respectively.

1.5. Proposed Approach

In the proposed system continuous recognition of 3D hand gestures that involve both trajectory and posture information in real time is attempted using a stereo vision–

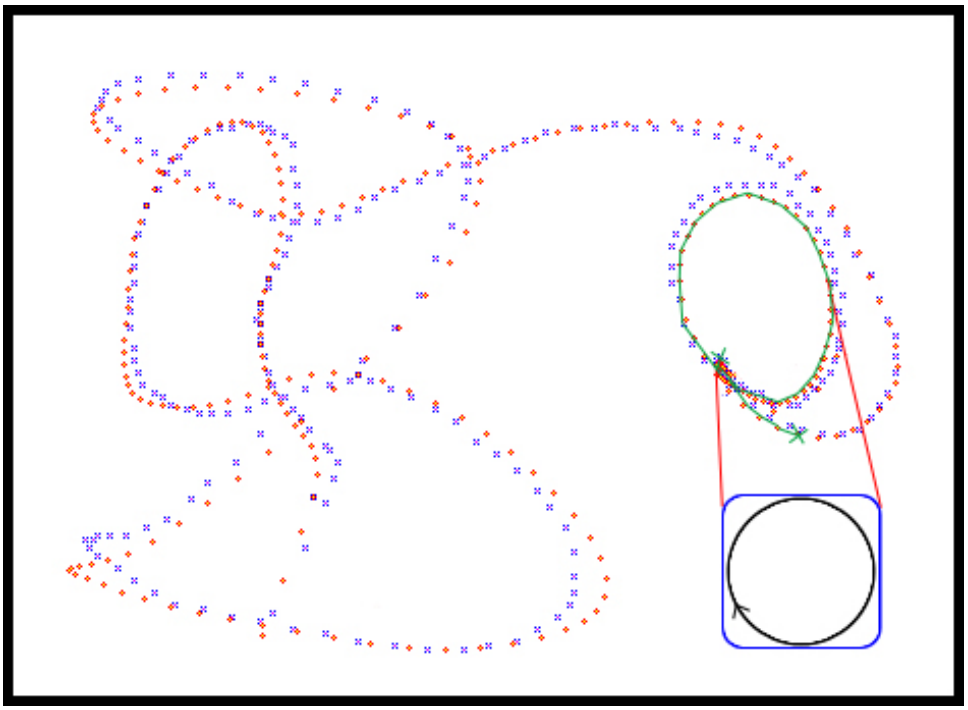


Figure 1.10. Gesture spotting

based approach on an IOHMM–based framework. The system is developed as an interface for generic desktop applications, where the hand motion replaces the mouse using manipulative gestures, and communicative gestures can be trained to the system that can be used to control target applications.

The system makes use of two simple cameras to retrieve stereo images of the scene. Color based methods are researched to detect and localize the hand in the input images. A uniquely colored glove is used to distinguish the hand and extract the hand image. The marker registration phase uses motion cues to detect the marker color and color cues based on this color are used to detect and localize the hand.

The centroid of the hand is found in each view and two 2D trajectories are constructed from the estimated centroid coordinates. Kalman filter is used to smooth the trajectories and filter the noise. The filtered trajectories and the camera calibration matrices are used to reconstruct the 3D hand trajectory using a least squares approach.

Camera calibration is dealt with by using a special calibration object, which automatizes the calibration process using a least squares approach on multiple correspondence pairs. The reconstructed 3D trajectory is smoothed using a 3D Kalman filter.

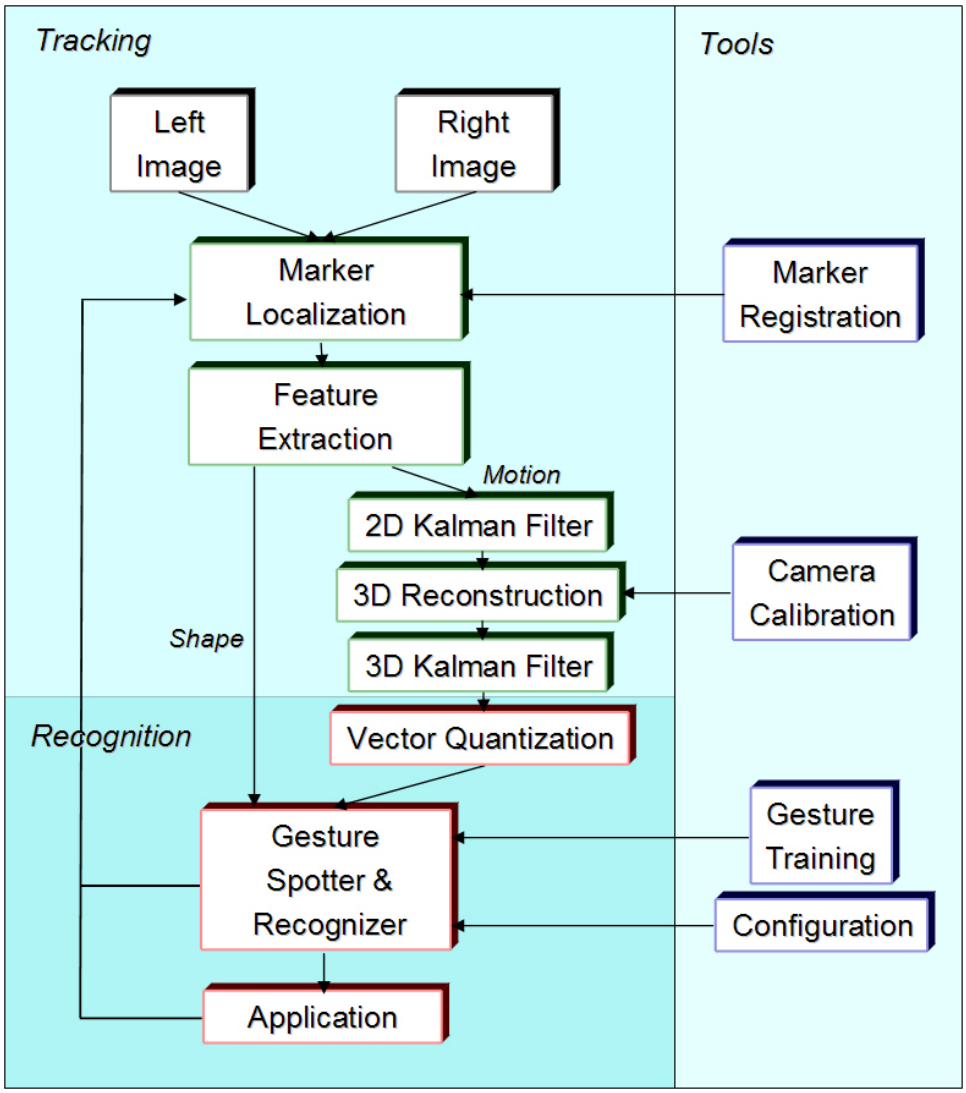


Figure 1.11. Flowchart of the system

The extracted hand features in each image are the 3D hand motion parameters, the scale and rotation invariant Hu moments of the hand image and the estimated angle of the hand. These observations are collected in two separate sequences. Motion related parameters form a sequence of velocity vectors that are quantized according to

a partitioning of the 3D space. The magnitude of the velocity is discarded to ensure scale invariability in the recognition phase. Hu moments and the angle form another sequence, whose mapping to the velocity symbol sequence is learned by an IOHMM–based framework, i.e. the sequence of Hu moments and the angle is taken to be the input sequence of the IOHMMs where the velocity symbols form the output sequence.

IOHMMs employ MLPs to map the input variables to the Bayesian network probabilities, which requires the GEM algorithm to train the system. A dataset of gestures is created, where gestures are allowed to share the same trajectories to test the success the system attains at distinguishing different hand postures.

A novel method to distinguish the communicative gestures in an IOHMM–based framework is proposed and tested, which allows the system to continuously handle input streams without requiring the user to indicate the start and end points of gestures.

The CPU time spent by each phase is logged to detect bottlenecks of the system and to optimize for speed to conform to the real time requirements. Currently the system is able to attain 93.8% success in recognition of 20 gestures in the presence of the threshold model, while being able to run at 30 fps from each camera.

1.6. Outline of the Thesis

In Chapter 1 hand gesture recognition and its use in HCI applications are introduced, followed by a review of methods used in in hand pose estimation, object tracking and gesture classification. The approach that will be used to attack the real time hand gesture recognition problem concludes this chapter.

Chapter 2 covers the evaluation and training methods used for gesture recognition in an HMM–based framework and their extension to IOHMMs. In Chapter 3, the

methodology of the proposed system is explained in detail, where brief discussions on the choice of methods and models are given. Chapter 4 covers the tests conducted to measure the gesture recognition and hand tracking computation time performance of the proposed system, and presents the results. Chapter 5 concludes the thesis with conclusions, discussion of the results and future work.

2. GESTURE RECOGNITION USING HIDDEN MARKOV MODELS

2.1. Gesture Recognition using an HMM–Based Framework

As reviewed in Chapter 1, HMMs treat gestures as sequences generated by a parametric random process with Markov property. Therefore HMMs assume that, given all present and past events, the conditional probability density of the current event depends only on the most recent k events, where k is the order of the Markov process and is usually taken as $k = 1$ for computational reasons. This means that the process is assumed to be a first order Markov process in which the current event depends only on the most recent past event. This chapter will discuss modeling of gestures with HMMs and learning of the parameters from training sets.

HMMs can be viewed as probabilistic finite state automatons, where the states are not observable but each state S_i generates observations according to a probabilistic function of the state. It is assumed that neither the observation nor the state transition function depends on time and therefore HMMs are called homogeneous models. Each state has a probability of making a transition to itself or other states, which is determined by the topology of the HMM. The fact that the states are allowed to make transitions to themselves gives HMMs a high degree of time-scale invariance, which is a desirable aspect of the gesture recognition process.

The transition probability from state i to state j under the assumption of first order Markov property can be given as:

$$a_{ij} \equiv P(q_{t+1} = S_j | q_t = S_i) \ , \quad i, j = 1, \ldots, N \tag{2.1}$$

where q_t is the state variable, N is the number of distinct states and S_i and S_j are states that are not necessarily distinct. The values a_{ij} should satisfy

$$a_{ij} \geq 0 \quad \forall i, j \tag{2.2}$$

and

$$\sum_{j=1}^{N} a_{ij} = 1 \tag{2.3}$$

These values can be thought of as forming an $N \times N$ matrix A with rows that sum up to one.

Even though the matrix A explains the transitions on the state model, it does not explain how the first state is chosen. Therefore initial probability π_i for each state i should be defined, which is the probability that the state sequence starts with S_i.

$$\pi_i \equiv P(q_1 = S_i) \ , \quad i = 1, \ldots, N \tag{2.4}$$

Similarly, the values π_i can be thought of as forming a vector $\mathbf{\Pi}$ that sum up to one.

$$\sum_{i=1}^{N} \pi_i = 1 \tag{2.5}$$

The observations are allowed to be discrete or real valued, which will be taken as distinct in this chapter for simplicity. The proposed system also makes use of distinct observation symbols. In the case of a system with M distinct observable symbols from the set $\{v_1, v_2, \ldots, v_M\}$ the probability of observing the symbol v_m in state j is given as:

$$b_j(m) \equiv P(O_t = v_m | q_t = S_j) \tag{2.6}$$

where O_t is the observation sequence. Similarly an $N \times M$ matrix B can be defined that consists of the elements $b_j(m)$ that satisfies the normalization condition:

$$\sum_{m=1}^{M} b_j(m) = 1 \tag{2.7}$$

The parameter set of an HMM can defined as $\Theta = (A, B, \mathbf{\Pi})$. For gesture recognition there are two main problems of HMM, namely the recognition and training problems. Recognition, or the evaluation problem is the estimation of likelihood of a sequence for a given HMM, whereas training is estimation of these parameters from a set of sequences.

2.1.1. Recognition

Given an observation sequence O and a model Θ, evaluation is calculating the probability that the observed sequence was generated by the model, which can be denoted as $P(O|\Theta)$. Recognition is performed by choosing the model with the highest probability after evaluating the probability value for all competing models.

If the state sequence Q was known, the calculation of $P(O|\Theta)$ would be simply the product of the probabilities of observing each O_t in the corresponding state q_t. Since the states are hidden, likelihood needs to be calculated by summing up over all possible state sequences.

$$P(O|\Theta) = \sum_{all\ \ possible\ \ Q} \pi_{q_1} \left[\prod_{t=2}^{T} a_{q_{t-1},q_t} \right] \left[\prod_{t=1}^{T} b_{q_t}(O_t) \right] \tag{2.8}$$

where T is the length of the sequence. Assuming an ergodic topology, the number of all possible Q is N^T and therefore this method is not tractable for large N and T. A recursive method called the forward-backward procedure is used to efficiently estimate

the likelihood. The forward variable, denoted $\alpha_t(i)$ is defined as follows:

$$\alpha_t(i) \equiv P(O_1, \ldots, O_t, q_t = S_i|\Theta) \tag{2.9}$$

which is the probability of observing the partial sequence $\{O_1, \ldots, O_t\}$ and being in state S_i at time t. $\alpha_t(i)$ can be calculated using recursion as follows:

$$\alpha_1(i) = \pi_i b_i(O_1) \tag{2.10}$$

$$\alpha_{t+1}(i) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}) \tag{2.11}$$

$\alpha_T(f)$ is the probability of observing the entire observation sequence and ending in the state S_f. The HMM topology can be restricted such that the gestures are allowed to end only in certain states, which are called the final states. In that case the summation of $\alpha_T(f)$ over all final states gives the likelihood of a sequence for a model with parameters Θ, which is the desired value of the evaluation problem.

$$P(O|\Theta) = \sum_{f} \alpha_T(f) \ , \quad \forall f \in F \tag{2.12}$$

where F is the set of final states. For a given observation sequence O, this value needs to be calculated and compared to the likelihoods given by other models for recognition. This calculation is $O(N^2T)$ for each model.

Similar to α a backward variable, denoted β, is used to explain the partial observation sequence that starts at t.

$$\beta_t(i) \equiv P(O_{t+1}, \ldots, O_T|q_t = S_i, \Theta) \tag{2.13}$$

which is the probability of observing the partial sequence $\{O_{t+1}, \ldots, O_T\}$ and being in

S_i at time t. β can also be computed recursively as follows:

$$\beta_T(i) = 1 \tag{2.14}$$

$$\beta_{t-1}(i) = \sum_{j=1}^{N} a_{ij} b_j(O_t) \beta_t(j) \tag{2.15}$$

β is not used for recognition, but will be useful for the training of HMMs as explained in the next section.

Note that this recognition method does not distinguish meaningful and meaningless gestures and simply selects the model with the highest likelihood. Numerically the likelihood is affected by the sequence length and a static threshold cannot be used to distinguish the meaningful gestures. The methods to solve this problem have been reviewed in Chapter 1, and the implemented algorithm is explained in Chapter 3.

2.1.2. Training

In order to be able to recognize gestures the model parameters for each HMM needs to be known beforehand. The parameter set Θ of an HMM can be learned using maximum likelihood approach, by calculating Θ^* that maximizes the likelihood $P(X|\Theta)$ of a training set X. The approach to be explained here is an Expectation Maximization (EM) procedure called the Baum-Welch algorithm, which needs two new variables, $\gamma_t(i)$ and $\xi_t(i,j)$ to be defined. EM procedure iteratively calls two steps, namely the E-step and the M-step, where $\gamma_t(i)$ and $\xi_t(i,j)$ are calculated using the current Θ in the E-step, and Θ is recalculated from these variables in the M-step until the likelihood converges.

The variable $\gamma_t(i)$ is defined as the probability of being in S_i at time t, given an observation sequence O and the model parameter set Θ, which can easily be computed

using the forward and backward variables as follows:

$$\gamma_t(i) = P(q_t = S_i|O,\Theta) \tag{2.16}$$
$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N}\alpha_t(i)\beta_t(i)} \tag{2.17}$$

Similarly, the variable $\xi_t(i,j)$ is defined as the probability of being in S_i at time t and being in S_j at time $t+1$, given O and Θ. $\xi_t(i,j)$ can be computed using the following equation:

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j|O,\Theta) \tag{2.18}$$
$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_k\sum_l \alpha_t(k)a_{kl}b_l(O_{t+1})\beta_{t+1}(l)} \tag{2.19}$$

In the case of a single training sample the elements of A, B and $\mathbf{\Pi}$ are reestimated in the M-step using the following equations:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)} \tag{2.20}$$
$$\hat{b}_j(m) = \frac{\sum_{t=1}^{T}\gamma_t(j)1(O_t = v_m)}{\sum_{t=1}^{T}\gamma_t(j)} \tag{2.21}$$
$$\hat{\pi}_i = \gamma_1(i) \tag{2.22}$$

In the case of a training set with P samples the equations become:

$$\hat{a}_{ij} = \frac{\sum_{p=1}^{P}\sum_{t=1}^{T_p-1}\xi_t^p(i,j)}{\sum_{p=1}^{P}\sum_{t=1}^{T_p-1}\gamma_t^p(i)} \tag{2.23}$$
$$\hat{b}_j(m) = \frac{\sum_{p=1}^{P}\sum_{t=1}^{T_p}\gamma_t^p(j)1(O_t^p = v_m)}{\sum_{p=1}^{P}\sum_{t=1}^{T_p}\gamma_t^p(j)} \tag{2.24}$$
$$\hat{\pi}_i = \frac{\sum_{p=1}^{P}\gamma_1^p(i)}{P} \tag{2.25}$$

The derivation of these equations can be found in several machine learning textbooks

and survey papers.

2.2. Gesture Recognition using an IOHMM–Based Framework

IOHMM is a variant of HMM that conditions each state transition and emission probability on an input sequence. Therefore the static parameters of HMM become dependent on time, and hence IOHMMs are inhomogeneous. The state transition probabilities become:

$$a_{ij}(t) = P(q_t = j | q_{t-1} = S_i, x_t) \tag{2.26}$$

where x_t is the input at time t. x_t can be a scalar or a vector of parameters. Note that this probability is conditioned on x_{t+1}. This means that in a time frame the new state depends on the current input and the state from the last time frame. Similarly the emission probabilities become:

$$b_{j,t}(m) = P(O_t = v_m | q_t = S_j, x_t) \tag{2.27}$$

In this case A and B matrices are dependent on time and therefore denoted as $A(t)$ and $B(t)$. $\mathbf{\Pi}$ vector can also be dependent on the input, even though in some applications they are independent from the input as in HMMs. In the case of dynamic $\mathbf{\Pi}$ the elements become:

$$\pi_i = P(q_1 = S_i | x_1) \tag{2.28}$$

If $\mathbf{\Pi}$ is not conditioned on the input, x_1 is only used for calculating the emission probability of the first state.

Local models, i.e. models that generate the state transition and emission proba-

bilities can be described by any complex model, in particular by Multi Layer Perceptrons (MLP), which are flexible and nonlinear. Input variables have no independence assumptions and therefore they can be used to incorporate a wider context of inputs without violating the Markov property. This means that they can use functions of previous observations as input, which enables them to learn long time dependencies better than HMMs.

