Ayman El-Sawah
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Ph.D. (Computer Science)
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Towards Context-aware Gesture Enables User Interfaces

TITRE DE LA THÈSE / TITLE OF THESIS

Nicolas Georganas
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

Emil Petriu
CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Alexandra Brantan Albu	Wonsook Lee
Abdulmotaled El-Saddik	Dorina Petriu

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Towards Context-Aware Gesture Enabled User Interfaces

A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

in

Computer Science

by

Ayman El-Sawah

Ottawa-Carleton Institute of Computer Science
School of Information Technology and Engineering
University of Ottawa

Your file Votre référence
ISBN: 978-0-494-41631-0
Our file Notre référence
ISBN: 978-0-494-41631-0

To my wife and my mother, as a sign of gratitude for their love and inspiration for me and our family.

Acknowledgements

This effort would not be completed had it not been for the direct and indirect help of many, many people that shared their love, inspiration, time, advice, and many other essentials with me. It would fill countless more pages than the technical content of this dissertation to name all of them and their contributions. However, I would like to thank in particular:

Professor Nicholas D. Georganas for the opportunities he provided me during my studies and the freedom he provided me. Mostly for being the best role model who inspires everyone and gives hope and confidence to everyone who deals with him. I would like to thank Professor Emil M. Petriu for his technical guidance and support during this study. He challenged me to make a turn in this dissertation that turned out to be one of the best moves I made, and I received two awards because of that move. I would like also to thank Dr. Chris Joslin for his role at the beginning of this dissertation; Dr. WonSook Lee for her kind comments about the research and inspiring me to explore the limit of my model; Prof. Dorina Petriu for her advice to put the contribution upfront in this dissertation.

My deep and sincere gratitude goes to:

My wife, whom her love and support ever since we were married helped me carry on with this effort. I would not be able to complete this effort if not she believed in me and the benefit of high education. My mother for inspiring me and educating me the value of education and for challenging me with math races when I was in the elementary school. It deepened my love to mathematics and it became my best essence in research. For all the school teachers and university professors who taught me in my school and undergraduate studies. For Professor Mohamed Kamel, who supervised me in my Masters, and continued advising me even after I graduated. For François Marlic, who provided support with computer problems in the Lab, and for friendly discussions and advice. For all the members of the DISCOVER LAB, who are so friendly and technically inspiring.

Acronyms

ASL	American Sign Language
HMM	Hidden Markov Model
P2DHMM	Pseudo 2D HMM
LLE	Local Linear Embedding Algorithm
GA	Genetic Algorithm
SA	Simulated Annealing
LMS	Least Median Square Algorithm
OSG	Open Scene Graph
IK	Inverse Kinematics
MCP	Metacarpophalangeal joint
PIP	Proximal Interphalangial joint
DIP	Distal Interphalangial joint
CMC	Carpometacarpal joint
IP	Interphalangial joint
RBF	Radial Basis Function
TDRBF	Time-delay Radial Basis Function
DBN	Dynamic Bayesian Network
SVM	Support Vector Machines
FCM	Fuzzy C-Means clustering
PCA	Principal Component Analysis

Abstract

Conventional graphical user interface techniques appear to be ill-suited for the kinds of interactive platforms that are required for future generations of computing devices. 3D graphics and immersive virtual reality applications require interactive 3D object manipulation and navigation. Perceptual user interfaces using speech and gestures are in high demand to provide a more natural human-computer interaction modality. The major challenge facing Perceptual user interfaces is the lack of a standard application programming interfaces capable of handling ambiguity and providing the means to include domain-specific knowledge about the context in which the user interface is used.

In this dissertation, we study dynamic hand gestures, which are defined as a sequence of hand postures. We emphasize the generality of our dynamic gesture model, which is capable of recognizing essentially any dynamic hand gesture confined in a sequence of postures. Hand postures are static poses and are defined by an array of posture attributes. We use a generic definition hand postures capable of covering the space of hand postures at different levels of granularity and abstraction; and we timely monitor the posture variation as it unfolds within the dynamic gesture.

We also study the role of context in gesture interpretation without making assumptions about a specific application. We view the hand-tracking and gesture-recognition subsystems as integral parts of a larger distributed and multi-user multi-service application, where gesture interpretation plays the role of resolving ambiguity of the recognized gesture. We identify the relevant aspects to hand gesture interpretation and we propose agent-based system architecture for gesture interpretation.

We finally propose a framework for gesture-enabled system design, where context is placed in a middleware layer that interfaces with all sub modules in the system and plays a dialectic role and keeping the overall system stable.

Contents

List of Figures

List of Tables

Chapter 1

1. Introduction

Conventional graphical user interface techniques appear to be ill-suited for the kinds of interactive platforms that are required for future generations of computing devices. Ubiquitous computing devices having very small or very large displays are breaking out of the desktop box and motivate an increasing diversity of user interfaces. Examples of such devices include: personal digital assistants, cell phones, pagers, computerized pens, computerized notepads, and various kinds of desk and wall-size computers, as well as devices in everyday objects such as mounted on refrigerators, or even embedded in vehicles' dash boards. Moreover, 3D graphics and immersive virtual reality applications require interactive 3D object manipulation and navigation. Perceptual user interfaces using speech and gestures are in high demand to provide a more natural human-computer interaction modality. The major challenge facing Perceptual user interfaces is, in our opinion, the lack of a standard application programming interface capable of handling ambiguity, which is the ability to process incomplete or undetermined data, and providing the means to include domain-specific knowledge about the context in which the user interface is used. In this dissertation we present a framework for hand tracking and gesture recognition suitable for generic perceptual user interface applications. Figure 1 shows an overview of the framework.

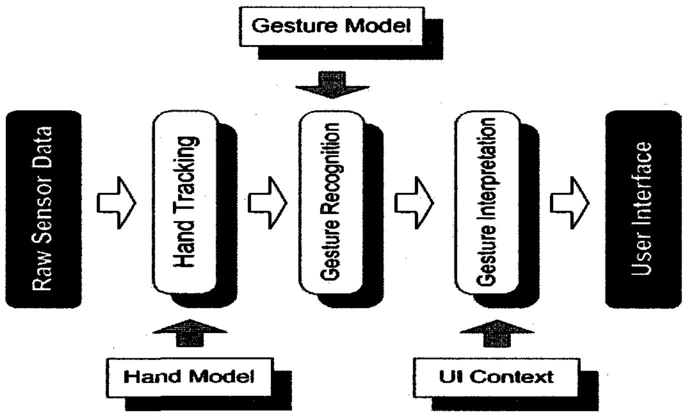


Figure 1: Overview of the Gesture Information Flow in a Perceptual User Interface

Raw sensor data, in the form of sensor readings or extracted image features in the case of CyberGlove and vision-based tracking systems, is used to acquire 3D hand postures in the hand tracking module. The variation of the hand posture over time is used to recognize dynamic gestures. The recognized gestures are then interpreted according to the user's context and are delivered to the user interface to perform the required action and display the proper feedback.