Once the dynamic state transition and emission probabilities are calculated for a time frame using the input and state information, IOHMMs behave exactly like HMMs. The equations related to the recognition and training problems will be given in the following sections.

2.2.1. Recognition

As in the case of HMMs the recognition problem is calculating the likelihood of a sequence given model parameters Θ, this time conditioned on the input sequence x^t. This likelihood, denoted as $P(O|x^t\Theta)$, can be calculated in a similar manner:

$$P(O|\Theta) = \sum_{all\ possible\ Q} \pi_{q_1} \left[\prod_{t=2}^{T} a_{q_{t-1},q_t}(t)\right] \left[\prod_{t=1}^{T} b_{q_t,t}(O_t)\right] \tag{2.29}$$

The forward variable defined in the previous section can be used to compute this likelihood efficiently in a similar fashion:

$$\alpha_t(i) \equiv P(O_1, \ldots, O_t, q_t = S_i|x_1^t, \Theta) \tag{2.30}$$

which is the probability of observing the partial sequence $\{O_1, \ldots, O_t\}$, being in state S_i at time t and observing the input sequence x_1^t until time t. $\alpha_t(i)$ can be calculated

using the same recursion:

$$\alpha_1(i) = \pi_i b_{i,1}(O_1) \tag{2.31}$$

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}(t+1)\right] b_{j,t+1}(O_{t+1}) \tag{2.32}$$

The likelihood of a sequence for a model with parameters Θ and the input sequence x^t can be computed by summing up $\alpha_T(i)$ over the final states as before.

$$P(O|\Theta) = \sum_f \alpha_T(f) \ , \quad \forall f \in F \tag{2.33}$$

where F is the set of final states. This calculation is $O(N^2T)$ times the time complexity of generating a probability using a local model.

The backward variable $\beta_t(i)$ is defined in a similar manner, this time using variables that are conditioned on the input:

$$\beta_t(i) \equiv P(O_{t+1}, \ldots, O_T | q_t = S_i, x_t^T, \Theta) \tag{2.34}$$

which is the probability of observing the partial sequence $\{O_{t+1}, \ldots, O_T\}$, being in S_i at time t and observing the input sequence x_t^T. $\beta_t(i)$ is computed as follows:

$$\beta_T(i) = 1 \tag{2.35}$$

$$\beta_{t-1}(i) = \sum_{j=1}^{N} a_{ij}(t) b_{j,t}(O_t) \beta_t(j) \tag{2.36}$$

$\beta_t(i)$ is not needed for the recognition phase and will be used in the training phase.

2.2.2. Training

As in the case of HMMs, the parameter set Θ of an IOHMM can be learned using maximum likelihood approach, by calculating Θ^* that maximizes the likelihood $P(X|\Theta)$ of a training set X. In particular the E-step to train IOHMMs is the same as the E-step of the EM algorithm used for HMMs, but the M-step may differ significantly due to the local models implemented. In general for complex models the maximization process has no analytical solution and other methods are used to maximize the likelihood. In the case of MLPs as local models, the maximization process becomes the gradient ascent process that iteratively increases the likelihood by backpropagation algorithm. This version of the EM procedure is called the Generalized Expectation Maximization (GEM) procedure.

As in the case of Baum-Welch algorithm, variables $\gamma_t(i)$ and $\xi_t(i,j)$ need to be defined. In order to avoid confusion these variables will be renamed to $g_t(i)$ and $h_t(i,j)$ respectively. Therefore $g_t(i)$, the state posterior probability, is defined as the probability of being in S_i at time t, given an observation sequence O, the input sequence x^t and the model parameter set Θ, which can easily be computed using the forward and backward variables as follows:

$$g_t(i) = P(q_t = S_i|O, x^t, \Theta) \tag{2.37}$$

$$g_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(i)\beta_t(i)} \tag{2.38}$$

The variable $h_t(i,j)$, the transition posterior probability, is defined as the probability of being in S_i at time $t-1$ and being in S_j at time t, given O, x^t and Θ. $h_t(i,j)$ can be computed using the following equation:

$$h_t(i,j) = P(q_{t-1} = S_i, q_t = S_j|O, x^t, \Theta) \tag{2.39}$$

$$h_t(i,j) = \frac{\alpha_{t-1}(i)a_{ij}(t)b_{j,t}(O_t)\beta_t(j)}{\sum_k \sum_l \alpha_{t-1}(k)a_{kl}(t)b_{l,t}(O_t)\beta_t(l)} \tag{2.40}$$

Note the subtle difference between the definitions of $\xi_t(i,j)$ and $h_t(i,j)$. $\xi_t(i,j)$ is not defined for $t = T$, whereas $h_t(i,j)$ is not defined for $t = 1$.

In the case of discrete input a multinomial model can be used to generate the probabilities, in which case EM algorithm can be used in a straightforward manner by replacing each probability in the HMM calculations by their conditional versions. The local models become simple lookup tables in that case and the implementation becomes straightforward. This case will not be explained here in detail, since quantizing the input variables is not a desired aspect of the proposed system.

Local models can be chosen as linear ANNs, in which case there is an analytical solution to the maximization problem and once again the EM procedure can be used. Due to the complexity of sign gesture recognition problem one cannot assume linearity of local models and therefore MLPs are better suited as local models. Accordingly the training process for IOHMMs with MLPs as local models will be discussed here.

In order to apply the EM procedure the state sequence Q needs to be estimated at each step in order to calculate the likelihood. If there is an estimation of Q, the likelihood of a given dataset X can be written as:

$$\begin{aligned} L_c(\Theta; X_c) \;=\; & \prod_{p=1}^{P} \left[\prod_{i=1}^{N} P(q_1 = S_i | x_1, \Theta)^{z_{i,1}} \right] \times \\ & \left[\prod_{t=1}^{T} \prod_{i=1}^{N} P(O_t | q_t = S_i, x_t, \Theta)^{z_{i,t}} \right] \times \\ & \left[\prod_{t=2}^{T} \prod_{i=1}^{N} \prod_{j=1}^{N} P(q_t = S_j | q_{t-1} = S_i, x_t, \Theta)^{z_{j,t} z_{i,t-1}} \right] \end{aligned} \tag{2.41}$$

where X_c is the complete dataset with known Q, and $z_{i,t}$ is a variable that takes the value one if $q_t = S_i$ and zero otherwise. To simplify the equations the log–likelihood

can be derived:

$$\begin{aligned}\log L_c(\Theta; X_c) &= \sum_{p=1}^{P}\sum_{i=1}^{N} z_{i,1} \log P(q_1 = S_i | x_1, \Theta) + \\ &\sum_{p=1}^{P}\sum_{t=1}^{T}\sum_{i=1}^{N} z_{i,t} \log P(O_t | q_t = S_i, x_t, \Theta) + \\ &\sum_{p=1}^{P}\sum_{t=2}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N} z_{j,t} z_{i,t-1} \log P(q_t = S_j | q_{t-1} = S_i, x_t, \Theta) \quad (2.42)\end{aligned}$$

Since Q is not known beforehand this likelihood needs to be approximated by integrating over all possible Q. The approximated log–likelihood can be calculated by estimating the expected value of $\log L_c(\Theta; X_c)$ given the current value of Q:

$$\Omega(\Theta, \hat{\Theta}) = E_Q \left[\log L_c(\Theta; X_c) | O, x^t, \hat{\Theta} \right] \quad (2.43)$$

$$\begin{aligned}&= \sum_{p=1}^{P}\sum_{i=1}^{N} E_Q \left[z_{i,1} | O, x^t, \hat{\Theta} \right] \log P(q_1 = S_i | x_1, \Theta) \quad + \\ &\sum_{p=1}^{P}\sum_{t=1}^{T}\sum_{i=1}^{N} E_Q \left[z_{i,t} | O, x^t, \hat{\Theta} \right] \log P(O_1 | q_t = S_i, x_t, \Theta) \quad + \\ &\sum_{p=1}^{P}\sum_{t=2}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N} E_Q \left[z_{j,t} z_{i,t-1} | O, x^t, \hat{\Theta} \right] \log P(q_t = S_j | q_{t-1} = S_i, x_t, \Theta) \quad (2.44)\end{aligned}$$

where

$$E_Q \left[z_{i,1} | O, x^t, \hat{\Theta} \right] = \hat{g}_{i,1} \quad (2.45)$$

$$E_Q \left[z_{i,t} | O, x^t, \hat{\Theta} \right] = \hat{g}_{i,t} \quad (2.46)$$

$$E_Q \left[z_{j,t} z_{i,t-1} | O, x^t, \hat{\Theta} \right] = \hat{h}_t(i, j) \quad (2.47)$$

Using the posterior probabilities $\Omega(\Theta, \hat{\Theta})$ can be written as:

$$\begin{aligned}\Omega(\Theta, \hat{\Theta}) = &\sum_{p=1}^{P}\sum_{i=1}^{N} \hat{g}_{i,1} \log P(q_1 = S_i | x_1, \Theta) \quad + \\ &\sum_{p=1}^{P}\sum_{t=1}^{T}\sum_{i=1}^{N} \hat{g}_{i,t} \log P(O_t | q_t = S_i, x_t, \Theta) \quad + \\ &\sum_{p=1}^{P}\sum_{t=2}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N} \hat{h}_t(i,j) \log P(q_t = S_j | q_{t-1} = S_i, x_t, \Theta)\end{aligned} \tag{2.48}$$

$\Omega(\Theta, \hat{\Theta})$ is the function to be maximized in the M-step. Intuitively the new parameter set Θ is calculated from the current set $\hat{\Theta}$.

Updating the parameters requires the $\Omega(\Theta, \hat{\Theta})$ function to be written in terms of the outputs of the MLPs. Then the network weights can be updated with the backpropagation procedure using gradient ascent. Usually the system can be thought of as starting from a certain state without loss of generality, which drops the first term in the M-step and simplifies the equation. Specifically if the first term is dropped and outputs of the networks of state i at time t are denoted $\eta_{im,t}$ for the emission networks, and $\varphi_{ij,t}$ for the state transition networks, $\Omega(\Theta, \hat{\Theta})$ becomes:

$$\Omega(\Theta, \hat{\Theta}) = \sum_{p=1}^{P} \left(\sum_{t=1}^{T}\sum_{i=1}^{N} \hat{g}_{i,t} \log \eta_{iO_t,t} + \sum_{t=2}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N} \hat{h}_t(i,j) \log \varphi_{ij,t} \right) \tag{2.49}$$

The update rules are derived by differentiating this function with respect to the specific weights of the MLPs. In gradient ascent procedure the update is in the same direction as the derivative and a generic weight update rule can be written as:

$$\rho_{jk} = \rho_{jk} + \mu \frac{\partial \Omega(\Theta, \hat{\Theta})}{\partial \rho_{jk}} \ , \ \forall j, k \tag{2.50}$$

where ρ_{jk} is a generic weight and μ is the learning factor. The specific update rules for

the proposed system are derived in Chapter 3.

3. METHODOLOGY

This chapter explains the methodology used for each task of the 3D continuous hand gesture recognition problem. The entire problem can be separated in two different sets of subtasks that are associated with either hand tracking or hand gesture recognition. Each specific task and the corresponding algorithm used will be explained in the following sections in detail.

3.1. Hand Tracking

In this section the subtasks related to the hand tracking problem will be defined and the methodology used in the proposed system will be explained and discussed.

The first subtask of the hand tracking phase is the detection of the hand, which is common to all vision based hand gesture recognition systems. Certain features of the detected hand shape also have to be extracted by the systems that aim to recognize hand postures as well as hand trajectories.



Figure 3.1. Hand tracking

Due to the ambiguity of the projections of the 3D hand trajectories on 2D image coordinates, a hand gesture recognition system has to make use of the 3D information in order to recognize the entire set of possible hand trajectories. Therefore the 3D reconstruction of the 3D hand coordinates is also an important subtask of the hand tracking problem.

The image retrieval process is often noisy in vision based systems either due to measurement, which naturally takes place due to the discretization of the acquired images, or due to the performance of the tracking algorithm. The elimination of the noise is more complicated for ergodic real–time systems. The noise can significantly reduce the performance of the hand gesture recognition module and is a crucial step in the hand tracking phase.

The hand tracking problem consists of the following subtasks:

- Hand Detection
- Hand Feature Extraction
- Filtering
- 3D Reconstruction

The rest of this section explains and discusses the algorithms used for each of these tasks.

3.1.1. Hand Detection

The purpose of the hand detection phase is to find the location of the hand in a given image. Hand detection is a search problem and relies on certain decision criteria such as the skin color, which is the natural choice to distinguish the hand from the background. However, it should also distinguish the hand from other skin colored parts

of the body, when it cannot be assumed that no such part exist in the image sequence. In a system that works with cameras pointed to the user, the skin colored face is always in front of the camera and such an assumption is always wrong. Such a system either has to make use of a hand model and fit the model to the hand in each image, or it should assume another color for the hand, which means that the user needs to wear a colored glove that separates the hand from the background. The possible approaches have been reviewed in Chapter 1.

Even though model based approaches are better suited for applications that aim to distinguish the hand from the face, the characteristic problem of such algorithms is that they are unsuitably slow for real–time applications. It is possible to accelerate these algorithms at the expense of robustness, which will reduce the hand gesture recognition rate accordingly and is therefore inappropriate.

The appearance based methods make use of color information. The natural choice of color in this case would be the skin color, but it fails to distinguish the hand from the face and is therefore not suitable. A solution that solves both the speed and the distinction problem is using a marker. Therefore markers are used in the system to detect the hand in each image in the image sequences.

A marker that can be used to distinguish the hand is simply a colored glove that has a solid color, chosen such that it can distinguish the hand from the background. The choice of color is not predefined, but is restricted to colors that are significantly different than the skin color or gray, since grayscale colors are not suitable for hue based search methods.

Once the marker color is chosen it needs to be identified by the system. This is called the “Marker Registration” task and has to be the first step each time the system is started. In this phase the system learns the parameters of the chosen marker color.

The choice of parameters and the registration algorithm are discussed in the following section.

3.1.1.1. Marker Registration. The marker registration task is estimating certain parameters of the marker color for identification. The color perceived by the cameras is a function of the light reflected by the hand and therefore the color space used in identification of the marker should not be dependent on the lighting conditions. Even though it is possible to use any color space for this task, the parameters to be estimated have different complexities for different color spaces. The proposed system identifies the markers using the HSI color space, since the brightness is an uncorrelated component of this color system. This allows the system to discard the brightness component and estimate the parameters using the other components, namely the hue and the saturation, and the parameters can be chosen to be the mean and variance in this case.

The hue is defined as the color tone and is not defined for grayscale colors. Therefore the distribution of hue on the color space of the image pixels, namely the RGB space, is discontinuous. Because of the discontinuity the pixel colors close to grayscale have unexpected hue values. Consequently using the HSI color space imposes requirements on the possible marker colors, and the chosen marker color cannot be grayscale.

Every time the application is started the marker color needs to be registered. To guide the system on where to look, the user executes the marker registration tool and moves their hand. In order to estimate the parameters the following method is used:

- Capture two consecutive images
- Compute the difference image of the two acquired images
- Apply thresholding on the difference image
- Generate the image of significantly different pixels using the thresholded difference image

- Compute the hue and saturation histograms of the image
- Find the bins with maximum number of pixels
- Estimate the mean and variance of the hue and saturation values corresponding to these bins

Due to the high sampling rate, the captured images are expected to be considerably similar in the first step of the algorithm, except for the regions where the hand is moving. Hence, in the difference image the pixels with the highest Euclidean distance to their former values will be likelier to correspond to the marker. Thresholding is necessary at this point because of either flickering or the automatic image enhancement algorithms of some cameras, which can considerably change how the scene looks between two consecutive frames. The thresholded difference image is used in the next step to find the actual pixels that correspond to the regions of highest difference. Figure 3.2 shows effects of several choices for the value of the threshold. The value 10% for the threshold has been experimentally found to be suitable for detecting the marker.

The estimated regions of highest difference are expected to contain mostly the pixels corresponding to the hand if the hand has moved quickly enough. Nevertheless Figure 3.2 reveals that this is not always the case, even for optimal values of the threshold, because significantly different regions correspond to both the former and current places of the marker. Some of the pixels in these regions can be eliminated using the aforementioned constraint of the HSI system. The marker color, if chosen according to the HSI criteria, should be considerably distinct from grayscale colors. Therefore as a secondary criterion the pixels that are close to black, white and gray can be eliminated. Figure 3.3 shows the successful results of employing the secondary criteria for a relatively harder case of the problem. In these figures the regions marked with red are pixels that are found to be too bright, the regions marked with blue are pixels that are too close to gray, and the black regions are the pixels corresponding to either regions of insignificant change or to regions too dark, all of which are eliminated



(a) Threshold = 1

(b) Threshold = 5

(c) Threshold = 10

(d) Threshold = 25

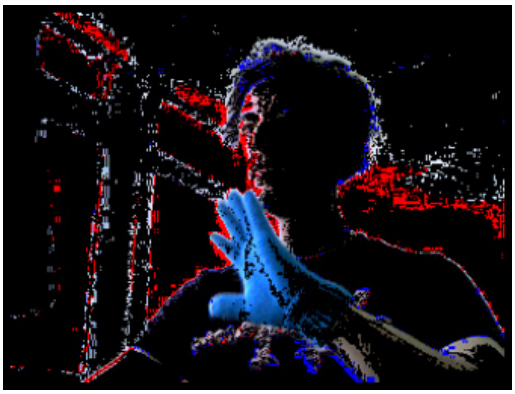
Figure 3.2. Significantly different regions estimated using different thresholds

from the final difference image.

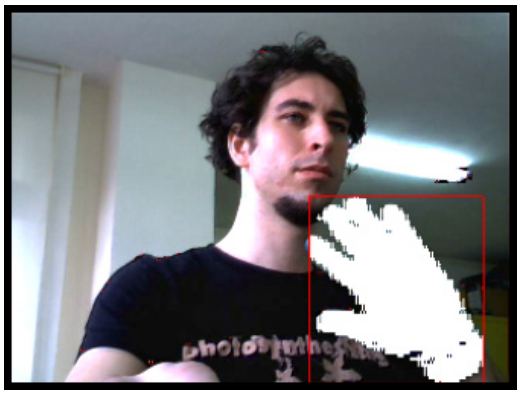
Addition of the secondary criteria increases the likelihood that the remaining pixels on the difference image correspond to the marker. There are a number of ways to determine the parameters of interest from this image. A basic idea is calculating the hue and saturation histograms and taking the bins with the maximum number of pixels. Figure 3.4 shows the two modes of the RGB histogram of a difference image, where one of the modes correspond to the marker and the other to the background. The assumption the proposed system relies on is that the mode with more pixels corresponds to the marker. To estimate which of the modes has more pixels, the hue



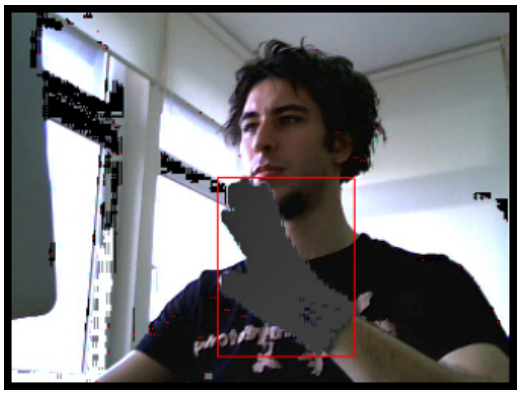
(a) Difference image for the left camera



(b) Difference image for the right camera



(c) Estimated marker for the left image



(d) Estimated marker for the right image

Figure 3.3. Effect of secondary criteria in marker registration

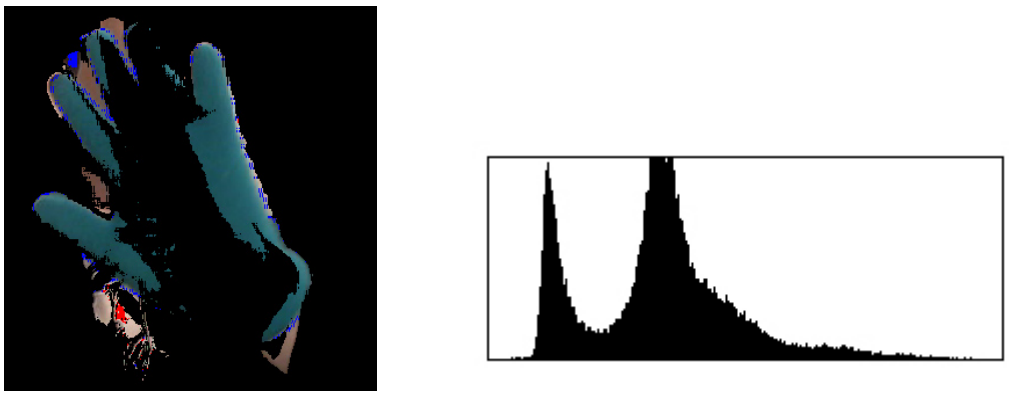
and saturation histograms use large bins, in order for the bins with maximum number of pixels to be likelier to correspond to the marker. The mean and variance of the pixels of these bins are estimations of the mean and variance of the hue and saturation values of the marker color. Figure 3.5 reveals that the mode that corresponds to the larger body of pixels correspond to the marker indeed. The definitions and the equations are as follows:

- $\mu_{hue/sat}$ = The mean of the hue or saturation component
- $\sigma^2_{hue/sat}$ = The variance of the hue or saturation component

- $S_{hue/sat}$ = The bin of either component with the maximum number of pixels
- $P_i^{hue/sat}$ = The hue or saturation value of the pixel i
- $N_{S_{hue/sat}}$ = The number or pixels in the bin

$$\mu_{hue/sat} = \frac{\sum_{i \in S_{hue/sat}} P_i^{hue/sat}}{N_{S_{hue/sat}}} \tag{3.1}$$

$$\sigma^2_{hue/sat} = \frac{\sum_{i \in S_{hue/sat}} (P_i^{hue/sat} - \mu_{hue/sat})^2}{N_{S_{hue/sat}} - 1} \tag{3.2}$$



(a) Hand image (b) Intensity histogram

Figure 3.4. RGB histogram of a final difference image

However there are a number of problems with this approach:

- The bins might be too small to contain all the marker pixels
- The bin boundaries might not be aligned with the exact range of the marker pixels

In both cases the marker pixels will fall into different bins and the estimated mean and variance values will not be precise. Nevertheless they will be close enough to allow

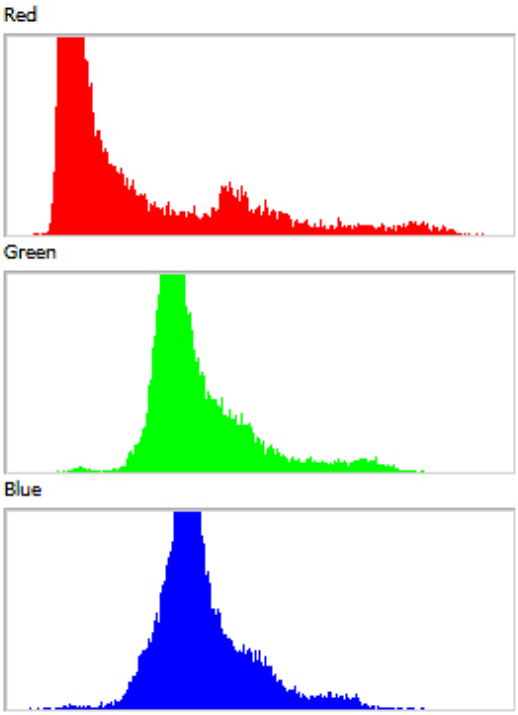


Figure 3.5. Red, green and blue components of the RGB histogram

employment of an adaptive algorithm later on that correctly estimates these values.

Another approach to this problem is using K-Means Clustering, a simple technique that would better estimate the means and variances of color clusters, but the system would need to estimate the number of modes in the histogram first. This is a more natural approach, but it is left as a future work due to the success rate of the currently employed algorithm, which adaptively recalculates the mean and variance from the found marker pixels in the hand tracking phase as explained in the next section.

3.1.2. Hand Region Extraction

The detection of the hand does not conclude the tracking problem. If the system does not define a certain single point on the hand for tracking, the algorithm will introduce unreasonable amounts of noise to the system because of jumps between

frames. The selection of a point to track should aim to minimize this error. The following points have been tested with the proposed system:

- The finger tip
- The centroid
- The center of the bounding box

The most robust feature to track has been found to be the center of the bounding box, since it changes slower than the other points and therefore generates less noise.

In addition to the point to be tracked, a hand gesture or hand posture recognition system needs to extract certain features from the hand images that describe the hand. If the system makes use of the hand shape information, it has to be able to correctly classify most of the marker pixels. Both false positive and false negative classifications considerably change the hand shape and affect the hand features, which in turn reduce the recognition success rate. Therefore the classification algorithm must be robust to lighting conditions and marker parameter estimation errors due to the marker registration algorithm.

The hand feature extraction phase consists of finding the hand in the image and extracting its shape and features, which are discussed in the following sections.

3.1.2.1. Finding The Hand. The marker parameters estimated in the marker registration phase are used to detect the location of the hand in a given image. This is a pixelwise search for a certain range of the marker color in the image. To accelerate this process the search window is set as a square centered on the previous location of the hand, because the hand is not expected to be too far away from its previous location in the current frame, given the high sampling rate. The window size is determined using

the image resolution and the sampling rate using the following equation:

$$WindowSize = \frac{2 * ImageWidth}{SamplingRate} \tag{3.3}$$

If the hand cannot be found in the search window, another search is conducted on the entire image.

The first point to be classified as a marker pixel will be the start point of the region growing algorithm that will estimate the shape of the hand. Therefore this first point is required to be sufficiently similar to the estimated marker color. The rest of the pixels to be connected are not required to be very close to the mean of the color, since it is believed that the first point is already on the marker. This is ensured by defining two ranges of colors based on the mean and variance of the marker color:

$$Th_{low} = \sigma \qquad R_{low} = [\mu - Th_{low}, \mu + Th_{low}] \tag{3.4}$$

$$Th_{high} = 3\sigma \qquad R_{high} = [\mu - Th_{high}, \mu + Th_{high}] \tag{3.5}$$

R_{low} is the range of hue or saturation values used to ensure that the first pixel classified as the marker pixel is a pixel on the marker with high probability, whereas R_{high} is used to connect most of the marker pixels. This algorithm is very similar to Canny Edge Linking, which is a double thresholding method, where the low threshold is used to ensure that the connection starts at a pixel that should certainly be included, and the high threshold makes sure the gradient range of the similar pixel values are included in the connected region. Figure 3.6 shows a case where both of the thresholds are set to $\sigma/2$. As seen the algorithms fails to connect the pixels of the marker due to too low thresholds.

This algorithm, while classifying most of the hand pixels correctly, fails to correctly classify some of the marker pixels either due to motion blur or due to the shadows on the hand. These cases are shown in Figure 3.7. In order to make sure that all the similarly colored pixels in the neighborhood are included, dynamic thresholding method is used:

- Find the initial point using R_{low}
- Apply region growing using R_{high}
- Whenever a candidate pixel is not in R_{high}, temporarily increase Th_{high} by 10% and check if the pixel is in the new range
- If the pixel is classified as a marker pixel this time, update Th_{high} using its new value
- If the pixel still cannot be included, discard the temporary value of Th_{high}
- When the algorithm concludes re-estimate the mean and variance of the hue and saturation using all the pixels labeled as marker pixels

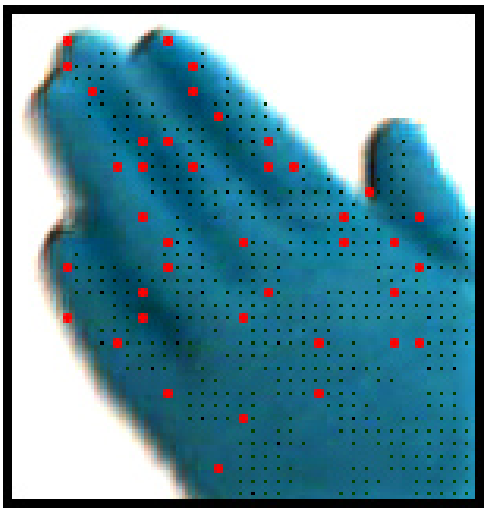


Figure 3.6. Effect of using very low thresholds for both R_{low} and R_{high}

This method has been thoroughly tested and has been found to give satisfactory results. Figure 3.8 shows several successful extractions of the hand image. The blue low resolution images are the binary maps indicating the hand shape, which are used later on for further feature extraction.



(a) Boundary pixels not connected

(b) Holes in the connected region

Figure 3.7. Problems with double thresholding using static thresholds

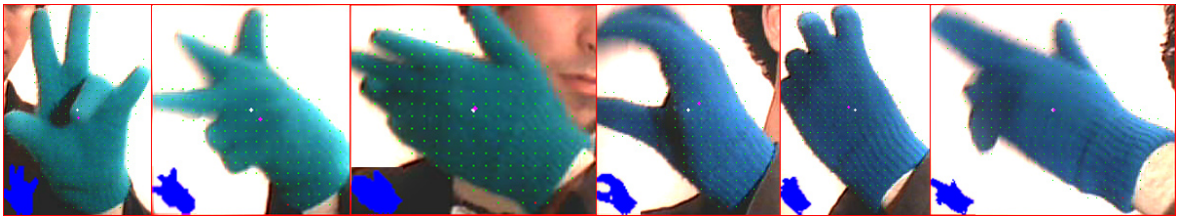


Figure 3.8. Results of using double thresholding with dynamic thresholds

This algorithm is called once for each frame that has the marker in the search window, and twice for each frame that has the marker elsewhere. Therefore another important issue to consider is the speed of the algorithm, which directly affects the frame rate of the application.

The proposed system aims to run at 30 fps for each camera, which means that 60 frames need to be processed each second. Without considering the significant amount of time the recognition phase requires, this leaves 16.67 ms for each frame to be processed. For high resolutions the region growing algorithm takes 8 ms on the average, which

becomes unacceptable when the recognition times are considered. Therefore instead of connecting every neighboring pixel, a step size based on the resolution is defined, such that only every n_{th} pixel is connected. This simple update causes n^2 times speedup at the expense of the extracted hand image resolution. Rather than being a problem, this trade off automatically solves the downsampling of the hand image problem, which is a required step to decrease the complexity of the feature extraction phase. For a step size of 8, the required time is reduced to values lower than 0.2 ms, which leaves enough time for recognition of the hand gesture.

3.1.2.2. Extracting Hand Features. The hand shape information extracted from the hand images will directly be fed to the neural networks of the IOHMMs in the gesture recognition phase. Therefore there are a number of requirements imposed on the selected hand features. Namely, the features should not be affected by the translation or size of the hand, and they should be as uncorrelated as possible to minimize the input size of the neural networks. The moments are a good candidate, since they are scale and rotation invariant. The image moments for a grayscale image $I(x, y)$ are calculated as follows:

$$M_{ij} = \sum_{x} \sum_{y} x^i y^j I(x, y) \tag{3.6}$$

The moments are useful in describing certain properties of segmented objects, such as the area and the centroid:

$$Area = M_{00} \tag{3.7}$$

$$Centroid \quad \bar{x} = M_{10}/M_{00} \ , \quad \bar{y} = M_{01}/M_{00} \tag{3.8}$$

From this set of features a new set of moments about the mean can be defined. These new centralized features are called the central moments:

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q I(x, y) \tag{3.9}$$

The proposed system makes use of the central moments of order up to 3, which can be defined as follows in terms of M_{ij}:

$$\begin{aligned}
\mu_{00} &= M_{00} && (3.10)\\
\mu_{01} &= 0 && (3.11)\\
\mu_{10} &= 0 && (3.12)\\
\mu_{11} &= M_{11} - \bar{x}M_{01} = M_{11} - \bar{y}M_{10} && (3.13)\\
\mu_{20} &= M_{20} - \bar{x}M_{10} && (3.14)\\
\mu_{02} &= M_{02} - \bar{y}M_{01} && (3.15)\\
\mu_{21} &= M_{21} - \bar{x}M_{11} - \bar{y}M_{20} + 2\bar{x}^2 M_{01} && (3.16)\\
\mu_{12} &= M_{12} - \bar{y}M_{11} - \bar{x}M_{02} + 2\bar{y}^2 M_{10} && (3.17)\\
\mu_{30} &= M_{30} - 3\bar{x}M_{20} + 2\bar{x}^2 M_{10} && (3.18)\\
\mu_{03} &= M_{03} - 3\bar{y}M_{02} + 2\bar{y}^2 M_{01} && (3.19)
\end{aligned}$$

The central moments can be used to estimate the angle of the image, which is one of the important features a gesture recognition system should make use of. This angle is given by the following equation, which is the angle of the eigenvector, that corresponds to the highest eigenvalue of the covariance matrix:

$$\theta = \frac{1}{2} \tan^{-1} \left(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}} \right) \tag{3.20}$$

In order to generate a scale invariant feature set, the central moments need to be

normalized using the area, which is M_{00}

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{\gamma}} \tag{3.21}$$

where $\gamma = (i+j)/2 + 1$ for $(i+j) \geq 2$.

It is possible to calculate moments which are invariant under translation, rotation and scaling of objects. Most frequently used are the Hu set of invariant moments:

$$I_1 = \eta_{20} + \eta_{02} \tag{3.22}$$

$$I_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{3.23}$$

$$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{3.24}$$

$$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{3.25}$$

$$\begin{aligned} I_5 = {} & (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + 3\eta_{03})^2] \\ & + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \tag{3.26}$$

$$I_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2 + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})] \tag{3.27}$$

$$\begin{aligned} I_7 = {} & (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ & + (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \tag{3.28}$$

The last one of Hu moments, I_7, is skew invariant, which enables it to distinguish mirror images.

The extracted hand features not only consist of the Hu moments. Even though having a set of rotation invariant set of features reduces the correlation of the parameters, a gesture recognition system requires the angle information to be present. Therefore the set of hand shape features used by the proposed system are the Hu moments and the angle for each view of the hand. For recognition of 3D gestures the features extracted from both of the views are supplied to the neural networks of

IOHMMs, since two different views of the hand can better explain the 3D shape information of the hand. Therefore the number of input nodes of the neural networks reserved for the hand shape features are:

$$N_{2D} = 8 \tag{3.29}$$

$$N_{3D} = 16 \tag{3.30}$$

It should be noted that these are not required to be the only inputs of the neural networks. The entire set of inputs the neural networks have are discussed in the following sections.

3.1.3. Filtering the Hand Trajectory

At the end of the hand detection phase the system is supplied with the current coordinates of the hand for each camera, which are concatanated in each frame to form the 2D hand trajectories. These trajectories are subject to noise that might significantly affect the recognition rates. Therefore a real–time filtering method needs to be employed, which requires the dynamics of the motion model of the hand to be known.

It is common for vision systems with a high sampling rate to assume linear motion between frames. Under this assumption the hand motion dynamics can be described by a Linear Dynamical System (LDS). An LDS is a partially observable stochastic process with linear dynamics and linear observations. In practice, the individual state variables of an LDS cannot be determined exactly by direct measurements, due to the fact that the measurements are functions of the state variables and these measurements are affected by random noise. The system itself may also be subject to random disturbances. It is then required to estimate the state variables from the noisy observations.

Such a system can be described with the following equations:

$$\vec{X}(t+1) = F\vec{X}(t) + \vec{w}(t) \quad , \quad \vec{\omega} \sim N(0, Q') \tag{3.31}$$

$$\vec{Y}(t) = H\vec{X}(t) + \vec{v}(t) \quad , \quad \vec{v} \sim N(0, R') \tag{3.32}$$

where $\vec{X}(t)$ is the hidden state at time t, $\vec{Y}(t)$ is the observation, $\vec{\omega}$ is the system noise, $\vec{v}$ is the measurement noise, F is the state transition matrix, H is the observation matrix, and Q and R are the covariance matrices of the Gaussian distributions, which approximate the system and measurement noise distributions. The Kalman Filter is applicable to systems that can be described as such [52].

Under the assumption of high and constant sampling rate it is possible to view the proposed system as an LDS described by the following simple equations:

$$\begin{bmatrix} X_1(t) \\ X_2(t) \\ V_{X_1}(t) \\ V_{X_2}(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_1(t-1) \\ X_2(t-1) \\ V_{X_1}(t-1) \\ V_{X_2}(t-1) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ w_3 \\ w_4 \end{bmatrix} \tag{3.33}$$

where $\vec{Y}$ is the 2D hand coordinate vector, $\vec{V}$ is the velocity vector of the hand and $\vec{X}$ is the hidden state vector that consists of the observation $\vec{Y}$ and the velocity $\vec{V}$. The observation process can be described by the following equation:

$$\begin{bmatrix} Y_1(t) \\ Y_2(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} X_1(t) \\ X_2(t) \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \tag{3.34}$$

For a symmetric system $v_1 = v_2 = v$ can be set. The system noise is proportional to the image resolution and should be estimated from a long observation of the system. Given the simplicity of the linear equations, experimentally selecting the system noise model allows quick estimation of acceptable parameters. It is also assumed that the

system noise does not favor any single direction. Therefore $w_3 = w_4 = w$, and the covariance matrices for the Gaussian noise distributions become:

$$Q' = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & w & 0 \\ 0 & 0 & 0 & w \end{bmatrix} \tag{3.35}$$

$$R' = \begin{bmatrix} v & 0 \\ 0 & v \end{bmatrix} \tag{3.36}$$

Note that $w_1 = w_2 = 0$ was already assumed, which means that the system noise affects the velocity.