Hand gesture recognition lets humans use their most versatile instrument, their hands, in more natural and effective ways than currently possible. The de facto standard in hand tracking and posture detection is the data glove. This device is capable of accurately recording finger joint motions via flex sensors in a tightly fitting glove, but it is usually cumbersome and expensive for most commercial applications. Computer vision based hand gesture recognition has the potential of low prices and more flexibility. The major potential of vision-based gesture recognition, in our opinion, is that it integrates naturally with the environment in which it is used. In other words, it can act as a rich resource of context information, beyond the task of hand tracking and gesture recognition. For example a vision-based perceptual user interface can identify the face of the gesture operator and the objects in his vicinity and consequently loads his gesture models and identifies the available services, which can be effective in the gesture interpretation.

While most hand gesture recognition research focus on recognizing a set of predefined postures, in this dissertation we study generic dynamic gestures. We acknowledge that gestures, as any form of communication, is evolving by nature. Thus, an efficient hand recognition system must be able to easily modify the gesture set by adding, removing or regrouping its elements. This implies that we: firstly have to use a generic definition of hand postures, which is capable of covering the space of hand postures at different levels of granularity and abstraction; and, secondly, we should be able to timely monitor the posture variation as it unfolds within the dynamic gesture, which means that the time must be explicitly visible in the gesture models used for gesture recognition. For that reason, we emphasize the fingers posture by using the fingers joint angles as the main descriptor of the hand posture, and then derive more meaningful attributes from joint angles to abstract their correlation and natural constraints. We also propose a dynamic gesture model based on sampling the high dimensional posture attributes at a constant frequency and modeling the posture variation using a dynamic Bayesian network.

We also study the role of context in gesture interpretation without making assumptions about a specific application. We view the hand tracking and gesture recognition subsystem as an integral part of a larger distributed and multi-user multi-service application, where gesture interpretation plays the role of resolving ambiguity of the recognized gesture. We identify, in the most generic form, the relevant aspects to hand gesture interpretation and we propose an agent based system architecture, which is taking all the relevant aspects of gesture interpretation into consideration. We thus propose a framework for gesture-enabled system design, where context is placed in a middleware layer that interfaces with all sub modules in the system and plays a dialectic role and keeping the overall system stable. By a dialectic role we mean that the context information is utilized by all the different subsystems to increase its performance and the output of all subsystems is used to modify the context.

1.1. Thesis Statement

This dissertation introduces and evaluates an integral framework for 3D hand tracking and dynamic gesture recognition. It demonstrates that computer vision is a feasible means to provide hand gesture interfaces. We take a pragmatic approach to vision-based hand tracking by allowing the human operator to wear a glove with fiducial markers, which are used to track the geometry of the hand. The dissertation serves as a proof of concept of the first prototype and provides recommendations for future generations of the fiducial glove. 3D computer games and immersive virtual environments really benefit from such an invention. By the reduced cost and the wide spread of web cameras and the progressive performance of computing devices, an efficient vision-based hand tracking subsystem can become a versatile technique for human-computer interface.

In comparison with the data glove, computer vision provides *less accurate* posture estimation because of the ambiguity inherent in the scene projection on the 2D image plane and due to discontinuity of the scene due to object occlusion. The answer to the accuracy problem, in our opinion is to use redundant fiducials and to utilize statistical priors acquired from the environment in the form of context information.

The remainder of this chapter motivates the problem setting even further, states our main contributions to overcome those problems, and finally provides an overview of the dissertation organization

1.2. Problem Statement

Perceptual user interfaces are an emerging technology that is gaining interest in the academic and industrial communities alike. Hand gesture recognition is one the most versatile modes of human communication but has not been exploited fully in the human computer interaction domain. The challenges facing the technology are diversified among all its aspects including hand tracking and posture acquisition, gesture recognition, gesture interpretation and the application design and integration process. In the following sections we describe the sub-problems we are addressing and objectives we are trying to reach in this dissertation in the different aspects of designing a context-aware gesture-enabled ubiquitous system.

1.2.1. Hand Tracking

Hand tracking is the foundation on which hand gesture recognition applications are built. The output of the hand tracking module, in terms of the type of attributes and their accuracy, directly affects the performance of the gesture recognition subsystem. We focus primarily on finger postures combined with 3D hand position and orientation. The de facto standard in hand tracking is the data glove and position sensors, but we acknowledge the potential of computer vision based hand tracking as an emerging rival for the data glove. To the best of our knowledge there are no available benchmarks to measure the accuracy of the approach in comparison with the data glove. Such benchmarks are essential for system designers of Perceptual user interfaces in utilizing vision-based tracking and gesture recognition.

Our objective is to be able to continuously track the hand in real time in a non constrained environment in terms of lighting and background, and to provide the hand posture in terms of a vector of attributes defining the hand position and orientation in 3D and the fingers joint angles. For that reason, our system undergoes a 2D/3D conversion where the detected 2D features of the hand are used to estimate the 3D hand posture as well as position and orientation. This transformation is ambiguous due to the loss of depth information in the projection process, and for that reason we use visual cues to estimate the hand posture. The 2D/3D conversion increases the complexity of the system, but its advantage lies in the flexibility of the system to recognize generic gestures and easily building up the gesture set without requiring major system changes. By providing posture description in 3D, the posture is viewpoint-independent and thus its model is simplified. Stereo vision can be used to acquire 3D information, but we are primarily interested in a single camera system, motivated by cheaper prices of the camera and we also believe that with adequate modeling and utilization of hand constraints and studying the motion priors we can achieve proper estimation of the hand posture using a single camera.

1.2.2. Dynamic Hand Gesture Recognition

Gesture and posture recognition have been studied by researchers in the last 20 years with variant degrees of success. In most cases, gestures are confined to a discrete set of postures recognized in static form. We would like to emphasize the dynamic aspect of the hand gestures, and be able to distinguish between different gestures that vary solely in their timing aspect. The problem of creating a generative model, capable of generalizing the training data to recognize untrained data, is usually based on assumptions about the nature of the process we are trying to recognize. Hidden Markov Models, the most commonly used approach in hand gesture recognition, makes the assumption that there is a hidden state causing the observed postures, and that the transition between such hidden states are defined by a static stochastic state transition matrix. The physical meaning of the hidden states and the number of such states is not defined in the HMM framework and is either determined by trial and error or from experience of the system designer. We would like to design a more flexible model, where the hidden states causing the observation are in fact the dynamic gestures we are trying to recognize. In this way, there will not be ambiguity about the number of states. We also would like to relax the assumption that the state transition between hidden states is governed by a static state transition stochastic matrix, because in fact gestures are emergent and situation bound depending on the context of the situation. We propose a novel dynamic gesture model based on dynamic Bayesian networks.