The filtering operation makes use of both the past and the future of a given frame. Therefore the Kalman Filter uses two sets of equations to predict the value of a state variable, namely the Time Update equations and the Measurement Update equations. Given the current state $\vec{X}$ and the current observation $\vec{Y}$:

Time Update Equations:

$$\begin{aligned} \vec{X}_{prior} &= F\vec{X} \\ P_{prior} &= FP'F^T + Q' \end{aligned}$$

Measurement Update Equations:

$$\begin{aligned} K &= P_{prior}H^T(HP_{prior}H^T + R)^{-1} \\ \vec{X}_{posterior} &= \vec{X_{prior}} + K(Y - (H\vec{X}_{prior})) \\ P' &= (I - KH)P_{prior}; \end{aligned}$$

where

$$\begin{aligned}
\vec{X}_{prior} &= Prior\ State\ Estimate \\
\vec{X}_{posterior} &= Posterior\ State\ Estimate \\
\vec{Y} &= StateMeasurement \\
F &= State\ Transition\ Matrix \\
H &= Observation\ Matrix \\
Q' &= System\ Error\ Covariance \\
R &= Measurement\ Error\ Covariance \\
P' &= Prior\ Error\ Covariance \\
K &= Kalman\ Gain
\end{aligned}$$

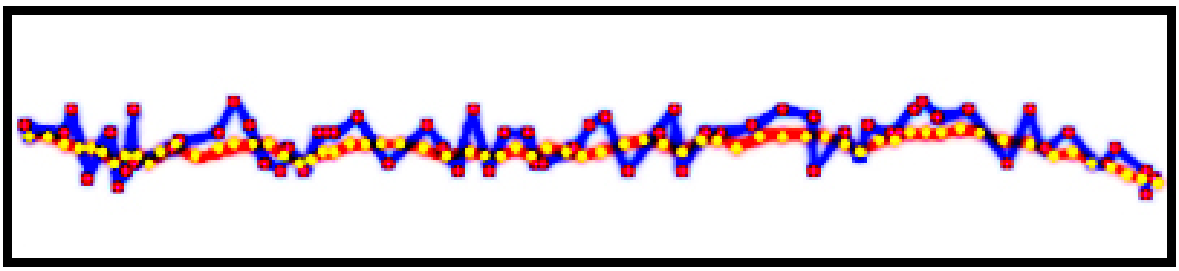


Figure 3.9. Result of filtering the hand trajectory

Figure 3.9 shows the result of filtering of the 2D trajectory of the hand. The blue line is the unfiltered trajectory and is achieved by running the system with suboptimal hand detection parameters. The red line is the filtered trajectory that is used in 3D reconstruction or 2D gesture recognition.

The 3D hand trajectory is calculated from the 3D reconstruction of filtered 2D trajectories. The 3D reconstruction process is not very robust to noise and the error generated or not eliminated in the Kalman Filtering phase directly affect the smoothness of the 3D trajectory. Therefore, for 3D gestures the 3D hand trajectory should also be filtered. The extension of this method to 3D is straightforward.

3.1.4. Estimating the 3D Hand Trajectory

The proposed system aims to recognize 3D hand gestures and therefore it should be able to reconstruct the 3D coordinates of the hand from the 2D image coordinates found and filtered in the previous phases. This 3D reconstruction process requires knowledge of the scene and the cameras.

The acquisition of the images is mapping of the 3D coordinates of the scene to a 2D plane. The parameters involved in such a mapping are the translation, rotation and scaling, namely the projection parameters, along with the intrinsic parameters of the camera, which cause effects such as radial distortion and decentering. If the intrinsic parameters are known, a metric measurement can be performed from the images, which is not required for the purposes of a hand gesture recognition system.

The 3D to 2D mapping can be constructed as a linear operation:

$$P\vec{X} = \vec{x} \tag{3.37}$$

$$\begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha x \\ \alpha y \\ \alpha \end{bmatrix} \tag{3.38}$$

where $\vec{X}$ is the space coordinate vector, $\vec{x}$ is the image coordinate vector, α is the scaling factor and P is a 3×4 matrix that is referred to as the "Calibration Matrix". For 3D stereo reconstruction this matrix should be known for each camera. Estimation of P is known as the calibration process. Note that P is not unique, since the target space has smaller dimensionality than the source space and there are an unlimited number of matrices that can map the given 3D coordinates to the 2D coordinates on the image. One way to overcome this problem is normalization by setting the last

element of the matrix to unity. This normalization will be used in deriving the method for finding the elements of the calibration matrix in the following section.

3.1.4.1. Camera Calibration. The camera calibration method used by the proposed system starts with the normalization of the last parameter of the calibration matrix to unity, thereby reducing the number of parameters to be estimated to 11. Solving for α in the Equation 3.38 results in the following equations:

$$\begin{aligned} x(XP_{31} + YP_{32} + ZP_{33} + 1) &= XP_{11} + YP_{12} + ZP_{13} + P_{14} \\ y(XP_{31} + YP_{32} + ZP_{33} + 1) &= XP_{21} + YP_{22} + ZP_{23} + P_{24} \end{aligned}$$

After some algebraic manipulations it is possible to convert these equations to matrix notation, where the elements of the calibration matrix create a vector:

$$\begin{bmatrix} X & Y & Z & 1 & 0 & 0 & 0 & 0 & -xX & -xY & -xZ \\ 0 & 0 & 0 & 0 & X & Y & Z & 1 & -yX & -yY & -yZ \\ & & & & & & & \vdots & & & \end{bmatrix} \begin{bmatrix} P_{11} \\ P_{12} \\ P_{13} \\ P_{14} \\ P_{21} \\ P_{22} \\ P_{23} \\ P_{24} \\ P_{31} \\ P_{32} \\ P_{33} \end{bmatrix} = \begin{bmatrix} x \\ y \\ \vdots \end{bmatrix} \tag{3.39}$$

which can be denoted as:

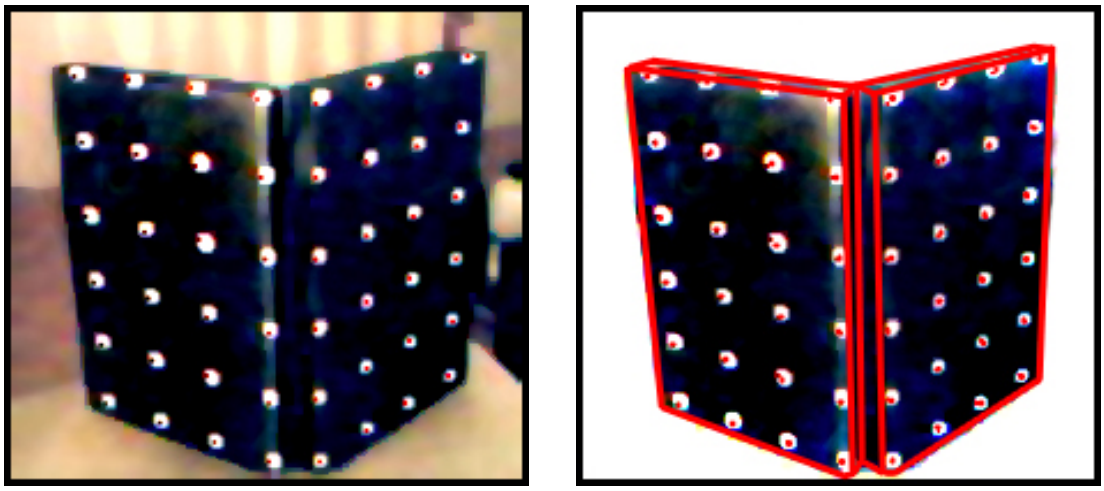
$$E\vec{p} = \vec{d} \tag{3.40}$$

After normalization, there are eleven unknowns left, which need to be solved from this linear set of equations. Since each 3D–2D point pair yields two equations, theoretically at least six points are needed to estimate the eleven unknowns. This process is well known for not being robust to noise and therefore usually more than six point pairs are used to estimate the parameters. These point pairs yield more equations than the number of unknowns present at the system and the least squares estimation approach is used in estimating the parameters. The least squares solution can be formulated in terms of the pseudo inverse of E, denoted $E^{\dagger}$, as:

$$\vec{p} = E^{\dagger}\vec{d} \qquad E^{\dagger} = (E^{T}E)^{-1}E^{T} \tag{3.41}$$

The 3D reconstruction process numerically relies on the accuracy of the calibration matrix. In practice, as many as possible point pairs are used in the estimation problem to increase the accuracy. A common method to retrieve multiple points is using an object with features that have known 3D coordinates, which is called a calibration object. This is usually an object with several corners that are easy to find like a checkerboard, or an object with markers that indicate the features points. For a feasible solution the feature points on the calibration object should not be coplanar. Therefore planar calibration objects like the checkerboard must be moved during the calibration process. The proposed system includes a tool to automatically estimate the calibration matrices using a specific object that does not require being moved. This object is shown in Figure 3.10(a).

In the calibration process the system finds the white points on the object and classifies them by assuming vertical clusters. It is then possible to label each point in the clusters vertically and to establish the correspondences between the 3D coordinates and the image coordinates of the points. The calibration matrix is estimated by applying the least squares method to the point pairs resulting from the automated process.



(a) Calibration object (b) Calibration points

Figure 3.10. The special object used for the calibration process

3.1.4.2. Stereo Reconstruction. The image acquisition process maps 3D coordinates of object points to 2D coordinates on the image plane. The 3D reconstruction operation aims to reverse this process using the calibration matrices. Given the the calibration matrices and the 2D coordinates on the image, linear equations can be derived. The unknowns are the object coordinates $\vec{X}$. Hence there are two equations for three unknowns, meaning that only a line passing through the actual 3D point instead of the point itself can be found using a single point. If there are multiple images of the same scene, the intersection point of the lines passing through the same points can be used to estimate the actual 3D coordinates. In the case of two cameras the number of equations is four for three unknowns. The set of equations is as follows:

$$\begin{aligned}
x_1(XP^1_{31} + YP^1_{32} + ZP^1_{33} + 1) &= XP^1_{11} + YP^1_{12} + ZP^1_{13} + P^1_{14} \\
y_1(XP^1_{31} + YP^1_{32} + ZP^1_{33} + 1) &= XP^1_{21} + YP^1_{22} + ZP^1_{23} + P^1_{24} \\
x_2(XP^2_{31} + YP^2_{32} + ZP^2_{33} + 1) &= XP^2_{11} + YP^2_{12} + ZP^2_{13} + P^2_{14} \\
y_2(XP^2_{31} + YP^2_{32} + ZP^2_{33} + 1) &= XP^2_{21} + YP^2_{22} + ZP^2_{23} + P^2_{24}
\end{aligned}$$

where P^1 and P^2 are the calibration matrices of the cameras, (x_1, y_1) and (x_2, y_2) are the 2D image coordinates, and (X, Y, Z) is the 3D coordinates of the point. Manipulating these equations algebraically yields the following operation in matrix notation:

$$\begin{bmatrix} P^1_{11} - x_1 P^1_{31} & P^1_{12} - x_1 P^1_{32} & P^1_{13} - x_1 P^1_{33} \\ P^1_{21} - y_1 P^1_{31} & P^1_{22} - y_1 P^1_{32} & P^1_{23} - y_1 P^1_{33} \\ P^2_{11} - x_2 P^2_{31} & P^2_{12} - x_2 P^2_{32} & P^2_{13} - x_2 P^2_{33} \\ P^2_{21} - y_2 P^2_{31} & P^2_{22} - y_2 P^2_{32} & P^2_{23} - y_2 P^2_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} x_1 - P^1_{14} \\ y_1 - P^1_{24} \\ x_2 - P^2_{14} \\ y_2 - P^2_{24} \end{bmatrix} \tag{3.42}$$

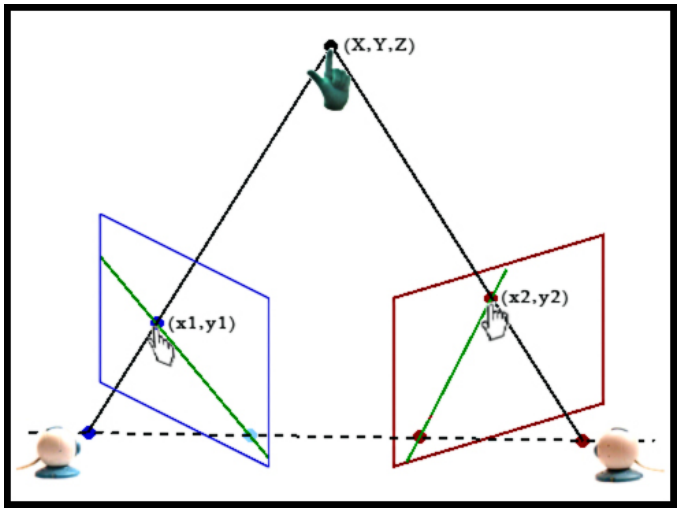


Figure 3.11. Camera setup and 3D reconstruction

Again there are more equations than unknowns, and as in the calibration method the least squares solution is formulated in terms of the pseudo inverse of the first matrix, which is denoted as E'. The solution vector contains the 3D coordinates of the point:

$$\vec{p} = E'^{\dagger}\vec{X} \qquad E'^{\dagger} = (E'TE')^{-1}E'^{T} \tag{3.43}$$

Because of the noise and the sampling error present in the images, 3D reconstruction

may yield considerably noisy results. Even though filtering the 2D trajectories help smoothing of the 3D trajectory, when the predictions of 2D Kalman Filter fail, the effect on the 3D trajectory is usually unacceptable. The 3D Kalman Filter solves this problem very effectively and the filtered 3D trajectory is used for the recognition phase of the proposed system.

3.2. Hand Gesture Recognition

The hand tracking phase of the system continuously keeps track of the location and shape of the hand and estimates the filtered 3D hand trajectory, along with a sequence of hand shape features, which are to be used in the gesture recognition phase. The location and motion information extracted from the hand coordinates are used for gesture recognition, whereas the hand shape information is used to estimate the hand posture. A hand gesture recognition system should make use of both the gesture and posture of the hand.

The proposed system assumes that the same gesture can vary in location, size or speed. Therefore, the recognition of a gesture should not be dependent on:

- the exact location the user performs the gesture
- the overall speed the gesture is performed
- the size of the gesture performed

Each of the assumptions prohibit certain sets of gestures to coexist in the system. The first assumption ensures that a certain gesture is recognized regardless of the relative location of the hand with respect to any point on the scene or the body of the user. Therefore, the system does not allow definition of two gestures that differ only in their locations. This might be undesired for sign language recognition systems, where the meanings of the gestures might differ according to this information, but it is usually a

desirable aspect for HCI applications.

The second assumption ensures that recognition of a gesture is not dependent on the speed it is performed. Therefore the system cannot differentiate gestures that differ only in their speed. Accordingly such gestures are not allowed by the proposed system. Note that this assumption does not implicate that the gestures are not allowed to differ in the speeds of their subsequences. Specifically how the time is distributed on different parts of the hand gesture might be a differentiating feature of two hand gestures.

The third assumption allows users to perform the same gestures without considering the size of the gestures, and ensures that the gesture recognition phase is not dependent on the distance between the user and the camera. Even though this assumption restricts the gestures by not allowing size to be a differentiating factor, this does not cause a loss of generality, since the HCI system can be designed to allow parametric hand gestures. In that case if a hand gesture is successfully recognized it can wait for the input of user, which can be given with a secondary hand motion that need not be a predefined gesture. An example is a hand gesture used to turn up the volume for a target media player application. When the hand gesture is recognized the system can wait for the user to indicate how much the volume should be turned up or down with a hand motion, where the size of the motion can be taken as the parameter.

The observation sequence that is supplied by the hand tracking phase consists of the coordinates of the hand, which can be normalized to satisfy the given assumptions. However this method is prone to error, since any outlier caused by the noise in the system can drastically change the normalized coordinates of the specific points on the gesture. Therefore velocity vectors are used instead, which also satisfy the assumptions except for the size independence. To ensure size independent modeling of a gesture either the coordinates should be normalized beforehand, or the velocity vectors need

to be quantized, which entirely discards the magnitude of the motion. The main difference between these approaches is that quantization generates symbolic observation, whereas the velocity vectors of normalized coordinates have continuous values. Note that the normalization process still suffers from outliers, reducing the robustness of the recognition process. Also the model that describes the gesture is more complex for the continuous observation vectors. Therefore the magnitude of the velocity vectors is not reliable and should be discarded. In both cases the angle of the velocity vectors are used.

In the proposed system, models do not yet support continuous observations. Allowing continuous observations is a straightforward extension to the system. This is left as future work, since gestures of any complexity can be described with quantized velocity vectors by increasing the number of quantization levels.

In Chapter 1, several methods to model and recognize the gestures have been mentioned, such as Dynamic Time Warping (DTW), Artificial Neural Networks (ANN), Hidden Markov Models (HMM) and Input–Output Hidden Markov Models (IOHMM). The studies on DWT and ANN show that the former lacks robust performance with respect to shape variations while problems arise during modeling dynamic patterns with the latter. Therefore the use of HMM models has become a common method in gesture recognition. IOHMM is a variant of HMM that is capable of conditioning the recognition phase on an input sequence, which is a promising method to describe the relationship between the hand gesture and posture information. The proposed system uses HMMs to recognize hand trajectories and IOHMMs to recognize hand gestures. These methods and the vector quantization process are explained in detail in the following sections.

3.2.1. Vector Quantization

Calculating the velocity vectors and discarding the magnitude information is equivalent to estimating the spherical coordinates of the 3D hand motion between the frames and discarding the radius. To quantize the estimated angles the 3D space is segmented in a preset number of partitions. These partitions need not be distributed homogeneously over the 3D space. For certain applications it can be assumed that some of the partitions are rarely used and such partitions can be merged with their neighbors. A system with a single camera is such an example, since only 2D hand trajectories can be realized and the depth component does not need to be partitioned.

The proposed system has preset quantization levels that have been tested and found to be acceptable. If the target system has a single camera, the vector quantization algorithm quantifies the velocity vector using the eight main compass directions on the image plane. If the system can perceive the depth as well, the union of the eight main compass directions on each of the three scene planes are taken as the quantization levels, generating 18 distinct symbols when the overlapping directions are discarded. Four of these partitions have been found to be rarely used and eliminated to reduce the complexity.

Note that there is no symbol signifying no motion, since the velocity vector always falls into one of the partitions as long as the magnitude is not zero, which is only possible when the hand is found in exactly the same coordinate in consecutive frames. Even if the hand is not moved between the frames, there might be an estimated motion due to the measurement noise. This suggests that the velocity vector should be quantized only if it has a greater magnitude than a preset threshold and it should be classified as “No Motion” otherwise. This threshold is called the “Radius of No Motion” and is dependent on the resolution, the calibration matrices and the sampling rate. This radius $R_{3D}(t)$ is calculated on each frame as follows:

- Set $R_{2D}(t) = \frac{Image\ Width}{7*FrameRate}$
- Set $p_1 = (x_1, y_1)$, where (x_1, y_1) is the hand coordinate on the first image
- Set $p_2 = (x_2, y_2)$, where (x_2, y_2) is the hand coordinate on the second image
- Set $p'_1 = (x_1 + R_{2D}(t), y_1 + R_{2D}(t))$
- Set $p'_2 = (x_2 - R_{2D}(t), y_2 - R_{2D}(t))$
- Set P_1 to be the 3D hand coordinate reconstructed from p_1 and p_2
- Calculate P_2 by reconstructing the 3D hand coordinate from p'_1 and p'_2
- Calculate the Euclidean distance $D = \|P_1 - P_2\|$
- Set $R_{3D}(t) = D$

In the case of execution with a single camera $R_{2D}(t)$ is used as the threshold at each frame to decide if a motion is significant. Otherwise $R_{2D}(t)$ is used to determine the maximum distance the 2D hand locations can have in two consecutive frames, where the hand is kept still or slightly moved. The most distant points are used to reconstruct a 3D hand coordinate. The distance of this 3D hand coordinate to the actual 3D hand coordinate is the result of only noise and insignificant movements of the hand. Therefore, the distance between these two locations is used as the threshold to determine whether a 3D hand motion is significant.

With the inclusion of the class corresponding to no motion, the total number of preset quantization symbols becomes one5, as listed in Table 3.1.

3.2.2. HMM–Based Hand Gesture Recognizer

The structure and mechanism of HMMs have been thoroughly discussed in Chapter 2. In this section, the specific topology and implementation of the HMM based framework will be explained in detail.

In the proposed system, the HMM implementation aims to recognize hand tra-

Class	Symbol	Direction
0	C	No Motion
1	U	Up
2	D	Down
3	R	Right
4	L	Left
5	F	Forward
6	B	Backward
7	UR	Up Right
8	UL	Up Left
9	UF	Up Forward
10	UB	Up Backward
11	DR	Down Right
12	DL	Down Left
13	DF	Down Forward
14	DB	Down Backward

Table 3.1. Vector quantization classes

jectories without the posture information and therefore allows training and recognition of only pure hand motions. The observation sequence is the quantized velocity vectors that describe the hand trajectory and the models are trained using the EM algorithm, as explained in Chapter 2.

In this framework the hidden states correspond to the memory of the model. A hidden state determines the observation probability of each symbol and the state transition probability to other states. The observation and state transition probabilities are static for a hidden state. Therefore, intuitively the hidden states should correspond to the partitions or edges of a gesture, where the probability of observing certain symbols is nearly constant. For instance while moving the hand on an edge of a square

shaped gesture, the probability of observing the symbol corresponding to the direction the user traverses the edge is nearly unity, while other observation probabilities are close to zero.

If a state covers more than one edge, the information related to the order of two edges is lost. Consider the case where a state corresponds to two consecutive edges of the square shaped gesture. Observation probability of both of the symbols corresponding to the directions of the edges are nearly equal in this case, but the model does not know which of the edges come first. In particular this model will give high likelihoods to any gesture that has these two symbols in any order in the subsequence corresponding to the two edges. Therefore, in order to accurately model the gesture the number of states an HMM has should be nearly equal to the number of distinct edges or partitions of the gesture. The estimation of required number of hidden states is done automatically by the proposed system using a state merging method explained in the next section.

Another important observation is that at the end of an edge the probability of transition to the state corresponding to the next edge is close to unity, while the other probabilities are zero, except for the state itself. This lets the topology of the HMM networks to be constrained such that only transition to the same or the next state is allowed for each state. An HMM having this topology is called a Left–Right HMM.

In addition to the common requirements of an HMM based gesture recognition system, our system should also handle continuous streams of data. Therefore the gestures need to be spotted first. This problem could easily be solved by defining a certain motion or posture that triggered and ended gesture recognition. However, this is not natural and is very tiresome when the system is used for continuous manipulation of an application. Therefore our system solves this problem by employing an adaptive threshold model.

3.2.2.1. Left–Right HMMs. The Left–Right HMMs have a special topology that only allows self–transitions and transitions to the next state. This allows numbering of states in a specific order, where a state S_i can only make a transition to state S_{i+1}. Figure 3.12 shows the topology of a Left Right HMM with four states. This changes the Π and A matrices of the HMM considerably and simplifies the training and evaluation algorithms given in Chapter 2:

$$\Pi = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.44}$$

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & \cdots & 0 \\ 0 & a_{22} & a_{23} & 0 & \cdots & 0 \\ 0 & 0 & a_{33} & a_{34} & \cdots & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & \cdots & 0 & a_{n-1n-1} & a_{n-1n} \\ 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \tag{3.45}$$

For the evaluation problem, the equations to calculate the forward variable α

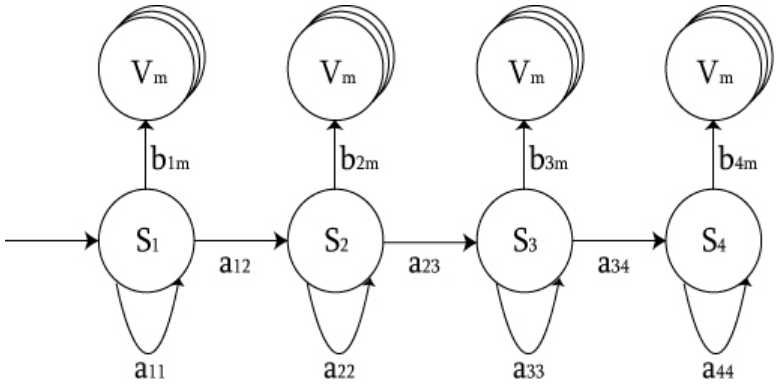


Figure 3.12. Left–Right Hidden Markov Model

become:

$$\alpha_1(1) = b_1(O_1) \tag{3.46}$$

$$\alpha_1(i) = 0 \quad \forall i \neq 1 \tag{3.47}$$

$$\alpha_{t+1}(j) = b_j(O_{t+1})[\alpha_{t-1}(j-1)a_{j-1j} + \alpha_{t-1}(j)a_{jj}] \tag{3.48}$$

For the Left–Right HMMs it is usually assumed that the last state is the final state. Intuitively this forces the recognition to occur only when the gesture is completed. The effect of this assumption is that the likelihood of a gesture is $\alpha_T(N)$ instead of $\sum_i \alpha_T(i)$.

The equations for the backward variable β become:

$$\beta_T(N) = 1 \tag{3.49}$$

$$\beta_T(i) = 0 \quad \forall i \neq N \tag{3.50}$$

$$\beta_t(i) = a_{ii}b_i(O_{t+1})\beta_{t+1}(i) + a_{ii+1}b_{i+1}(O_{t+1})\beta_{t+1}(i+1) \tag{3.51}$$

The time complexity of computing α and β decreases from $O(N^2T)$ to $O(NT)$,

where T is the sequence length and N is the number of states the Left Right HMM has. The training phase is also simplified, since most of the a_{ij} parameters are already zero.

3.2.2.2. Automatic Estimation of the Number of States. For an HMM to be able to describe a gesture accurately it needs to align itself with the observation sequence. Therefore the complexity of the gesture directly affects the number of hidden states the corresponding model should have. The system allows the user to supply the number of states for a gesture during training, but it also employs a method to automatically estimate this number. The automatic estimation algorithm first trains the system with 10 states and then examines the network to detect insignificant states. The insignificant states are defined as:

- states that have low self transition probabilities;
- states that are considerably similar to their neighboring states.

If a state has a very low self transition probability, it is either modeling noise or a small part of an edge that is already modeled by another state. Such states are dropped from the network. Also if a state is very similar to a neighboring state, this means that these states are trying to model the same edge, and therefore they can be merged. Note that dropping a state is straightforward for a Left–Right HMM. Experiments have shown that as the number of training samples increase, the effect of noise decreases and this algorithm can correctly reduce the number of states to the number of edges for geometric gestures such as triangle, square or star.

3.2.2.3. Gesture Spotting. The proposed system allows simulation of the mouse in order to replace it as an input device. Therefore the input stream coming from the cameras do not only contain meaningful gestures. The gesture spotting problem is detecting the defined gestures from the continuous input stream.

Spotting gestures in a continuous stream is a complex task, because the start and end points of a gesture are not known beforehand. Therefore the frames that are candidates for being a start frame need to be determined and all the gestures starting from a candidate frame and ending in the current frame should be considered as possible gestures. The candidate frames are defined as:

- frames in which the observation symbol has changed
- frames that are not older than a preset value, always including the oldest point

This simple algorithm detects the start of possible edges and considers them as the start points of gestures. Even though this method extracts the candidate sequences, it cannot decide whether a given sequence is one of the defined gestures or not.

Lee and Kim propose a method for online gesture spotting [62]. In particular, they define an adaptive threshold model, which is an ergodic model that connects every single state used in the defined gestures. The formal description of the threshold model is as follows:

- The states of the model are the states of each gesture in the system
- The states allow transitions to each other, i.e. they are fully connected
- The observation probabilities of the gesture states remain unchanged
- The self transition probabilities of the gesture states remain unchanged
- The rest of the transition probabilities are set to $a_{ij} = \frac{1-a_{ii}}{N-1} \forall j \neq i$

This ergodic model has each of the states every gesture has. Therefore the threshold model gives high likelihoods for every gesture sequence, but for meaningful gestures it is expected to give lower likelihoods than the dedicated HMMs due to the loss of the temporal information.

The adaptive threshold model has been implemented in the HMM based framework of the proposed system and it is found to give satisfactory results in distinguishing meaningful gestures. However as the number of gestures increases, the effect of the $O(N^2T)$ time complexity of evaluation of the threshold model becomes considerably high. Therefore Kim and Lee propose eliminating similar states by considering their relative entropy.

An important observation at this point reveals that the method used to estimate the number of hidden states of an HMM ensures that most of the hidden states will correspond to pure states, where only one observation symbol has a very high probability of being observed. Such state have high self transition probabilities and only one symbol is expected to be observed. Now consider the case, where such a pure state is included in the adaptive threshold model. The model will always favor such pure states, since the other states will reduce the likelihood of the candidate sequence. Therefore all such states can be omitted. Moreover the pure states should intuitively have the same self transition probabilities, since no direction is favored in the system. When only such pure states are allowed in the adaptive threshold model, the Viterbi algorithm reduces to a simple formula, which is based on the sequence length and number of symbol changes that occurs over the sequence. This simple approach is yet to be tested and is left as a future work.

3.2.3. IOHMM–Based Hand Gesture Recognizer

The complexity of hand gesture recognition arises from the existence of two sequences involved in the recognition process, since hand gestures may consist of combinations of hand trajectories and postures. Several methods used in supervised sequential learning can be applied to this problem with varying success rates. The reason for choosing IOHMMs and the implementation details will be discussed in this section.

The reasons behind employing Markovian models for recognition of hand trajectories have been previously discussed. Following the same reasoning, HMM and its variants are expected to give satisfactory results for the hand gesture recognition problem as well, due to the similar natures of both problems. The main difference between hand trajectory and hand gesture recognition is the involvement of a second sequence in the case of gestures, which is not a straightforward extension to the HMM based framework implemented in the proposed system. The system also aims to be dynamic and flexible enough to allow additional sequences to be included in recognition. The main purpose of a flexible design is allowing hand gestures with two hands.

There are a number of HMM variants suitable for hand gesture recognition that should be considered:

- Basic HMMs
- Coupled HMMs
- Asynchronous HMMs
- Input–Output HMMs

Extending the basic HMM based framework to include the posture sequence is possible by quantizing the hand shape parameters, namely the hand image moments and the angle, and considering them as part of the observable sequence. This is equivalent to forming new output symbols from the cross product of the hand motion and shape classes. This approach has several disadvantages, such as

- not being able to model gradual changes
- limiting the number of hand shapes beforehand
- coarse approximation of natural hand shapes
- increasing the effects of the imbalance problem of HMMs
- needing a large number of states to accurately model the gesture

The first three problems mentioned are caused by the quantization of hand shapes, which is a required step due to the quantization of hand motion parameters. Even though quantizing the postures significantly simplifies the problem, the trade off is not being able to model natural hand gestures accurately, which is one of the desired aspects of the proposed system. Moreover following the same logic behind choosing the number of states for the basic HMMs, if the combination of hand gesture and posture classes are taken as observation symbols, the number of states needed to model the gestures will be considerably higher. Consider a triangle shaped gesture where the hand needs to be waved in certain directions and order. The waving of the hand consists of gradual changing of the angle of a static hand shape, which needs at least three distinct posture classes to be approximated. If the gesture is simply waving the hand from left to right and then right to left on each edge of the triangle, 12 distinct states are required to accurately model this gesture, whereas three states are enough to model the triangular gesture. Therefore basic HMMs are not suitable for the gesture recognition problem. A more natural selection of a model is Coupled or Asynchronous HMMs, which are meant to model multiple data streams. In particular the Asynchronous HMMs use the same state sequence for all streams, but allow multiple emissions from each state. Therefore it can model both sequences without quantizing the posture sequence by letting the secondary emissions of each state to be continuous. Once again the trade–off is the number of states needed to correctly recognize gradual changes of the hand because of the static nature of each state. This suggests the need for a model with states that have dynamic probabilities.

The Coupled HMMs are also meant to model multiple streams of data, and they model the streams with multiple state sequences that are connected to each other in parallel. The number of states needed are even more than the basic HMMs and asynchronous HMMs under the assumption of a Left–Right topology. Once again the gradual changes of the hand cannot be modeled accurately due to the static nature of the states. These observations suggest that discrete states always fail to accurately

model gradual changes in the hand shape and a state space model is needed instead. Nevertheless the discrete states are good at modeling the abrupt changes of the gesture. This discussion reveals that a Switching State Space Model (SSSM) is more suitable for the task of gesture recognition.

SSSMs describe sequential data by modeling abrupt changes with discrete states and by modeling the gradual changes with state space models. Note that the proposed system employs the Kalman Filter, which is a state space model that describes the motion of the hand. The disadvantage of this method is that it is not an easy task to model the changes in the hand shape as a state space model.

Unlike the other HMM variants discussed here other than the SSSMs, the Input–Output HMMs have dynamic states that learn to adapt themselves to gradual changes in the data. Therefore the IOHMMs are inhomogeneous and a promising candidate for gesture recognition task. In particular the local models for transition and emission of IOHMMs can be flexible and non–linear models like MLPs, which can take a wider context of input into account, such as previous observations, time, or even other sequences. IOHMMs satisfy all the requirements of the proposed system and are expected to model hand gestures better than basic, coupled or asynchronous HMMs. An important observation is that if the input sequence of an IOHMM has a time dimension, the state networks adapt themselves according to the time, which reveals the similarities between SSSMs and IOHMMs. In this case the IOHMMs have the properties of SSSMs, and unlike the SSSMs they can learn the local motion models, omitting the complexity of describing the change of the hand motion as a state space model.

Each IOHMM state consists of neural networks that estimate the transition and emission probabilities from a given input sequence. Therefore it is a complex method to map the input sequence to the output sequence. Note that the output sequence need not be the direct consequence of the input and as long as the input and output

are correlated the neural networks can estimate the Bayesian network that describes their relationship. The structure, training and evaluation methods of HMMs have been discussed in Chapter 2 of this thesis.

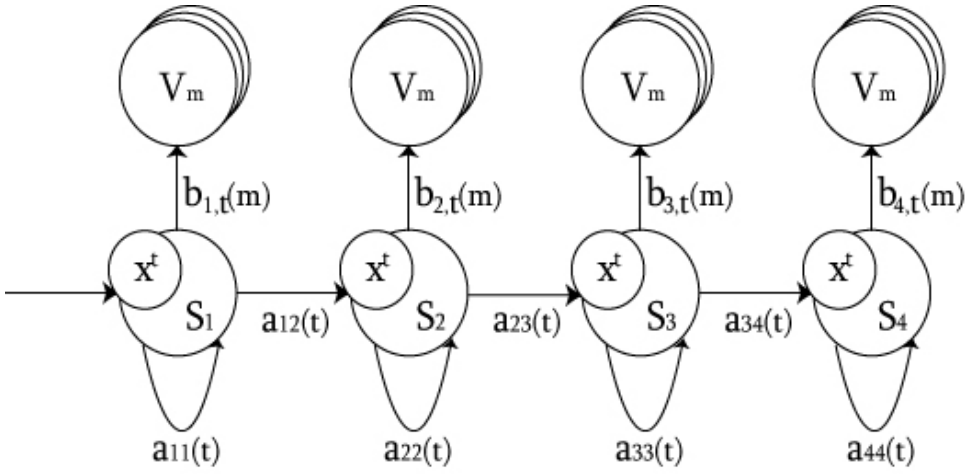


Figure 3.13. Left–Right IOHMM architecture

3.2.3.1. Left–Right IOHMMs. As in the case of HMMs, constraints on the topology of the IOHMM network using our a priori knowledge on gesture recognition problem can be imposed. It is intuitive to choose the topology as the analogous version of Left–Right HMMs, which are the Left–Right IOHMMs. Similarly each state is allowed to make a transition either to itself or to the next state, therefore restricting the number of outputs of the state transition neural networks to two. Note that the state transmission network of the last state has only one output, namely itself, that is always unity. The topology of a LR–IOHMM can be seen in Figure 3.13.

Each state of the IOHMMs consist of two neural networks, namely the state transition network N_i and the emission network O_i, where i indicates the state. The neural networks estimate the state transition and emission probabilities for each given input, therefore the output is continuous and the networks are regressors. At each time frame these networks estimate the A and B matrices, which are not static in the case of IOHMMs and therefore can be denoted as $A(t)$ and $B(t)$. The elements of these matrices become $a_{ij}(t)$ and $b_{j,t}(m)$. Once these matrices are calculated the evaluation

equations are the same as that of basic HMMs. The Π vector can be omitted due to the Left–Right topology constraint since $\pi_1 = 1$ and the other elements are zero without the need to consider x_1.

$$\Pi = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.52}$$

$$A(t) = \begin{bmatrix} a_{11}(t) & a_{12}(t) & 0 & 0 & \cdots & 0 \\ 0 & a_{22}(t) & a_{23}(t) & 0 & \cdots & 0 \\ 0 & 0 & a_{33}(t) & a_{34}(t) & \cdots & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & \cdots & 0 & a_{n-1n-1}(t) & a_{n-1n}(t) \\ 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \tag{3.53}$$

$$B(t) = \begin{bmatrix} b_{1,t}(1) & b_{1,t}(2) & \cdots & b_{1,t}(m) & \cdots & b_{1,t}(M) \\ b_{2,t}(1) & b_{2,t}(2) & \cdots & b_{2,t}(m) & \cdots & b_{2,t}(M) \\ & & \vdots & & & \\ b_{N-1,t}(1) & b_{N-1,t}(2) & \cdots & b_{N-1,t}(m) & \cdots & b_{N-1,t}(M) \\ b_{N,t}(1) & b_{N,t}(2) & \cdots & b_{N,t}(m) & \cdots & b_{N,t}(M) \end{bmatrix} \tag{3.54}$$

For the evaluation problem, the equations to calculate the forward variable α become:

$$\alpha_1(1) = b_{1,1}(O_1) \tag{3.55}$$

$$\alpha_1(i) = 0 \quad \forall i \neq 1 \tag{3.56}$$

$$\alpha_{t+1}(j) = b_{j,t+1}(O_{t+1})[\alpha_t(j-1)a_{j-1j}(t+1) + \alpha_t(j)a_{jj}(t+1)] \tag{3.57}$$

As for the Left–Right HMMs it is assumed that the last state is the final state. Therefore the likelihood of a gesture is $\alpha_T(N)$ instead of $\sum_i \alpha_T(i)$.