Our objective is to be able to recognize dynamic gestures in real time. We emphasize the system's latency, which is the time spent between the end of the time the final posture is recognized and the time the dynamic gesture is recognized. Hidden Markov Models require either Viterbi algorithm or forward-backward algorithm for recognition, which consumes time. A more direct model is required to minimize the overall system latency and enhance the system usability.

1.2.3. Context Awareness

Context-aware systems are an emergent technology branching from ubiquitous and mobile computing. The role of context in gesture recognition and interpretation have not been addressed by researchers in the field of hand gesture recognition, but have been studied in psychology, linguistics, anthropology, ethno methodology, discourse analysis, and conversation analysis. We would like to transfer this knowledge to the information technology domain and propose a framework for context-aware gesture-enabled system design.

Context is defined as any information that can be used to characterize the situation of an entity, where an entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves. A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task. The relevant information to a gesture enabled user interface is the information used in posture acquisition such as the hand model, the information used in gesture recognition such as the gesture models, and the information used in gesture interpretation such as the operator identification, the location, the available services or applications, which are supported by the gesture recognition subsystem.

We divide the context information into two types, non-volatile context that represent the essential information that must be provided to the system to achieve its basic tasks, such as the hand model, the gesture models, and low level features required to estimate the hand posture. Volatile context, on the other hand is ambiguous pieces of information that can be used to improve the performance of the system, such as visual artifacts, recognized people in the neighborhood, the location, the running applications. This information can be used to modify the probabilistic priors of the gesture set and gesture interpretation. Our framework utilizes stochastic, fuzzy, and agent based approach to be able to utilize the ambiguous information of the context.

1.3. Key Contributions

The key contributions of this dissertation are distributed amongst the hand gesture recognition system design aspects discussed in the problem statement.

- The first contribution is that it provides a *framework to model and recognize generic dynamic gestures in real time.* The method does not contain hidden structure and thus does not require a domain expert to configure the gesture classifier, but only require a segmented example of the dynamic gestures to be recognized, defined in the form of a sequence of hand postures sampled at constant frequency. The dynamic gestures are modeled using a dynamic Bayesian network, which is used to provide a probabilistic measure of the gesture, given the observed posture in real time with minimum latency.

- The second contribution of this dissertation is that it *presents the 3D vision-based hand tracking and posture estimation as a nonlinear optimization problem*, where the search space is the high dimension hand articulation (finger joints) and the objective function is a probabilistic observation model derived from generic features in the acquired video. The search is guided by domain specific features on the hand defined by colored markers. Other means of guiding the posture space are surveyed but, to the best of our knowledge, do not provide real time hand tracking. We then analyze the accuracy of the vision tracking approach in comparison with the de facto standard data glove and provide a valuable benchmark for vision-based hand tracking. *We also qualify the posture attributes that can be accurately estimated using computer vision* and conclude that finger-level posture attributes such as curvature provide better accuracy than sub-finger level attributes, such as the joint angles.

- The third contribution of this dissertation is a *novel method to solve complex and slightly under constrained inverse kinematics problems using error model analysis* of a rough estimate derived from a simpler and directly related inverse kinematics problem. This method is demonstrated to effectively solve the finger four degrees of freedom inverse kinematics, which is slightly under constrained given the 3D position of the fingertip. First, the first two degrees of freedom are solved using the error model of the simplified two degrees of freedom problem assuming the finger is not curved, and then the remaining two degrees of freedom are solved given the first two in the same way.

- Finally, the dissertation *describes the role of context in vision-based hand tracking and gesture recognition*, which is in our opinion the main advantage of the vision approach over the classical exoskeleton and data glove approaches. It concludes that *context must not be localized at a particular stage in the system, but should act a dialectic role in all aspects of the system.* An agent based context-aware gesture interpretation system design is proposed and middleware context layer is recommended for future context-aware systems design. For example in a gesture-enabled user interface, the identification of the active operator can be utilized by the hand tracking subsystem by activating the hand model associated with the current operator, on the same token, the gesture recognition subsystem can activate the gesture models associated with the current operator. All other operator related context such as preference roles played by the operator are loaded and are continuously updated at the same time.

1.4. Publications resulting from this thesis

- A. El-Sawah, C. Joslin, N.D. Georganas, E.M. Petriu, *A Framework for 3D Hand Tracking and Gesture Recognition using Elements of Genetic Programming*, Proceedings of International Workshop on Video Recognition, VideoRec`07, Montreal, May 2007 - **Best Paper Award**

- A. El-Sawah N.D. Georganas, E.M. Petriu, *Calibration and Error Model Analysis of 3D Monocular Vision Model Based Hand Posture Estimation*, IEEE Instrumentation and Measurement Technology Conference, IMTC07, Warsaw, Poland, May 2007 - **Travel Award**

- A. El-Sawah, N.D. Georganas, E.M. Petriu, *Finger inverse kinematics using error model analysis for gesture enabled navigation in virtual environments*, IEEE HAVE06, Ottawa, Oct. 2006

- Q. Chen, A.M. Rahman, A. El-Sawah, X. Shen, A. El Saddik and N.D. Georganas, *Accessing Learning Objects in Virtual Environment by Hand Gestures and Voice*, I2LOR-06, 3rd Annual Scientific Conference, LORNET Research Network, TELUQ-UQAM University in Montreal, Canada, Nov. 2006.

- C. Joslin, A. El-Sawah, D. Chen, N.D. Georganas, *Dynamic Gesture Recognition*, Proc. IEEE IMTC, Ottawa, May, 2005, pp 1706-1711

1.5. Dissertation Overview

This dissertation is divided into three main components, Hand Tracking, Dynamic Gesture Recognition and Context-Awareness. In the next chapter we will survey the state of art in the different components of this dissertation. Next are dedicated chapters for Hand Tracking, Dynamic Gesture Recognition and the Role of Context in Hand Gesture Recognition. We will explain our methodology, experimental results and our conclusions for each component. Finally we will make our overall conclusion and will make recommendations for future work.

Chapter 2

2. Literature Review

Aspects related to hand tracking and gesture recognition include the tracking technology, the output of the tracking process, the attributes used in recognition process, the cardinality of the gesture set, the limitation and restriction of tracking environment, and the provision of temporal aspects of gesture. Literature surveys about hand tracking and gesture recognition include [42]-[47].