The equations for the backward variable β become:

$$\beta_T(N) = 1 \tag{3.58}$$

$$\beta_T(i) = 0 \quad \forall i \neq N \tag{3.59}$$

$$\beta_{t-1}(i) = a_{ii}(t)b_{i,t}(O_t)\beta_t(i) + a_{ii+1}(t)b_{i+1,t}(O_t)\beta_t(i+1) \tag{3.60}$$

The number of output nodes of emission networks O_i should be equal to the number of observation symbols M, which is 15 for the proposed system, whereas the number of output nodes of state transition networks should be the number of possible states the state allows transitions to, which is two for a Left–Right IOHMM, namely the transitions to itself and the next state. The transition network of the last state allows transitions to only itself. The outputs of the networks of state i at time t are denoted $\eta_{im,t}$ for the emission networks, and $\varphi_{ij,t}$ for the state transition networks:

$$a_{ij}(t) = \varphi_{ij,t} \tag{3.61}$$

$$b_{i,t}(m) = \eta_{im,t} \tag{3.62}$$

In Chapter 2 the log–likelihood of a training set has been shown to be:

$$\Omega(\Theta, \hat{\Theta}) = \sum_{p=1}^{P} \left(\sum_{t=1}^{T} \sum_{i=1}^{N} \hat{g}_t(i) \log \eta_{iO_t,t} + \sum_{t=2}^{T} \sum_{i=1}^{N} \sum_{j=1}^{N} \hat{h}_t(i,j) \log \varphi_{ij,t} \right) \tag{3.63}$$

For Left–Right IOHMMs the possible values of j are i and $i+1$ and the equation becomes:

$$\Omega(\Theta, \hat{\Theta}) = \sum_{p=1}^{P} \left(\sum_{t=1}^{T} \sum_{i=1}^{N} \hat{g}_t(i) \log \eta_{iO_t,t} + \sum_{t=2}^{T} \sum_{i=1}^{N} \sum_{j=i,i+1} \hat{h}_t(i,j) \log \varphi_{ij,t} \right) \tag{3.64}$$

where $\hat{g}_t(i)$ and $\hat{h}_t(i,j)$ are the estimated values of $g_t(i)$ and $h_t(i,j)$ respectively. The values $g_t(i)$ and $h_t(i,j)$ are the state and transition posterior probabilities defined as before. Note that $h_t(i,j) = 0$ when j is not i or $i+1$.

The neural networks N_i and O_i are MLPs, which allows the system to learn nonlinear functions that map the input sequence to the state transition and emission probabilities. In particular the state transition networks have H_N and the emission networks have H_O hidden nodes. The outputs of the networks give the rows of the $A(t)$ and $B(t)$ matrices, which requires them to be normalized. The normalization of the outputs are ensured by applying the softmax operation to the output nodes. The detailed description of these networks will be given in the next sections.

3.2.3.2. Neural Network Architectures. The state transition and emission networks are both MLPs with similar structures, which differ in number of output and hidden nodes. The structure of the networks are that of a common MLP classifier, where the hidden nodes apply the nonlinear sigmoid function to the weighted sum of the inputs, and the output nodes apply the softmax function to the weighted sum of the hidden node values. However unlike the classifiers, the IOHMM networks are regressors and try to estimate the real valued probabilities. Therefore the output of the softmax

operation is left as it is in the IOHMM networks. The equations of each layer are as follows:

$$z_h = \frac{1}{1+\exp\left[-\left(\sum_{d=1}^{D} w_{hn}x_n + w_{h0}\right)\right]} \quad h = 1,\ldots,H \tag{3.65}$$

$$o_j = \mathbf{v}_j^T\mathbf{z} = \sum_{h=1}^{H} v_{jh}z_h + v_{j0} \tag{3.66}$$

$$y_j = softmax(o_j) = \frac{\exp o_j}{\sum_k \exp o_k} \tag{3.67}$$

where $\mathbf{x}$ is the input vector with D dimensions, w_{hn} is the weight between hidden node h and input node n, $\mathbf{z}$ is the vector of the H hidden node values and v_{jh} is the hidden weight between the output node j and the hidden node h. w_{h0} and v_{j0} are the weights of the bias nodes of corresponding layers. Note that H may be different for state transition and emission networks, whereas the number of input dimensions D does not differ. The number of output nodes is two for state transition networks and 15 for emission networks as explained in the previous section. Since the number of output nodes is two for state transition networks, sigmoid can be used instead of softmax in the final step, but softmax handles more general IOHMM topologies as well, where there may be more than two allowed states for transition.

3.2.3.3. Training. Training an IOHMM based system using the Generalized Expectation Maximization (GEM) method has been explained in Chapter 2 of this thesis, where it is stated that the M–Step depends on the specific structures of the neural networks. Therefore the E–Step derived for the IOHMMs can directly be used for Left–Right IOHMMs, but the M–Step should be specifically derived for the networks described in the previous section.

The GEM method uses gradient ascent procedure to maximize the likelihood of a given training set. Therefore the weights of the neural networks should be updated

using the derivative of the log–likelihood expectation function $\Omega(\Theta, \hat{\Theta})$ with respect to the specific weights. In gradient ascent procedure the update is in the same direction as the derivative:

$$\rho_{jk} = \rho_{jk} + \mu \frac{\partial \Omega(\Theta, \hat{\Theta})}{\partial \rho_{jk}} \ , \ \forall j, k \tag{3.68}$$

where ρ_{jk} is a generic weight and μ is the learning factor. The derivatives are:

$$N_i \ : \ \frac{\partial \Omega(\Theta, \hat{\Theta})}{\partial \theta_{jk}} = \sum_{p=1}^{P} \sum_{t=2}^{T} \sum_{j=i,i+1}^{N} \frac{\hat{h}_t(i,j)}{\varphi_{ij,t}} \frac{\partial \varphi_{ij,t}}{\partial \theta_{jk}} \tag{3.69}$$

$$O_i \ : \ \frac{\partial \Omega(\Theta, \hat{\Theta})}{\partial \phi_{jk}} = \sum_{p=1}^{P} \sum_{t=1}^{T} \frac{\hat{g}_t(i)}{\eta_{iO_t,t}} \frac{\partial \eta_{iO_t,t}}{\partial \phi_{jk}} \tag{3.70}$$

where θ_{jk} needs to be replaced with the weights of the N network, and ϕ_{jk} needs to be replaced with the weights of the O network. The specific update rules for the actual hidden weights v_{sh} of the state transition network are as follows:

$$\begin{aligned}
\Delta v_{sh} &= \mu \frac{\partial \Omega(\Theta, \hat{\Theta})}{\partial v_{sh}} \\
&= \mu \sum_{p=1}^{P} \sum_{t=2}^{T} \sum_{j=i,i+1}^{N} \frac{\hat{h}_t(i,j)}{\varphi_{ij,t}} \frac{\partial \varphi_{ij,t}}{\partial v_{sh}} \\
&= \mu \sum_{p=1}^{P} \sum_{t=2}^{T} \sum_{j=i,i+1} \frac{\hat{h}_t(i,j)}{\varphi_{ij,t}} \varphi_{ij,t}(\delta_{js} - \varphi_{is,t}) z_h^t \\
&= \mu \sum_{p=1}^{P} \sum_{t=2}^{T} \left(\sum_{j=i,i+1} \hat{h}_t(i,j)\delta_{js} - \sum_{j=i,i+1} \hat{h}_t(i,j)\varphi_{is,t} \right) z_h^t \\
&= \mu \sum_{p=1}^{P} \sum_{t=2}^{T} \left(\hat{h}_{is,t} - \hat{g}_t(i)\varphi_{is,t} \right) z_h^t \qquad (3.71) \\
v_{sh} &= v_{sh} + \Delta v_{sh} \qquad (3.72)
\end{aligned}$$

From these equations the update rules for the weights w_{hn} can be derived using the

chain rule:

$$\begin{aligned}
\Delta w_{hn} &= \mu \frac{\partial \Omega(\Theta, \hat{\Theta})}{\partial w_{hn}} \\
&= \mu \sum_{p=1}^{P} \sum_{t=2}^{T} \left[\sum_{j=i,i+1} \left(\hat{h}_t(i,j) - \hat{g}_t(i) \varphi_{ij,t} \right) v_{jh}^t \right] z_h^t (1 - z_h^t) x_n^t \qquad (3.73) \\
w_{hn} &= w_{hn} + \Delta w_{hn} \qquad (3.74)
\end{aligned}$$

Note that for simplicity the two output nodes of N_i are always numbered as i and $i+1$ so that $\varphi_{ii,t}$ and $\varphi_{ii+1,t}$ refer to the correct outputs. The update rules for the hidden weights v_{sh} of the emission network are as follows:

$$\begin{aligned}
\Delta v_{sh} &= \mu \frac{\partial \Omega(\Theta, \hat{\Theta})}{\partial v_{sh}} \\
&= \mu \sum_{p=1}^{P} \sum_{t=1}^{T} \frac{\hat{g}_t(i)}{\eta_{iO_t,t}} \frac{\partial \eta_{iO_t,t}}{\partial v_{sh}} \\
&= \mu \sum_{p=1}^{P} \sum_{t=1}^{T} \frac{\hat{g}_t(i)}{\eta_{iO_t,t}} \eta_{iO_t,t} (\delta_{O_t s} - \eta_{is,t}) z_h^t \\
&= \mu \sum_{p=1}^{P} \sum_{t=1}^{T} \hat{g}_t(i) (\delta_{O_t s} - \eta_{is,t}) z_h^t \qquad (3.75) \\
v_{sh} &= v_{sh} + \Delta v_{sh} \qquad (3.76)
\end{aligned}$$

Once again from these equations the update rules for the weights w_{hn} can be derived using the chain rule:

$$\begin{aligned}
\Delta w_{hn} &= \mu \frac{\partial \Omega(\Theta, \hat{\Theta})}{\partial w_{hn}} \\
&= \mu \sum_{p=1}^{P} \sum_{t=1}^{T} \left[\sum_{m=1}^{M} \hat{g}_t(i) (\delta_{O_t m} - \eta_{im,t})) v_{mh}^t \right] z_h^t (1 - z_h^t) x_n^t \\
&= \mu \sum_{p=1}^{P} \sum_{t=1}^{T} \hat{g}_t(i) \left(v_{O_t h} - \sum_{m=1}^{M} \eta_{im,t} v_{mh}^t \right) z_h^t (1 - z_h^t) x_n^t \qquad (3.77) \\
w_{hn} &= w_{hn} + \Delta w_{hn} \qquad (3.78)
\end{aligned}$$

This concludes the M–Step of the GEM method. The training stops when the log–likelihood converges. The choice of parameters and the results are given in Chapter 4.

3.2.3.4. Estimating the Number Of States. In the HMM based framework it has been shown that the number of states should be chosen at least as much as the number of distinct edges a gesture has. The same logic can be followed for IOHMMs that have normalized time as input, by noticing their similarity to SSSMs, where the discrete models are used to model abrupt changes in the sequential data. Since the complexity of the state networks are adjustable they can describe any local model, and the discrete states of the IOHMM can be used to model the different edges. Note that unlike the SSSMs the neural networks of the IOHMMs can also learn the abrupt changes, and in the extreme case a single state can model the entire gesture. This observation reveals that there is a trade off between the local model complexity and the number of discrete states, which corresponds to the complexities of the MLPs and the HMM network.

Different combinations of complexities for MLPs and IOHMMs have been tested, and it has been concluded that the best results are achieved where the MLPs and IOHMMs have similar complexity levels. In particular the best results have been reached for three distinct IOHMM states and five hidden nodes for each network. In the case of 2D gestures, three hidden nodes were enough to describe the local models.

3.2.3.5. Input Sequence. The structure of IOHMM allows learning a mapping function of some input sequence onto an output sequence. In the case of gesture recognition this brings up the question of which of the sequences to choose as the input sequence. Therefore the differences of the sequences should be considered.

The main difference between gesture and posture sequences is that the hand gesture is discrete, whereas the hand shape features, namely Hu Moments and the angle are continuous. Note that it is more natural to map continuous sequences to

discrete ones using MLPs. Moreover the shape information has more dimensions. But the IOHMM also conditions the state transition probabilities on the input sequence. Therefore setting the posture information as the input sequence implies that the state transition probabilities are conditioned on the hand shape, which is not necessarily true, since the abrupt changes are usually caused by the changes in the hand motion, but also can be caused by the changes in hand shape. Addition of normalized time Γ as an input regardless of the chosen input sequence handles this problem, since the actual state transition are actually based on fractions of the time required to perform the entire gesture.

The normalized time variable is defined intuitively as follows:

- $\Gamma(0) = 0$
- $\Gamma(T) = 1$
- $\Gamma(t) = \frac{t}{T}$

where T is the sequence length.

Note that Γ is the actual reason for the similarity between the proposed IOHMM based framework and the SSSM, since it causes the local model information to be indexed and to behave like a state space machine. Therefore Γ is the reason the gradual changes are accurately modeled and correctly ordered in the local models. Several tests have been conducted to see the effect of the time variable, and it has been shown that its inclusion as in input significantly increases the recognition rate and reduces the complexity by allowing less states, since the MLPs are already capable of handling abrupt changes. In the extreme case an IOHMM with only one state has been shown to accurately model gestures, since the temporal information is not lost due to the existence of the time variable.

Following the discussion above the proposed system uses the state networks to map the hand shape sequence onto the hand motion, where the normalized time is also included as input and the hand shape sequence consists of the Hu moments and the angle of the hand. The number of input nodes the MLPs have is therefore 17. In the case of a single camera the number of nodes is reduced to nine, which consist of the seven Hu moments for the single view, the angle and the time variable.

3.2.3.6. Adaptive Threshold Model. The approach of Lee and Kim in their method for generating an adaptive threshold model for HMMs relies on the fact that HMMs are homogeneous. Therefore when the gesture networks are disrupted to build an ergodic model the temporal information is lost. The problem of this approach with IOHMMs is that even a single hidden state is capable of describing the entire gesture. Therefore the same method is not likely to hold for IOHMMs.

Before designing and testing an analogous model, however, the previous discussion reveals that it is possible in an IOHMM based framework, to describe every gesture in the system with a single model, which has only a single hidden state. This novel approach creates a threshold model by training an IOHMM with a single state with the entire dataset. The tests with such a model confirm that such a model has the properties of an adaptive threshold model:

- it gives higher likelihoods for meaningless, undefined gestures, since it is a more general model than the others
- it gives higher likelihoods than all the IOHMMs but the dedicated one in the case of a meaningful gesture, since the dedicated IOHMM better describes the gesture, because it is trained only with the actual training set corresponding to that gesture

As in the case of Lee and Kim's approach, this is a heuristic that is expected to

hold, and a factor may need to be introduced to the system that slightly lowers the likelihoods of the threshold model to make sure a higher likelihood is attainable by the dedicated model. This factor was around 5% for the HMM based framework, and it is experimentally shown that this factor gives very good results for IOHMMs as well.

This new adaptive threshold model is characteristically different than Lee and Kim's model, since unlike their model this model needs to be trained. Also the time complexity of their approach is that of evaluating an ergodic threshold model, which is $O(N^2T)$ and suffers from the number of gestures and therefore distinct hidden states in the system. The model proposed here has only one state and therefore the time complexity of evaluating this model is $O(T)$ times the time complexity of generating an output from the emission network of the state, which is much faster and suitable for real time implementations.

4. EXPERIMENTS

4.1. Datasets

In order to test the proposed system two separate datasets have been created, one consisting of eight 2D hand signs and one consisting of 20 3D hand signs. Each dataset consists of hand sign pairs that share the same motion but require different hand posture sequences to be performed. The smaller 2D dataset is used to optimize the training parameters, and the 3D dataset is used to test the recognition success rates for different cases.

Motion	Sign #1	# Samples	Sign #2	# Samples
Circle	Circle A	102	Circle N	84
Cross	Cross A	71	Cross N	79
Stamp	Stamp 0	61	Stamp 1	59
Triangle	Triangle A	20	Triangle N	66
Total: 577 Samples				

Table 4.1. 2D Dataset

Information on the datasets and the signs are given Table 4.1 and Table 4.2. "Motion" is the distinct hand motion that distinguishes different signs. Each motion is shared by two distinct signs, except for the "Left–Right" motion, which is shared by four different signs. The letter "A" in the sign names signify an artificial hand posture, where the hand is open and the posture does not change, whereas "N" stands for "natural". The natural hand signs usually involve a relaxed hand shape that dynamically changes according to the hand motion. Similarly the letter "0" stands for a fist, while "1" signifies one stretched finger.

The smaller 2D dataset is performed by one user, since it is not used for final

Motion	Sign #1	# Samples	Sign #2	# Samples
Circle Left	CircleL A	128	CircleL N	98
Circle Right	CircleR A	149	CircleR N	150
Circle Forward	Next	152	CircleF A	118
Cross	Cross A	56	Cross N	38
Forward Thrust	Peace	77	Grab	166
Stamp	Stamp 0	67	Stamp 1	49
Down	Sit	32	No	138
Triangle	Triangle A	20	Triangle N	66
Left–Right	Applause	100	Wave	55
Left–Right	Page	34	Cut	42
Total: 1735 Samples				

Table 4.2. 3D Dataset

recognition rate tests. The larger 3D dataset consists of gestures that were performed by either two or three users. The signs with fewer than 50 samples are put to the dataset to test how many sequences are needed to accurately learn a sign.

The previous discussion on using normalized time as input in Chapter 2 suggested that the inclusion of this input would significantly increase the recognition success rate, especially for distinguishing meaningless gestures. In order to test this hypothesis copies of the datasets have been created that includes the time input. The tests performed and their results are given in the following sections.

4.2. Training

The training of an IOHMM that has MLPs as state networks is a complex task that involves several parameters, which need to be optimized to minimize the test error and to maximize the convergence speed. A series of tests were conducted to

first choose the MLP parameters that minimized the test error for a given number of IOHMM states and hidden nodes for the state networks. A number of new parameters were introduced and optimized to accelerate the convergence. Secondary tests were then conducted to choose the number of IOHMM states and MLP hidden nodes to minimize the test error. In order to attain consistent and reliable test error rates, 5×2 Cross Validation method is used. The following sections introduce the parameters used in the training and modeling process.

4.2.1. Training Parameters

The GEM algorithm first calculates the expectations from the current parameter set of the IOHMM in the E-Step, and then updates the neural networks accordingly in the M-Step. The training stops when the likelihood of the training set converges. As in the gradient descent algorithms the absolute value of the function that maximizes the expectation is not directly used. Instead, the learning process is expected to slow down to a certain rate, where the function is taken to be converged. Therefore the relative change of the likelihood between the epochs is used to decide whether the training is finished or not. If the relative change is lower than a threshold ϵ, the method stops.

As a consequence of having a large learning factor μ, the likelihood may start to oscillate. Therefore the learning factor should be decreased after each iteration. The rate at which the learning factor decreases is called the decay rate. If the learning factor is too small, it takes too many iterations to reach the local maximum. Therefore if there is learning, the learning factor should be increased by a small value. To decide if there is learning, the mean of several older likelihoods is used, the number of which is called the memory size. Similarly if there is no learning, the learning factor is higher than required and needs to be decreased. Each of these operations require introduction of another parameter to the system that needs to be optimized to ensure that:

- the learning always stops,
- a local maximum is reached,
- the likelihood does not diverge,
- the algorithm does not finish prematurely due to oscillations,
- the learning finishes before a certain maximum number of epochs is reached.