Ong and Ranganath [42] survey automatic sign language recognition from different aspects including useful applications, relevant aspects such as lexical meaning and grammar, vision-based tracking, feature extraction and parameter estimation, gesture recognition schemes and classification methods, gesture segmentation, grammar processing and signer independence and integration of hand gesture recognition with other modes such as facial expression and speech recognition.

Mitra and Acharya [43] surveyed research in hand gesture recognition and face and head gestures. They surveyed the major methods successfully utilized in gesture recognition, e.g. Hidden Markov Models, Particle Filtering / Condensation, Functional State Machines, and Artificial Neural Networks. Mahmoudi and Parviz [44] survey methods to track visual objects, particularly hands, in video images. They focus on color segment region extraction algorithms such as CamShift and contour based algorithms such as Condensation. Pentland [45] proposes a mathematical model of social signaling from audio and video feeds. His goal is to be able to predict interest, engagement, emphasis from thin slices of conversational and/or gesture-full multimedia clips.

Pavlovic et al. [35] provide a unified approach to modeling, analysis and recognition of hand gestures for visual interpretation, taking into account the characteristics of natural hand gestures. Starting by defining hand gesture representation as a trajectory in the parameter space (posture attributes) over a suitably defined interval (time). Temporarily, gestures undergo three main phases, namely preparation, stroke, and retraction. A set of rules about these phases suggested by Queck ([46]) include:

- Hand posture during the stroke follows a classifiable path in the parameter space

- Gestures are confined to a restrictive spatial volume
- Repetitive hand movements are gestures (beats)
- Manipulative gestures are longer than communicative ones

The above rules remain the key assumptions utilized by many researchers to justify their gesture recognition approaches. Pavlovic classifies the approaches to hand gesture recognition into two main categories, namely *3D model based* and *appearance based* techniques. The 3D model based category provides a complete description of the hand gesture in the parameter and time spaces but it lacks the simplicity and computational efficiency (as per 1997), the appearance based category, on the other hand, is more restrictive in terms of its coverage of the parameter space of gestures, thus provides a simpler alternative.

La Viola Jr. [34] compares between the direct measure device gloves and vision based hand tracking and posture estimation in terms of cost, user comfort, model-to-user adaptation in terms of size and anatomy, calibration, and accuracy. He found the vision based tracking advantageous in the price, comfort and adaptation aspects and lags in the calibration and accuracy aspects. He then survey the algorithmic techniques of recognizing hand postures and gestures from three different aspects, namely feature extraction and modeling, learning and recognition, and other techniques, e.g. linguistic and grammatical approach, appearance based motion, and spatio-temporal analysis. Finally he survey the applications of hand gesture recognition including sign language recognition, gesture-to-speech translation, presentation aids, virtual environment, 3D modeling, multimodal interaction, and robot control.

Hereafter, we survey the latest research in vision based hand tracking, hand gesture recognition and context-aware computing.

2.1. Hand Tracking

Examining the work related to hand tracking, Liu et al. [1] evaluate HMM algorithms for letter hand gesture recognition, i.e. when the letters are traced in the image plane similar to hand writing recognition. They use YUV color space to extract the skin color, morphological operations to filter the image noisy artifacts, a modified CamShift algorithm [2] – a histogram based region segmentation algorithm – for hand extraction, and a proprietary curve smoothing algorithm to extract the letter contours. A gesture letter is defined as a sequence of directional angles which are the observation symbols. Each letter is mapped to one hidden Markov model, where Baum-Welch and Viterbi Path Counting algorithms were used to train the Hidden Markov Models over a range of model structures from Fully-Connected to Left-Right with number of states ranging from 4 to 10. In conclusion Baum-Welch algorithm gives generally better recognition than Viterbi Path Counting and Left-Right topology is better than Fully-Connected, but is more dependent on the number of states. Finally, 9 states give the best results.

Binh et al. [3] recognize single hand gestures of 36 alphanumeric gestures from the American Sign Language (ASL) in real time (25 fps). They use the CamShift algorithm to extract the skin color and a Kalman filter to predict the hand region of interest (ROI). A pseudo 2D Hidden Markov Model (P2DHMM) is used to recognize the 2D gestures. The performance is improved by selecting the distinctive frames from the training sample and using the rest for recognition, and by utilizing a probabilistic model to adaptively filter gesture patterns from non-gesture patterns.

Licsár et al. [4], [5] developed a gesture enabled user interface to control a projector system. A video camera is used to segment the user's arm and hand using background subtraction, the background image is provided from the projector system and undergoes special correction to compensate for projection/lens distortion. Gesture contour is classified by the nearest neighbor rule and the distance metric is defined using a modified Fourier descriptor. 9 gestures were recognized from the projected hand silhouette. The user interface is adapted for multiple users by an online training process.

Utsumi et al. [6] use a view-based appearance model to detect the interaction between hand and object in video frames. The object's appearance model consists of 4 layers per viewpoint; namely texture image, texture reliability map, object/background mask, and the reliability of the foreground/background. The hand blob is extracted based on skin color and motion filter, which is extracted by inter-frame subtraction. Region of interest of objects manipulated by the hand is estimated from the moving region and used to create a multiple view object model by subtracting the hand skin color. A similarity measure is used to compare the observed objects with the observation model database to estimate the object's orientation in 3D – in fact the search is restricted by reducing the dimensionality of the search space. Good estimation of the object's position and orientation is obtained for non symmetric objects, where the appearance varies significantly by changing the orientation, but symmetric objects failed to provide good estimate of the orientation.

Lin et al. [7] developed a hand posture estimation system based on searching the feasible posture set from a lookup table generated using the CyberGlove [8]. The direct search algorithm, namely Nedler-Mead Simplex, starts the search using an arbitrary set of points in the search space and systematically drops the worst point and generates a new one based on a cost/objective function. All points in the simplex are approximated by the closest points from the set of feasible postures – obtained using the CyberGlove. ND-Simplex search provides multiple candidate postures, which are then followed by sequential Monte Carlo simulation seeded by the latest vertices of simplex, a.k.a. multiple objectives simplex algorithm. The search's objective function is derived from an observation model, derived from the image edges and model's silhouette. As many as 30 simplexes for each finger articulation and 10 simplexes for the global hand orientation are used in the search, and it takes 2 sec/frame to run. This approach is similar to ours, but we utilize hand markers to direct the search for posture hypotheses and we use a more sophisticated observation model.

Fei et al. [9], [10] propose a hybrid hand tracker using a particle filter to track the rigid (global) hand motion and HMM to estimate the intrinsic hand articulation. In other words, 2D position of the hand is tracked using Monte Carlo (Condensation) estimation and HMM is used to learn the timely variation of the hand shape (posture), which is represented by the hand silhouette moments. The hand motion is restricted to 2D motion in the camera plane and the hand articulations vary from open hand to fist position. The tracker follows the hand position and provides an estimate of the current hand shape (a.k.a. articulation).

Lu et al. [11], [12] use multiple optical cues in the form of edges, optical flow and shading to derive 3D forces that are applied to a 26 DOF hand model. They use a forward recursive dynamic model to track the motion in response to the 3D derived forces. We consider their work complementary to our own as we are both using multiple visual cues and a 26-DOF hand model, although there approach used dynamics and forward kinematics concepts while ours uses inverse kinematics to track the hand. The main difference is that their approach is incremental, i.e. it incrementally updates the state (posture) of the hand based on differential forces (e.g. edge movement, optical flow, etc.), which means it requires an accurate initial estimate of the hand posture and that it is prone to drifting error along the tracking process. Our tracking technique is stand alone and does not build on the previous posture and can estimate the initial posture. A hybrid approach can definitely improve the stability of both tracking systems.

Stenger et al. [13] use a 27 degrees-of-freedom (DOF) truncated conics-based hand model and an unscented Kalman filter to track hand postures in a 3D trajectory. They extract the hand model edges, handle occlusion using a simple ray tracing technique, comparing the projected model edges with edges extracted from video, and produce a difference vector that is used by a Kalman gain matrix to estimate the new posture (state) of the model in an attempt to minimize the difference between expected and observed postures. Another approach proposed by Stenger et al. [14] uses a hierarchical Bayesian filter to track the hand in 3D. The parameter space is divided hierarchically and posterior probability of the central posture, within each range, given the hand silhouette and skin color, is derived. Edge likelihood is based on the chamfer distance between the extracted hand silhouette and the expected silhouette and color likelihood is based on skin color probability distribution and background color probability distribution. Highly probable ranges are further divided to reach fine resolution/accuracy and low probability regions are used to reject hypotheses early in the search process. The algorithm is demonstrated to estimate global hand position and orientation in and out of the image plane, given the hand articulation. The method does not scale nicely with the observation model, in other words the posterior probability is generated separately for every hand posture and can expand exponentially to consider all possible postures.

Nirei et al. [15] track the hand by maximizing the intersection between the extracted hand silhouette region and the projected hand model and minimizing the error between the detected optical flow vectors and the predicted optical flow vectors from the model. They use two optimization algorithms; first a genetic algorithm (GA), a.k.a. random search, is used to quickly approach neighborhood of the best answer, and then simulated annealing (SA) is used to reach the local optimum. Simulated annealing is a generic probabilistic meta-algorithm for the global optimization problem, namely locating a good approximation to the global optimum of a given function in a large search space. It was independently presented by S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi in 1983 [102], and by V. Černý in 1985 [103]. The method is an adaptation of the Metropolis-Hastings algorithm, a Monte Carlo method to generate sample states of a thermodynamic system, invented by N. Metropolis et al in 1953 [104]. The method is shown to provide good estimation of hand postures from synthetic images with constant background.

Lee et al. [16] analyze model based hand tracking using a color marked glove in stereo images. They state that the challenges facing posture estimation are high level of articulation (DOF), self occlusion and shape deformation. They approximate the finger movement using a hand model and provide a comprehensive analysis of the fingers motion constraints. Seven characteristic points on the hand are identified of great importance in posture estimation, including the finger tips and the thumb MCP and wrist intersection with the middle finger extension. They utilize the hand natural constraints and provide a set of static (independent) and dynamic (inter-dependent) finger joints constraints. They provide a model fitting algorithm based on two algorithms, one for fitting the hand and another for fitting the fingers. The hand fitting algorithm calculates a rotation axis passing through the wrist as the axis of the highest weighted torque, which is generated from the difference between the position of the characteristic points expected and their observed positions from the image. The hand is rotated incrementally trying to minimize the torque, while in every increment the finger fitting algorithm is used to update the corresponding hand posture. The finger fitting algorithm is an inverse kinematics module utilizing the fingers natural constraints.

Guan et al. [17] use multi-view appearance based approach to extract 3D hand still-postures using multiple cameras. They compare the detection results against the single camera approach. Skin color represented by a 2D Gaussian model in normalized r - g color space. The hand silhouette is extracted using active contours. Synthetic images of 15 gestures, distributed over the viewpoint sphere were generated for training, and the associated hand configuration was used as a reference. A shape context feature descriptor [105] of the hand contour is used for matching. The descriptor is based on counting the edge pixels in the 8 surrounding partition of the image at every edge point, normalizing the number of pixels to add to one, threshold into binary [0,1] value and then encode the result into an 8-bit number in the range [0, 256]. The histogram of all encoded numbers provides a shape descriptor, which is translation and scale invariant. The method of nearest-neighbor is used to select the recognized hand posture. They compare the success rate of retrieving the correct hand posture from the best N matches. The success rate was found to be significantly higher using two cameras than using each camera separately.

Kolsch et al. [18], [19], [20] use a flock of pyramidal KLT features to track the hand in variant background and lighting conditions. The candidate features (flock of features) are continuously updated using location and color constraints to maintain hand tracking. A customized posture recognition based on the Viola-Jones method is used to recognize 5 gestures. The method is made scale invariant by resizing the hand bounding box to a predefined size. The method reported to perform in real-time speed and provide good recognition rate.

Yeasin et al. [21] analyze oscillatory dynamic hand gestures. The gestures are conformed by directional hand motions (no articulation) and are segmented at points of directional change. The gestures are modeled using state diagrams, where the states are the hand motion direction (start, left, right, up, down). A temporal segmentation of image sequences, based on Laplacian of Gaussian operator of image sequences in the temporal direction, is performed followed by dominant motion estimation. the resulting motion segments are then used to travel the corresponding gesture state machine.

Gui et al. [36] combine energy based segmentation, i.e. segmentation based on static image data, with dynamic posture priors derived from observing the behavior of the segmented object over time, to segment the hand in clutter and noisy backgrounds. Their work follows the school of active contours by adding dynamical statistical shape priors [37]. They show that utilizing the behavioral priors can improve the hand segmentation even under temporary clutter. The assumption is that the leaned timely shape variation is recurring during tracking, i.e. behavioral priors are utilized to assert proper segmentation in the case clutter. While behavioral priors are evident in the case of walking or running (gait) motions, we find less evidence of such recurrence in the case of hand motion and/or articulation.