There are other methods to accelerate the learning process, namely by introducing momentum factor or by means of alternative update rules. The momentum factor has been found to be hard to stabilize as it often caused oscillations and divergence. Therefore it is omitted in the current implementation of the system.

One important observation is that the auxiliary function Ω used to derive the weight update rules is twice differentiable with respect to the weights w_{hn}, which allows using Newton's method to find the local maximum, which is the root of the second derivative of Ω with respect to the weights. This promising approach is left as a future work, since it only affects the convergence rate and not the recognition rates.

4.2.2. Hidden Nodes vs. IOHMM States

In Chapter 3 the effect of the time input Γ is shown to be giving the IOHMM a structure similar to an SSSM. It is suggested that an IOHMM that has Γ as input can represent a gesture with only one state without losing the temporal information. This implied a trade off between the number of IOHMM states and the number of hidden nodes the MLPs have, since MLPs need to be more complex to be able to describe the entire gesture if there are fewer distinct states. It is of interest for the system what combination of these values will maximize the recognition rate. Therefore a series of tests have been conducted to find these values. In particular it is shown that having more than three distinct states does not increase the recognition rate of gestures significantly, where three hidden nodes for each state transition and emission

network were enough to handle the local complexity of 2D signs and five hidden nodes were enough for the 3D signs. The test results are given at the end of this chapter.

4.2.3. 5×2 Cross Validation

In order to get reliable test results 5×2 Cross Validation method is used. This method shuffles and divides the training set in two different sets and uses one of the sets for training and the other for testing. Then it exchanges the test and training sets and recalculates the test error. This is repeated five times, ending with results from ten different training processes, of which the mean is estimated and taken as the final test error. All of the test errors given in the next section are results of 5×2 Cross Validation tests applied to the training sets.

4.3. Test Results

This section gives the results of the tests conducted to estimate the best combination of number of hidden nodes and distinct states, and the recognition rates on different training sets.

4.3.1. Parameter Optimization Tests

Parameters of the training process given in the previous sections can be handled in two different sets. One set, which includes the learning factor, its decay, increase and decrease rates, convergence threshold, memory size and maximum number of epochs, only deals with the speed of the convergence rate, premature conclusions and divergences. The other set consists of the number of IOHMM states and the number of hidden nodes of each MLP, and deals with undertraining and overtraining. The first series of tests were conducted on the 2D dataset to optimize the first set of parameters in order to ensure efficient convergence. Combinations of several different values for

each parameter have been tested with cross validation. Estimated optimum values of these parameters are given in Table 4.3.

Parameter	Explanation	Estimated Value
μ	Learning Factor	0.65
ϵ	Convergence Threshold	0.0005
M_e	Maximum Epoch #	100
ψ	Memory Size	4
δ	Decay Rate	0.9
λ^+	Increase in Learning Factor	0.01
λ^-	Decrease Rate in Learning Factor	0.6

Table 4.3. Optimized training parameters

The second series of tests were conducted using the estimated learning parameters to find the optimum number of hidden nodes and IOHMM states. As explained in Chapter 2, as the number of IOHMM states decreases the local models need to be more complex to avoid undertraining, and hence the number of hidden nodes increases. Following the same logic, the number of hidden nodes decrease as the number of distinct states increases, since the local models need to describe simpler subsequences of gestures. Note that even though there seems to be a trade off between these parameters, they both can be increased at the same time, in which case either overtraining occurs, or the evaluation speed of the models decrease to unacceptable levels with insignificant changes in success rate. Therefore the optimization should be done by considering:

- undertraining
- overtraining
- time complexity

and the optimized parameters are the numbers of:

- IOHMM states K'
- hidden nodes of state transition network H_N
- hidden nodes of emission network H_O

An exhaustive search for the best parameters using cross validation is not tractable. Therefore the best combination of hidden node numbers were estimated first. The test conducted is as follows:

- For each combination of parameters from the set $\{1, 3, 5, 7, 20\} \times \{1, 3, 5, 7, 20\}$
  - Each IOHMM is initialized with a random number of states $K' \in \{1, \ldots, 8\}$
  - 5×2 Cross Validation is performed
  - Results are tabulated
- Results are compared

The results revealed that the best results were achieved when $H_N = H_O$, regardless of the number except for the case when they were both one, which gave significantly worse results. This implies that three hidden nodes are enough to model most of the hand signs. Note that the dataset used for this test is the 2D dataset, which has nine dimensional inputs instead of 17 and therefore require less hidden nodes to represent the model. This test shows that H_N and H_O can be taken as equal:

$$H_N = H_O = H \tag{4.1}$$

# of IOHMM States K'	1	2	3	5	8	20
# of Hidden Nodes H	8	5	3	3	3	3

Table 4.4. Best H values for given K'

# of Hidden Nodes H	1	2	3	5	7	20
# of IOHMM States K'	7	5	3	3	3	2

Table 4.5. Best K' values for given H

The last series of tests are to estimate the best combination of H and K'. The test results are given in Tables 4.4 and 4.5. These tests reveal that for $H \geq 3$ increasing K' beyond three does not significantly increase the recognition rate, and similarly increasing H beyond three does not decrease K' further, unless 20 hidden nodes are used, which significantly reduces the evaluation speed. Therefore the best combination of H and K' has been found to be $H = 3$, $K' = 3$ for the 2D dataset.

Parameter	Explanation	2D Signs	3D Signs
H_N	# Hidden Nodes of State Transition Network	3	5
H_O	# Hidden Nodes of Emission Network	3	5
K'	# IOHMM States	3	3

Table 4.6. Optimized architecture parameters

Even though these tests do not reveal the best combination of parameters for the 3D dataset, they provide enough heuristics to significantly simplify the parameter search problem. In particular the best choice of number of IOHMM states remains the same for the 3D dataset, since the complexity of the gestures does not change. Since the number of inputs increase from nine to 17, the mapping problem is more complex and the number of hidden nodes needed to avoid undertraining is expected to be higher for the 3D dataset. Hence the parameter search problem can be reduced to testing a few values $H \geq 3$ for $K' = 3$. Setting $K' = 3$ and $H = 3, 5, 7, 9$ and conducting cross validation reveals that the best value of H is five for three IOHMM states, which is higher than three as expected. This concludes the parameter optimization phase and the estimated optimum parameters can be used to test the proposed system for recognition performance. These parameters are listed in Table 4.6.

4.3.2. Recognition Performance Tests

This section gives the tests conducted to measure the recognition success rate of the proposed IOHMM based framework with the optimized parameters of the last section. The success rate has been tested on training sets that consist of

- hand signs with distinct hand motions,
- hand signs that may have the same hand motion but with different hand posture sequences,
- continuous runs of the system to test the system's ability to differentiate meaningful gestures,
- the above sets without normalized time Γ as input.

To generate the required datasets the 3D dataset given in Table 4.2 is used. Tests conducted and modifications on the database will be explained in detail in the following sections.

4.3.2.1. Hand Signs with Distinct Hand Motions. To test the performance of the system for hand signs with distinct hand motions, the signs in the column "Sign #1" of Table 4.2 are used. Note that the last two signs, "Applause" and "Page" share the same hand motion, but the same motion is performed twice for the former sign. Therefore related hand motions can be regarded as distinct and both of the signs are included in this set and the set consists of ten distinct hand signs. Note that Γ is included as input by default.

Using the optimal parameters of the previous section, 5×2 Cross Validation is used to estimate the success rate of recognition. Table 4.7 gives the results and reveals that the system attains 97.6% accuracy in classification of hand signs with distinct hand motions, with Γ as input. The next test is conducted to measure the performance gain

Sign #	Sign Name	# Samples	Mean Error % in Gesture	Normalized Error % in Dataset
1	CircleL A	128	7.03	1.1
2	CircleR A	149	1.2	0.22
3	Cross A	56	0.0	0.0
4	Peace	77	0.0	0.0
5	Stamp 0	67	0.9	0.07
6	Sit	32	0.0	0.0
7	Applause	100	4.6	0.56
8	Next	152	0.26	0.04
9	Page	34	0.0	0.0
10	Triangle A	20	18.0	0.44
Total # Samples:		815	Total Error:	2.4%
Total Success Rate: 97.6%				

Table 4.7. Recognition rate for hand gestures with distinct hand motions with normalized time Γ as Input

due to inclusion of Γ as input in this problem. Therefore the same training set is used without Γ. Table 4.8 shows that the attained accuracy is 95.91% in this case, implying that the time input caused an increase of 1.69% in the first case.

4.3.2.2. Hand Signs with Unrestricted Hand Motions. The test that will reveal the true recognition accuracy of the proposed system for the general hand sign recognition problem should be conducted on an unrestricted dataset, which may consist of hand signs sharing the same motion or posture sequences. Therefore the entire 3D dataset is used in the tests of this section. The hand signs and recognition rates are given in Table 4.9. Note that Γ is included by default.

Sign #	Sign Name	# Samples	Mean Error % in Gesture	Normalized Error % in Dataset
1	CircleL A	128	2.03	0.31
2	CircleR A	149	10.93	2.0
3	Cross A	56	0.0	0.0
4	Peace	77	0.26	0.02
5	Stamp 0	67	1.2	0.1
6	Sit	32	0.0	0.0
7	Applause	100	12.0	1.46
8	Next	152	0.26	0.04
9	Page	34	1.17	0.04
10	Triangle A	20	2.99	0.07
Total # Samples:		815	Total Error:	4.09%
Total Success Rate: 95.91%				

Table 4.8. Recognition rate for hand gestures with distinct hand motions without the time input

Table 4.9 reveals that the sign "Stamp 1" is not classified correctly half of the time. The reason is that the signs "Stamp 0" and "Stamp 1" differ only by an outstretched finger, which is the fastest point on the hand due to the hand motion and gets blurred out. It is also not clear whether Hu moments are capable of creating enough distinction between the two hand postures that differ just by a finger. The analysis of this problem is left as future work. In particular Hu moments of the hand contour will be considered to increase the effect of the hand silhouette, and hence the fingers. Including the "Stamp" signs in the database reduces the recognition rate by 2.05%. Since the recognition rate for distinct motions is 97.6%, this suggests that the proposed system can distinguish the most general hand signs with an accuracy rate very close to the recognition rate of signs involving distinct hand motions.

Sign #	Sign Name	# Samples	Mean Error % in Gesture	Normalized Error % in Dataset
1	CircleL A	128	9.51	0.7
2	CircleR A	149	0.48	0.04
3	Cross A	56	2.13	0.06
4	Peace	77	14.41	0.65
5	Stamp 0	67	14.52	0.57
6	Sit	32	0.74	0.01
7	Applause	100	2.38	0.13
8	Next	152	13.66	1.2
9	Page	34	3.51	0.06
10	Wave	20	13.79	0.15
11	CircleL N	98	2.03	0.11
12	CircleR N	150	1.11	0.09
13	Cross N	38	3.41	0.05
14	Grab	166	3.02	0.29
15	Stamp 1	49	51.28	1.48
16	No	138	1.18	0.09
17	Triangle A	55	0.65	0.02
18	Triangle N	66	0.7	0.02
19	Cut	42	2.37	0.05
20	CircleF A	118	0.53	0.03
Total # Samples:		1735	Total Error:	5.9%
Total Success Rate: 94.1%				

Table 4.9. Recognition rate for unrestricted hand gestures with the time input

As before, the tests are repeated without including Γ as input to see its effect on the recognition. The results in Table 4.10 show that without Γ as input, the recognition rate decreases by 0.4%. It is not expected at this point to see a significant effect of Γ,

since all the gestures classified are already meaningful and have different hand posture sequences. Γ is especially expected to be of use while ditinguishing meaningful gestures from continuous runs.

Sign #	Sign Name	# Samples	Mean Error % in Gesture	Normalized Error % in Dataset
1	CircleL A	128	1.65	0.12
2	CircleR A	149	1.95	0.17
3	Cross A	56	2.83	0.09
4	Peace	77	2.72	0.12
5	Stamp 0	67	21.41	0.84
6	Sit	32	0.0	0.0
7	Applause	100	11.35	0.66
8	Next	152	1.56	0.14
9	Page	34	0.0	0.0
10	Wave	20	13.21	0.15
11	CircleL N	98	9.93	0.56
12	CircleR N	150	1.23	0.11
13	Cross N	38	4.71	0.08
14	Grab	166	8.27	0.79
15	Stamp 1	49	51.69	1.49
16	No	138	5.75	0.46
17	Triangle A	55	0.62	0.02
18	Triangle N	66	0.68	0.03
19	Cut	42	11.58	0.28
20	CircleF A	118	2.60	0.18
Total # Samples:		1735	Total Error:	6.3%
Total Success Rate: 93.7%				

Table 4.10. Recognition rate for unrestricted hand gestures without the time input

The result given in Table 4.9 is the overall success rate of the proposed system, which is 94.1% including the "Stamp" postures that are not learned properly. With more distinct signs the system is expected to attain a 2% higher accuracy.

4.4. Reasons for Error

The possible reasons for error are as follows:

- The training samples are extracted from continuous runs of the system and are prone to segmentation error.
- The samples are captured on different computers and therefore with different camera setups, which may unexpectedly increase or decrease the correlation of distinct hand signs.
- For fast hand motions the fingers tend to disappear due to motion blur.

Note that each of these problems can be avoided by:

- properly capturing samples one by one,
- capturing the samples using a single camera setup,
- using a camera with frame rate higher than 30 fps.

and the system can attain higher accuracies. This is left as future work since the proposed system already attains very high recognition rates, which proves its validity.

4.5. Hand Tracking Computation Time Performance

The system keeps track of the CPU time needed for each of the algorithms it runs, which simplifies the task of detecting the bottlenecks and optimizing the system. A standard 30 fps camera prepares an image in its buffer every 33 ms on the average,

and the system aims not to miss any of these images. Consequently the system has to complete acquiring the images, detecting the hand in both images, extracting their shapes, calculating the Hu moments, filtering the trajectories, reconstructing the 3D coordinates, filtering the 3D trajectory, quantizing the velocity vector, spotting the gesture and recognizing the sign in less than 33 ms.

The proposed system is tested on a 2.4 GHz Pentium IV PC. The algorithms explained in Chapter 3 allows the system to complete the hand detection, segmentation and moment calculation phases in less than 1.5 ms for each image on the average. The Kalman Filtering phase takes 0.2 ms on the average, leaving enough time for gesture spotting and recognition phases, which are the actual bottlenecks in the algorithm. Nevertheless the system is optimized for speed and is capable of handling 30 stereo image pairs per second, which is the limit of the cameras used.

5. CONCLUSIONS

This study used IOHMMs to model hand gestures, which is not an extensively studied approach so far. In particular, IOHMMs were used to map the hand shape parameter sequence to the hand motion parameter sequence of gestures in the presence of normalized time as input, and MLPs were used as local models. The effect of the normalized input was discussed and shown to be an effective method to increase the recognition rate and to convert the local models to state space models. An adaptive threshold model for IOHMMs has not been proposed before. It is shown that an IOHMM with a single state can be used to distinguish defined gestures.

The system was able to attain very high recognition rates and proved to be a very promising method for hand gesture recognition problem.

5.1. Remarks and Future Work

The use of colored gloves is obtrusive and it is desirable to implement an approach that can localize the hand in real time without the need for such markers. Particle filters are a promising approach and will be implemented to test their real time capabilities.

For an HMM-based framework with distinct observable symbols it is discussed that the evaluation of adaptive threshold models proposed by Lee and Kim tends to choose the pure hidden states. Exploiting this observation could possibly reduce the computationally intensive evaluation process to a simple formula. This approach will be implemented and tested. If the observation is valid, this will allow the real time system to handle a serious bottleneck.

The choice of an input sequence for the IOHMMs was thoroughly discussed and

it was suggested that the same architecture could be used to map the motion vectors to hand shape features. IOHMMs are very flexible in the sense that they allow the output of a function of a segment of the output sequence to be an input. This flexibility is not fully explored in the current proposed system. In particular all of the observables could be used as a continuous output vector, and a segment of this sequence could be used as input.

The current gesture dataset consists of samples that have automatically been extracted from continuous streams, which is a process prone to segmentation error and noise. In fact, the main reason for the recognition error is expected to be caused by this kind of error. Therefore a large database of gestures will be created for testing purposes, using a method robust to noise and segmentation error.

APPENDIX A: APPLICATION

A.1. Introduction

To test the real time capabilities of the proposed system, a PC application is developed that aims to provide a gestural interface for generic applications. The purpose of the interface is:

- to track the 3D hand motion and simulate the 2D mouse motion accordingly,
- to let the user define new gestures and train the system,
- to distinguish the defined communicative gestures from the manipulative gestures or unintentional hand motions in real time,
- to fire user–defined low level operating system (OS) events upon recognition of gestures, which can be sequences of mouse motion, mouse clicks and keyboard input,
- to provide the necessary tools for system calibration.

Hence, the interface may replace mouse and keyboard by imitating their input as understood by the OS. In particular manipulative gestures are used to move the mouse pointer or simulate mouse clicks, whereas the communicative gestures are used to give complicated commands to the system.

Target applications may be specifically designed for the interface to make optimum use of it, or they may be unaware of the existence of the interface. The former kind of applications may be games designed for a gestural interface, sign language recognition or tutoring systems or any system that makes use of the complex information provided by hand shape and motion. The latter kind of applications include third party applications that handle mouse and keyboard input via OS messages. The

applications that are designed to manipulate 3D information, such as 3D modeling tools or virtual reality applications are expected to benefit from a gestural interface, which is capable of handling a higher DOF.

A.1.1. Requirements

The proposed system uses simple webcams to track the 3D motion of the hand. Hence, the system must be capable of synchronously handling two cameras at the same time in order to perceive the depth information. This imposes requirements on both the mainboard and the CPU of the system. Namely, the mainboard bus must be capable of transferring the image information coming from the cameras, and the CPU must be powerful enough to process 60 images per second in the background while executing the target application. All newer mainboards and CPUs can efficiently handle the load generated by the gestural interface.

The interface assumes RGB values when processing the image, and hence, the cameras must be capable of acquiring the images in RGB format. Since the 3D trajectory is computed using 3D reconstruction techniques, the system works with simple cameras instead of special stereo cameras. In particular the system can identify any camera that has WDM or VFW drivers.

The frame rate of the cameras is the current bottleneck of the application. Nevertheless, 30 fps per camera has been found to be enough for smooth operation.

A.2. Main Phases

A.2.1. System Setup

Prior to usage, several tasks should be completed for the interface to run:

- The cameras should be connected and configured
- Camera setup needs to be determined
- The cameras should be calibrated
- A suitable marker must be chosen

The interface allows configuration of the cameras from inside the application using the VFW function calls. Since the hand tracking mechanism is color–based, the colors should be made as distinct as possible in the configuration phase to ensure robust tracking.

The precision of the 3D reconstruction phase is dependent on the camera setup. Therefore, there are several considerations:

- Due to the nature of the interface the cameras need to face the user
- The views of the scene as acquired by the cameras should be as distinct as possible
- The space where the user performs the gestures should be viewable by each camera

In a home environment it is usually enough to put the cameras to each side of the monitor and make them face the user.