Brèthes et al. [63] use the condensation framework to track human faces and hand gestures. The gestures consist of a predefined posture set moving in a plane parallel to the camera plane. Multiple cues, such as hand posture silhouettes were fused with the skin color filters to provide more robust tracking

Bryll et al. [99] use an agent-based approach to track a two hands trajectory in conversations gesticulation experiments, particularly under trajectory intersection. The agents are arranged in a hierarchical architecture, where low level blobs detected are assigned an agent each and labeling agents are responsible for keeping track of trajectory of the blobs. While this is simple task in the case of non-intersecting blobs, it may cause some ambiguity in the case when blobs join and then separate when the hands trajectories intersect. The advantage of the agent-based approach is that it provides an abstraction tool to solving complex tracking and sensor fusion problems.

Table 1: Summary of Surveyed Hand Tracking Algorithms

Index	Color Space	Algorithm	Output	Setting
[1]	YUV	CamShift (cf. [2])	Hand blob centroid (x, y)	Single Camera, Planar Movement, Marker-less
[3]	YUV	CamShift + Kalman filter	Hand centroid and ROI	Single Camera, 2D, Marker-less, real-time 25 fps
[4][5]	RGB	Background Subtraction	Hand segment and silhouette	Single Camera, 2D hand projection, i.e. shade
[6]	RGB	Motion and skin color	Dynamic view-based object model	Single Camera, 2D, Multiple Views, Marker-less, predefined objects
[7]	N/A - edges	NM-Simplex and Condensation (cf. [56])	20-DOF articulated hand	Single Camera, CyberGlove Reference Table, Nonlinear optimization, Observation Model, Marker-less
[9][10]	YUV	Condensation and HMM	2D location and hand silhouette	Single Camera, Predefined viewpoint (front), marker-less

[11][12]	Gray	Physics-based deformable models (cf. [57])	20-DOF articulated hand	Single Camera, Marker-less, 3D motion
[14]	RGB*	Hierarchical Bayesian Filter	Partial estimation of 26-DOF articulated hand and pose	Single Camera, Marker-less, 3D motion
[16]	HSV	Proprietary model fitting	27-DOF articulated hand	Stereo Colored Markers
[17]	RGB*	Maximum a posteriori – Table lookup	Best N matches from table lookup	Multiple Camera, Static Postures, Synthetic postures as reference, shape context as search index
[19]	RGB+	Flock of features	Hand centroid and ROI	Single Camera, Predefined viewpoint, 2D postures, unconstrained background
[36]	Grey	Active Contour + Dynamic Shape Priors	Segmented Hand	Single Camera, predefined hand posture priors, Marker-less
[63]	RGB	Condensation + Multiple cues	ROI, Motion direction, classified Posture	Single Camera, Predefines viewpoint, Marker-less

* 2D Gaussian model in r-g space
+ Dynamically acquired histogram

Table 1 provides a summary of the surveyed hand tracking technology. It illustrates that the vision based hand tracking papers have been mostly concerned with global position in 2D plane trajectories. Some attempts to include a simplified articulation falls short from providing complete hand articulation estimation. Our interest is primarily focused on hand tracking resulting in resolving hand orientation and complete articulation, including finger joints. We are also interested in single camera hand tracking and multiple camera sensor fusion, but not stereoscopic cameras.

Researchers investigating hand posture estimation using a single camera use a non-linear search in the parameter space directed by an observation model as its objective function.

The characteristic function is usually *not convex* and the search requires *multiple hypothesis* [24], [7]; or they restrict the search in a subset of the posture attribute space [14]. The nonlinear optimization problem is usually processing intensive and too slow for real time applications. The search space can be restricted by detecting the fingertips but it requires the use of colored markers [16].

2.2. Gesture Recognition

Examining the work related to gesture recognition, Xu [41] uses a Back-Propagation Neural Network to recognize hand gestures in a virtual environment for a Self-Propelled Gun armed vehicle driver training. Hand posture is acquired using the CyberGlove and a posture attribute vector is constituted by 18 joint angles. The gestures are in fact static postures related to starting the vehicle, steering, changing gears, and chandelling instruments. The BP-NN has 18 inputs and 15 outputs, mapped on 15 gestures, and an optimal number of neurons in the hidden layer were determined experimentally to be 40. Data from five different operators was collected and recognition rate was 98%, but dropped to 92% when tried with alien operators.

Liu and Fujimura [48] use a depth camera to recognize hand gestures; they assume a single user and segment the hand based on aspect ratio. The hand gesture is loosely used to mean a fixed hand posture with a trajectory in 3D. Hand orientation is based on template matching with reference images obtained at different orientation angles. Wachs et al. ([49], [50]) use Fuzzy C-Means clustering (FCM) to recognize hand postures and control a robot motion. Posture attributes are directly extracted from 2D images using edge detection; horizontal and vertical image sub-division was used to generate an attribute vector for posture recognition. Min et al. [51] introduce the use of Hidden Markov Models (HMM) to recognize gestures. Bretzner el al. [52] use multi-scale color features to extract elliptical blobs representing the palm, the 5 fingers, and finger tips at a hierarchical, coarse-to-fine, hand model. They also use randomization of the training data to build their model, but they use randomization on the training data as well as the observed data, which could slow down recognition. Nickel and Stiefelhagen [53] use stereoscopic range information and skin color for hand and head segmentation. They also use an HMM to model the three different phases of the pointing gesture, namely Begin, Hold, and End. A pointing gesture is detected when the three models are triggered consecutively in order. Marcel et al. [54] use an input-output HMM architecture [55] to recognize hand gestures in one of two classes of gestures, namely deictic and symbolic; it can be used to propagate, backward in time, targets in a discrete space of states using a set of input and output state neural networks.

Starner et al. [62] use HMM to recognize 40 words of the American sign language, including pronouns, verbs, nouns, and adjectives. The hands are extracted using region growing of skin color. A sixteen-element feature vector is constructed from each hand's x and y position, change in x and y between frames, area (in pixels), angle of axis of least inertia (found by the first eigenvector of the blob) [5], length of this eigenvector, and eccentricity of bounding ellipse. A 4 state HMM with skip was used to model all the sign words. Recognition rate of 91% for second person view and 96% for first person view was achieved. Note that the hand posture details was de-emphasized as studies of sign readers suggest that little hand detail is required for recognition.

Malima et al. [66] detect hand gestures in the form of a count 1 to 5 by detecting the largest skin color in the image and the distant fingertips. A circle with radius of a percentage of the finger lengths is drawn and the number of skin colored regions intersecting the circle radius is used to recognize the gesture (1, 2... 5)

Kobayashi and Haruyama [95] use Partly-Hidden Markov Model (PHMM) to recognize six hand gestures from the Japanese sign language. They argue that in HMM the relationship between the output sequence and the hidden state is probabilistic, i.e. the same state may be associated to a variety of output sequences; on one hand this makes the model simple because it generalizes based on the training data, but on the other hand the process modeled by HMM is restricted to a piecewise stationary process, which does not suit the gesture recognition. In a PHMM the state is divided into two types: first the state associated with previous observations, and it is assumed probabilistic, second the state associated with the current observation, and it is assumed unique (determined).

Howell el al. [38] use a time-delay radial basis function network (TDRBF) to recognize the different phases (pre-, mid-, post-) of two gestures (get, return) using 3D trajectory data acquired using magnetic sensors from different subjects seeking (getting) targets in 3D and returning back. The RBF is a two-layer hybrid learning network (a.k.a. a special kind of neural networks). It combines a supervised layer from the hidden to the output units with an unsupervised layer from the input to the hidden units. The advantage of RBF, over a Multi-layer Perceptron (MLP), is that it is supported by a well developed mathematical theory; it provides rapid computation and robust generalization, nonlinear decision boundary - good functional approximation, and low false-positive classification rates.

Kahol et al. [116] use an event-driven-coupled HMM (CHMM) to recognize whole body mannerism gestures. A 23-state segmental force HMM, where segmental force is the body segment (whole body, lower body, upper body, left leg, etc.) acceleration times its mass. Segment events are triggered as local minimum in the segmental force. A 14-state joint-HMM where events are triggered when a specific joint angle is stabilized. Each of the states of the segment-HMM is coupled with all of the states in the joint-HMM, based on the work of Zhong et al. [117].

Table 2: Summary of Surveyed Gesture Recognition Algorithms

Index	Algorithm	S/D	Num Gestures	Rec. Rate
[41]	BP-NN	S	15	98%
[48]	Template Matching	S*	10	95%
[49][50]	FCM	S	13	98%
[62]	HMM	D	40	91% ~ 96%
[95]	PHMM	D	6	98.8% ~ 93.5%
[38]	TDRBF	D	2	100% ~ 94%
[116]	CHMM	D	26	90%

* Static hand posture plus a trajectory in 3D space

2.3. Context Awareness

The interest in ubiquitous computing started in the early 90s by Mark Weiser's vision [75] of the computers of the 21st century, in which he stated that the current computing machines "*cannot truly make computing an integral, invisible part of the way people live their lives.*" He also hinted to context-aware computing by stating that "*the arcane aura that surrounds personal computers is not just a 'user interface' problem;*" hinting that the surrounding environment must play a bigger role in the human-computer interface process. He set the goal of the new generation of computing devices as: "*integrating the computers seamlessly into the world at large*" by filling the world with "*invisible widgets*". A similar goal with set by Coen [77] in 1998 "*to bring computers into the real physical world and to allow people to interact with them in a more natural way: by talking, by moving, pointing, and gesturing.*" Gesture-enabled user interfaces aim at providing an alternative mode for human-computer interaction (HCI) that makes human-computer interaction more natural and intuitive to humans, forcing computer systems to bridge the gap, between human-to-human interaction and human-to-computer interaction, by freeing the user from keying in commands using a keyboard or using a standard graphical user interface with a pointing device. The notion of context in ubiquitous computing has a dual origin [67]. On the one hand, it is a technical notion, one that offers system developers new ways to conceptualize human action and the relationship between that action and computational systems to support it. On the other hand, it is also a notion drawn from social science, drawing analytic attention to certain aspects of social settings. ***Positivist*** (or Pragmatic) theories derive from the rational, empirical, scientific tradition. By analogy with the way that physical scientific theories seek to reduce complex observable phenomena to underlying idealized mathematical descriptions, positivist theories seek to reduce social phenomena to essences or simplified models that capture underlying patterns. Accordingly, positivist theories seek objective, independent descriptions of social phenomena, abstracting from the detail of particular occasions or settings, often in favor of broad statistical trends and idealized models. Positivist theories are often (although not always) quantitative or mathematical in nature. ***Phenomenological*** theories are subjective and qualitative in orientation. By "subjective" is meant that they regard social facts as having no objective reality beyond the ability of individuals and groups to recognize and orient towards them; in this view, social facts are emergent properties of interactions, not pre-given or absolute but negotiated, contested and subject to continual processes of interpretation and reinterpretation. The distinction between positivist and phenomenological theories is relevant because engineering approaches inherit a positivist tradition, while many approaches to social analysis relevant to HCI design, including the ethno methodological position practiced by Suchman [68] and cited by Weiser [76], are related to a phenomenological legacy.

We can think of the positivist account of context as defining the problem as one of representation. Software systems are representational, so a concern with context naturally leads to a concern with how context can be encoded and represented. In particular, four assumptions seem to underlie the notion of "context" as it operates in these systems. Firstly, *context is a form of information* and hence encoded and represented much as other information is encoded and represented in software systems. Secondly, context is delineable. We can, for some set of applications or application requirements, *define in advance what counts as the context* of activities that the application supports. Thirdly, *context is stable (static)*, invariant and *non-volatile*. Fourthly, and most importantly, *context and activity are separable*. Activity happens "within" a context. The context describes features of the environment within which the activity takes place, but which are separate from the activity itself.

An alternative view takes a different stance of each of the four assumptions mentioned above [107], [91]: Firstly, rather than considering context to be information, it instead argues that *contextuality is a relational property* that holds *between objects and/or activities*. It is not simply the case that something is or is not context; rather, it may or may not be contextually relevant to some particular activity. Secondly, rather than considering that context can be delineated and defined in advance, the alternative view argues that the scope of contextual features is defined dynamically, in other words *context can be negotiated and inferred*. Thirdly, rather than considering that context is stable, it instead argues that *context is an emergent phenomenon* particular to each occasion of activity or action. Context is an occasioned property, relevant to particular settings, particular instances of action and particular parties to that action. Fourthly, rather than taking context and content to be two separable entities, it instead argues that *context arises from the activity*. Context does not exist by itself, but is actively produced, maintained and enacted in the course of the activity at hand.

From a system designer point of view, it is inevitable that we must define the classes and objects that are modeled by our system, these objects include the hand model, which is required for modeling the hand kinematics, the dynamic gesture model, which is used in gesture training and recognition. We also acknowledge the existence of contextuality and to cope with such an ambiguous concept we utilize a stochastic model for posture estimation and gesture recognition and an agent-based model for gesture interpretation.

For a context-aware perceptual user interface, it is inevitable that the trained gesture set must be pre-defined, i.e. static, but the active gesture set at any time may be variant based on external context. In other words, even though the space of trained gestures may be pre-defined and statically bounded, the canonical probability of each gesture can be correlated with aspects of the surrounding context, such as people and objects in the scene.