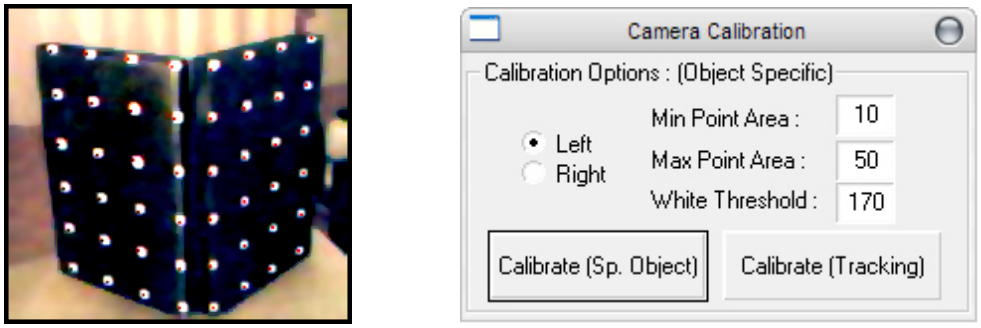


Figure A.1. Camera setup for Hand Gesture Interface

Once the camera setup is determined, the cameras should be calibrated using the tool provided in the interface. The calibration process consists of the following phases:

- Prepare the calibration object
- Make sure that both camera views capture the object in its entirety
- Click the calibrate button

The output of the calibration process is the calibration matrix of each camera. The matrices are written in text files and therefore they can be estimated using other external calibration tools and manually prepared by the user.



(a) Calibration object (b) Calibration interface

Figure A.2. Camera calibration using the internal tool

The final step prior to the execution of the interface is choosing a colored glove. The chosen color cannot be grayscale and it should be distinguishable from the skin color as explained in Chapter 3.

A.2.2. Preparation of a Gesture Database

Completing the tasks given in the previous section enables the interface to track and translate the hand motion into mouse input through manipulative gestures. In order for the interface to make use of communicative gestures, a gesture database should be created. The interface provides the necessary tool to train new gestures.

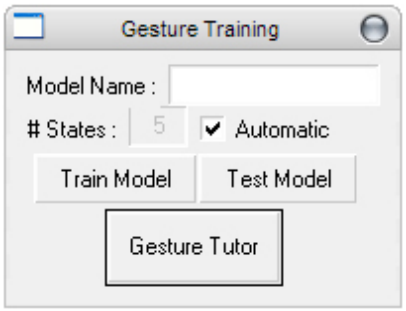


Figure A.3. Training interface

The recognized communicative gestures are translated into commands for the target application. The binding of commands to gestures is done using configuration files that have a very simple scripting language, which can handle firing of sequences of mouse and keyboard events.

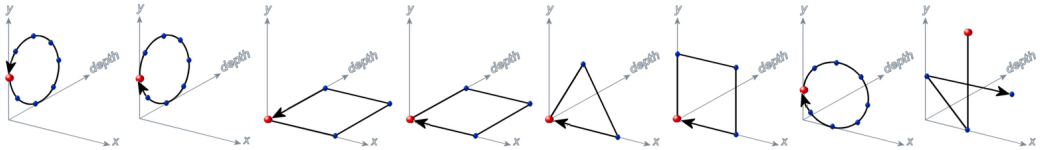


Figure A.4. Possible gestures

A.2.3. Execution

Once the system setup is complete and the gesture database with corresponding command lists exist, the user can start the interface. The initialization phase of the interface consists of the detection of the cameras and preloading of the calibration matrices, gesture models and command lists. If the system detects a single camera, the interface is forced to work in 2D mode. Figure A.5 shows the main graphical user interface (GUI).

The user can start the tracking of the hand by pushing the initially disabled Start button. To enable the Start button the marker registration process should be completed, which is dependent on the lighting conditions and should be went through

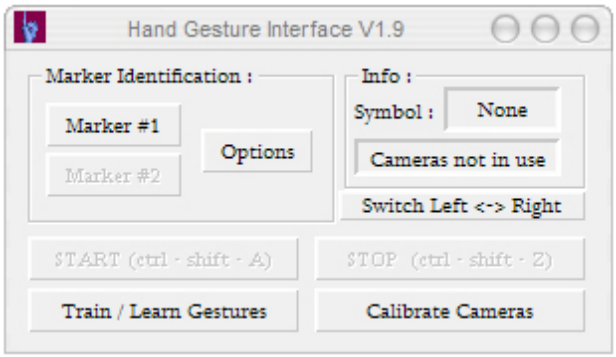


Figure A.5. Training interface

each time the application starts. Users can register the marker by waving their hand and pushing the button labeled as Marker#1. The interface automatically detects and identifies the marker color using the technique explained in Chapter 3. Alternatively users may want to use both hands for the interface. This is an experimental option and the secondary hand is currently not used in the recognition phase. The registration of the second marker can be initiated by pushing the button labeled Marker#2, which is enabled only if "Two hands mode" option is checked in the Options screen. The options screen can be seen in Figure A.6.

The Options screen allows the user to change the modes of operation as well as the parameters for marker registration and gesture recognition. The mode related options include:

- Selection of the primary hand
- 2D/3D modes
- Using both hands
- Filtering the trajectory
- Keeping a log
- Mouse click on no hand motion for three seconds

When the marker is registered the Start button is enabled and the user can start

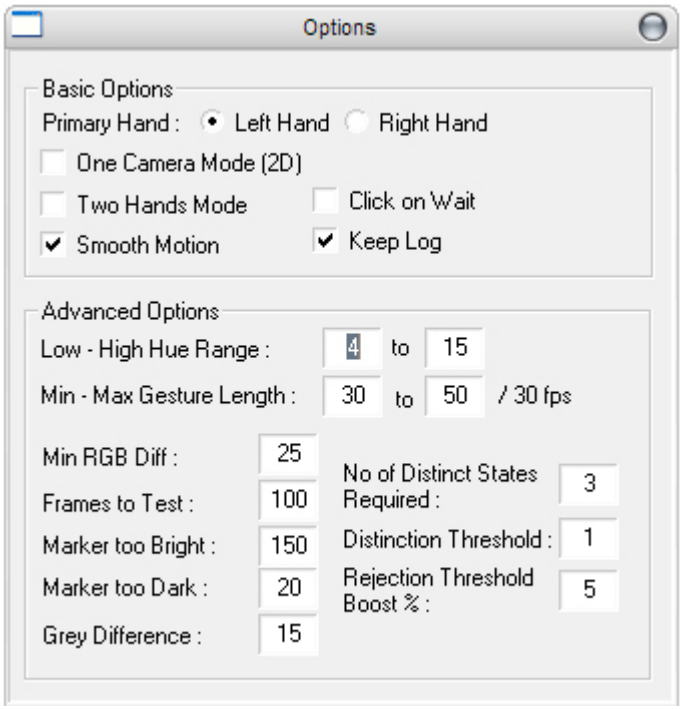


Figure A.6. Options interface

controlling the target application. The interface runs in background and sends the events indirectly to the target application through OS message queue. This system is tested on several third party applications such as photo editing tools and games and is demonstrated in [63].

REFERENCES

1. Baudel, T. and M. Beaudouin-Lafon, "Charade: Remote Control of Objects using Freehand Gestures", *Communications of the ACM*, vol. 36, no. 7, pp. 28–35, 1993.

2. Gupta, P., S. Mittal, R. Dutta, S. Chaudhury and S. Banerjee, "Developing a Gesture-based Interface", *IETE Journal of Research: Special Issue on Visual Media Processing*, 2002.

3. Freeman, W. and C. Weissman, "Television Control by Hand Gestures", *International Workshop on Automatic Face and Gesture Recognition*, pp. 179–183, 1995.

4. Zhang, Z., "Vision-based Interaction with Fingers and Papers", *International Symposium on the CREST Digital Archiving Project*, pp. 83–106, 2003.

5. Yee, K. P., "Two-handed Interaction on a Tablet Display", *Extended Abstracts of ACM CHI*, pp. 1493–1496, 2004.

6. Heckenberg, D. and B. C. Lovell, "MIME: A Gesture-Driven Computer Interface", *Visual Communications and Image Processing*, vol. 4067, pp. 261-268, 2000.

7. Le Gal, C., J. Martin, A. Lux and J. L. Crowley, "Smartoffice: Design of an Intelligent Environment", *IEEE Intelligent Systems*, vol. 16, pp. 60–66, 2001.

8. Keskin, C., A. N. Erkan and L. Akarun, "Real–Time Hand Tracking and 3D Gesture Recognition for Interactive Interfaces using HMM", *Proceedings of International Conference on Artificial Neural Networks*, 2003.

9. Ahmad, S., "A Usable Real–Time 3D Hand Tracker", *Proceedings 28th Asilomar Conference on Signals, Systems and Computers*, pp. 1257–1261. IEEE Computer Society Press, 1995.

10. Su, S.A. and R. Furuta, "Virtual Panel Architecture: a 3D Gesture Framework", *IEEE Annual Virtual Reality International Symposium*, pp. 387–393, 1993.

11. Zhang, Z., Y. Wu, Y. Shan and S. Shafer, "Visual Panel: Virtual Mouse Keyboard and 3D Controller with an Ordinary Piece of Paper", *Workshop on Perceptive User Interfaces*, 2001.

12. Codella, C., R. Jalili, L. Koved, J. B. Lewis, D. T. Ling, J. S. Lipscomb, D. A. Rabenhorst, C. P. Wang, A. Norton, P. Sweeney and G. Turk, "Interactive Simulation in a Multi-Person Virtual World", *Computer Human Interaction*, 1992.

13. Maggioni, C., "A Novel Gestural Input Device for Virtual Reality", *IEEE Annual Virtual Reality International Symposium*, pp. 118–124, 1993.

14. Kuhlen, T. and C. Dohle, "Virtual Reality for Physically Disabled People", *Computers in Biology and Medicine*, vol. 25, pp. 205–211, 1995.

15. Triesch, J. and C. Von der Malsburg, "A Gesture Interface for Human Robot Interaction", *IEEE Automatic Face and Gesture Recognition*, pp. 546–551, 1998.

16. Becker, M., E. Kefalea, E. Mael, C. von der Malsburg, M. Pagel, J. Triesch, J. C. Vorbruggen, R. P. Wurtz and S. Zadel, "GripSee: A Gesture Controlled Robot for Object Perception and Manipulation", *Autonomous Robots*, 1998.

17. Kortenkamp, D., E. Huber and R. P. Bonasso, "Recognizing and Interpreting Gestures on a Mobile Robot", *American Association for Artificial Intelligence*,

1996.

18. Ehrenmann, M., T. Lutticke and R. Dillmann, "Dynamic Gestures as an Input Device for Directing a Mobile Platform", *Proceedings IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2596–2601, 2001.

19. Oka, K., Y.Sato and H. Koike, "Real–Time Fingertip Tracking and Gesture Recognition", *IEEE Computer Graphics and Applications*, pp. 64–71, November-December, 2002.

20. Starner, T., B. Leibe, D. Minnen, T. Westyn, A. Hurst and J. Weeks, "The Perceptive Workbench: Computer Vision Based Gesture Tracking, Object Tracking, and 3D Reconstruction of Augmented Desks", *Machine Vision and Applications* vol. 14, pp. 59-71, 2003.

21. Starner, T. and A. Pentland, "Real–Time American Sign Language Recognition from Video using Hidden Markov Models", Technical Report, MIT Media Laboratory, 1996.

22. Starner, T., J. Weaver and A. Pentland, "Real–Time American Sign Language Recognition using Desk and Wearable Computer Based Video", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 12, pp. 1371–1375, December, 1998.

23. Poizner, H., U. Bellugi and V. Lutes-Driscoll, "Perception of American Sign Language in Dynamic Pointlight Displays", *Journal of Experimental Psychology: Human Perception and Performance*, vol. 7, pp. 430–440, 1981.

24. Sperling, G., M. Landy, Y. Cohen and M. Pavel, "Intelligible Encoding of ASL Image Sequences at Extremely Low Information Rates", *Computer Vision, Graphics and Image Processing*, vol. 31, pp. 335–391, 1985.

25. Tamura, S. and S. Kawasaki, "Recognition of Sign Language Motion Images", *Pattern Recognition*, vol. 21, pp. 343–353, 1988.

26. Vogler, G. and D. Metaxas, "Adapting Hidden Markov Models for ASL Recognition by using Three-dimensional Computer Vision Methods", *Conference on Systems, Man and Cybernetics*, pp. 156-161, 1997.

27. Vogler, C. and D. Metaxas, "ASL Recognition Based on a Coupling Between HMMs and 3D Motion Analysis", *International Conference on Computer Vision*, 1998.

28. Vamplew, P. and A. Adams, "Recognition of Sign Language Gestures using Neural Networks", *Australian Journal of Intelligent Information Processing Systems*, vol. 5, no. 2, pp. 94–102, 1998.

29. Fels, S. S. and G. E. Hinton, "Glove-Talk II: A Neural Network Interface which Maps Gestures to Parallel Formant Speech Synthesizer Controls", *IEEE Transactions on Neural Networks*, vol. 9, no. 1, pp. 205–212, 1998.

30. Perng, J., B. Fisher, S. Hollar and K.S.J. Pister, "Acceleration Sensing Glove (ASG)", *ISWC International Symposium on Wearable Computers*, 1999.

31. Rekimoto, J., "SmartSkin: An Infrastructure for Freehand Manipulation on Interactive Surfaces", *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 113–120, 2002.

32. Grimes, G., "Digital Data Entry Glove Interface Device", Bell Telephone Laboratories, Murray Hill, New Jersey. US Patent Number 4.414.537, 1983.

33. Harling, P. A., *"Gesture Input using Neural Networks"*, BSc. Project, Department of Computer Science, University of York, York, YO1 5DD, UK, 1993.

34. Pavlovic, V. I., R. Sharma and T. S. Huang, "Visual Interpretation of Hand Gestures for Human Computer Interaction: A Review", *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, 1997.

35. Rosales, R., V. Athitsos, L. Sigal and S. Sclaros, "3D Hand Pose Reconstruction using Specialized Mappings", *Proceedings of IEEE ICCV*, 2001.

36. Bobick, A. F. and J. W. Davis, "Real–Time Recognition of Activity using Temporal Templates,", *Proceedings of International Conference on Automatic Face and Gesture Recognition*, 1996.

37. Turk, M. and A. Pentland, "Eigenfaces for Recognition", *Journal of Cognitive Neuroscience*, vol. 3, pp. 71–86, 1991.

38. Black, M. J. and A. D. Jepson, "Eigentracking: Robust Matching and Tracking of Articulated Objects using a View-Based Representation", *European Conference on Computer Vision*, pp. 329–342, 1996.

39. Cui, Y. and J. Weng, "Learning-Based Hand Sign Recognition", *Proceedings of International Workshop on Automatic Face and Gesture Recognition*, pp. 201-206, 1995.

40. Moghaddam, B. and A. Pentland, "Maximum Likelihood Detection of Faces and Hands", *Proceedings of International Workshop on Automatic Face and Gesture Recognition*, pp. 122-128, 1995.

41. Erkan, A. N., *"Model Based Three Dimensional Hand Posture Recognition for Hand Tracking"*, Master's Thesis, Boğaziçi University, 2004.

42. Comaniciu, D. and P. Meer, "Mean Shift: A Robust Approach Toward Feature Space Analysis", *IEEE Transactions On PAMI*, vol. 24, no. 5, pp. 603–619, 2002.

43. Zarit, B. D., B. J. Super and F. K. H. Quek, "Comparison of Five Color Models in Skin Pixel Classification", *International Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real–Time Systems*, pp. 58-63, 1999.

44. Martinkauppi, J. B., *"Face Color Under Varying Illumination - Analysis and Applications"*, PhD thesis, Department of Electrical and Information Engineering, University of Oulu, Oulu, Finland, 2002.

45. Carpenter, J., P. Clifford and P. Fearnhead, "Improved Particle Filter for Nonlinear Problems", *IEE Proceedings - Radar, Sonar and Navigation*, vol. 146, Issue 1, pp. 2–7, 1999.

46. Cipolla, R., Y. Okamoto and Y. Kuno, "Robust Structure From Motion using Motion Parallax", *Proceedings of IEEE International Conference on Computer Vision*, pp. 374–382, 1993.

47. Kuch, J. J., *"Vision-Based Hand Modeling and Gesture Recognition for Human Computer Interaction"*, Masters thesis, University of Illinois at Urbana-Champaign, 1994.

48. Quek, F. K. H., "Eyes in the Interface", *Image and Vision Computing*, vol. 13, 1995.

49. Darrell, T., I. Essa and A. Pentland, "Task-Specific Gesture Analysis in Real–Time using Interpolated Views", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 12, pp. 1236–1242, 1996.

50. Pavlovic, V. I., R. Sharma and T. S. Huang, "Gestural Interface to a Visual Computing Environment for Molecular Biologists", *Proceedings of International Conference on Automatic Face and Gesture Recognition*, pp. 30–35, 1996.

51. Hu, M. K., "Visual Pattern Recognition by Moment Invariants", *IRE Transactions on Information Theory*, vol. 8, pp. 179–187, 1962.

52. Kalman, R., "A New Approach to Linear Filtering and Prediction Problems", *Transactions of the ASME - Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.

53. Cutler, R. and M. Turk, "View-Based Interpretation of Real–Time Optical Flow for Gesture Recognition", *Proceedings of the Third IEEE Conference on Face and Gesture Recognition*, 1998.

54. Lee, H. and J. H. Kim, "An HMM-Based Threshold Model Approach for Gesture Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 10, pp. 961–973, 1999.

55. Rabiner, L. and B. Juang, "An Introduction to Hidden Markov Models", *IEEE ASSP Magazine*, pp. 4–16, 1996.

56. L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedings of IEEE*, vol. 77, pp. 257–286, 1989.

57. Bengio, Y. and P. Frasconi, "Input/Output HMMs for Sequence Processing", *IEEE Transactions on Neural Networks* vol. 7, pp. 1231–1249, 1996.

58. Just, A., O. Bernier and S. Marcel, "HMM and IOHMM for the Recognition of Mono- and Bi-Manual 3D Hand Gestures", *Proceedings of the British Machine Vision Conference*, 2004.

59. Böhm K., W. Broll and M. Sokolewicz, "Dynamic Gesture Recognition using Neural Networks: A Fundament for Advanced Interaction Construction", *SPIE Conference on Electronic Imaging: Science & Technology*, vol. 2177, pp. 336–346,

1994.

60. Murakami, K. and H. Taguchi, "Gesture Recognition using Recurrent Neural Networks", *CHI Conference Proceedings*, pp. 237–241, 1991.

61. Yang, M., N. Ahuja and M. Tabb, "Extraction of 2D Motion Trajectories and its Application to Hand Gesture Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 1061–1074, 2002.

62. Lee, H. K. and J. H. Kim, "Gesture Spotting from Continuous Hand Motion", *Pattern Recognition Letters*, vol. 19, pp. 513-520, 1998.

63. Keskin, C., O. Aran and L. Akarun, "Real–Time Gestural Interface for Generic Applications", *European Signal Processing Conference, EUSIPCO Demonstration Session*, 2005.