Schilit et al. [69] introduced the term context-aware computing for the first time in 1994. They use the PARCTAB platform to investigate context-aware computing applications. PARCTAB is a small hand held device which uses an infrared-based cellular network for communication. The Tab acts as a graphics display terminal and most applications run on remote servers. It has three buttons and a touch-sensitive screen for input, and 128x64 pixels screen and a speaker for output. A room becomes a cell in the infrared network when it is wired by an infrared transceiver. In the context of mobile distributed computing, it is emphasized that a limited amount of information covering a person's proximate environment is very important, since "the interesting part of the world around us is what we can see, hear, and touch." They highlight three aspects of context useful in context-aware computing, namely location, people, and computing resources in proximity. Location plays the major role in context with proximity is the only measure used to infer context. They present two modes of context use in applications, a context-based browsing mode, and a proactive context-triggered action mode.

Strang and Linnhoff-Popien [70] provide a comprehensive survey for context modeling in ubiquitous computing applications and evaluate different context-modeling techniques based on the following qualitative measures:

- The ability for distributed composition, including the ability for partial validation of context information
- Support for quality of context information, including the ability to handle incomplete and ambiguous context information and the required level of formality of the context specifications
- Applicability to existing infrastructures, including availability of tools and software development trends.

Modeling approaches studied include Attribute-value pair, Markup schemes such as XML based Resource Description Framework and schemas (RDFs), Graphical Models such as Unified Modeling Language (UML), Object Oriented (OO) models, Logic Based models, and Ontology based models. They conclude that Ontology and Object Oriented models are best candidates for context modeling. Markup schemes (RDFs) run a second best but is missing handling the quality of context info and incompleteness and/or ambiguity of the data.

Strang and Linnhoff-Popien [71] motivate the need for a context level interoperability between services in a distributed system. They provide a brief history of distributed systems level interoperability, which started in the 1980's by the platform interoperability represented by the concept of Remote Procedure Call (RPC). An RPC framework allows a process to cause process to run on a different system/platform, the calling process providing the input parameters and receiving the results in the form of synchronous or asynchronous messages. Before and during the usage of a service, the relevance of context information is very important. The set of relevant entities is changing rapidly. In contrast, the set of the specific aspects determining the relevance of an entity is fixed. Thus, the relevant aspects have to be specified a priori in the system, and the relevant entities have to be evaluated dynamically at run time. The state of relevant entities is examined periodically to evaluate the current context information. For contextual service interoperability the terms compatibility and substitutability are defined. Compatibility is a common understanding of all relevant aspects, e.g. Ontology. Substitutability is defined as having an identical set of relevant aspects.

Dey et al. [73] provide a conceptual framework and a rapid prototyping toolkit for context-aware applications. They use the widget model in graphical user interface (GUI) design to hide the complexity of acquiring and modeling context in context-aware applications. Dey [74] defines context as any information that can be used to characterize the situation of an entity, where an entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves. A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task. They developed a Context Toolkit (CTK) for rapid prototyping of context aware applications. The context toolkit provides four levels of abstraction that the context-aware system developer can use to isolate the details of context acquisition and aggregation from the core of the application. These concepts are namely the context widget, context interpreter, context server aggregator, and a context enactor. The Widget is the lowest level of abstraction, which is responsible of a specific aspect of context in a particular format, e.g. room temperature in degrees Celsius. It encapsulates the details of how the context is acquired from the applications that uses them. Context widgets differ from the graphical user interface widgets in that they do not belong to a particular operating system or computer, but exist as a persistent process in a distributed system. Context interpreters use historical data of the context widget to enable detection of usage patterns and allow context learning. The interpreter can be attached to one or more widgets and provides information about a specific aspect of context. The context server, or aggregator, is yet another level of abstraction, where the output of one or more interpreters is collected to generate the context relevant to a specific (physical/abstract) entity. Aggregation is required in distributed systems to gather the information about a specific entity across platforms, systems, and databases. A context enactor infers an event or a situation from the status of one or more entities. It triggers the system to automatically respond to the relevant events happening in the environment.

Meyer et al. [78] present the special requirements for context-aware computing systems used at home in contrast to similar systems used at work; they explain that devices used at home have to be cheaper, easier to use, safer, more supportive, entertaining, relaxing, enjoyable, and pleasant. A special feature about homes is also privacy which is paramount there. Therefore, context-aware devices used at homes must pass rigorous test for usability, usefulness, social acceptance, privacy protection, low cost, and zero administration. They also classified the system design modules of context-aware systems, including the instrumentation (hardware) interface, the middleware (hardware abstraction, context management, and privacy management), and the application. They also present a survey of trends of instrumentation for ubiquitous computing, including:

- The Tele-cooperation Office (*TecO*) [79] of the University of Karlsruhe. It has developed *Smart-Its*, small-scale embedded devices equipped with sensing, processing, and communication capabilities which can be attached to everyday objects to let them establish dynamic digital relationships with their environment and users, the best known of which is the *MediaCup*.
- The *MOTES* [80], which have been developed at the University of Berkeley as part of the *Smart Dust* project, with the final goal being to make then as small as a grain of sand [81].
- The *Cricket* Indoor Location System [82], developed at MIT. It uses a combination of RF and ultrasound technologies. Wall and ceiling-mounted *beacons* are spread through

buildings, publishing information on an RF signal and concurrently sending an ultrasonic pulse. The mobile receivers of these signals can determine their positions due to their different propagation times.

- The AT&T Cambridge *Bat* location sensor system [83] uses the same technologies, though their mobile *bats* send ultrasound pulses instead of receiving.

They also present more vertically integrated systems research including:

- The MIT *tangible media* group lead by Hiroshi Ishii [84].
- Saul Greenberg from the University of Calgary [85] developed physical widgets, which he called *Phidgets*, as building blocks to help developers to construct physical user interfaces.
- Video camera based systems to track people moving around in a room have been developed by the Vision Interfaces Group at MIT [86].
- Microsoft *Easy-Living* project [87] research revealed that users prefer to communicate with their environment through gestures and speech [88] which could be realized by using the cameras in combination with microphones.

Chapter 3

3. Hand Tracking

Dynamic gesture recognition applications require continuous provision of hand postures at high data rate. Hand postures are completely defined by the global or external geometric parameters, namely the hand's position and orientation, and the local or intrinsic parameters, namely the finger joint angles. The posture is visualized using a kinematics hand model defining the link and joint structure of the hand. Researchers have used models of variant degrees of freedom ranging from 19 [23] to 57 [17] DOF, but typically 26~27 DOF is mostly used in the literature.

Traditionally hand tracking is provided by motion capture instruments such as exoskeletons and data gloves [22]. Data gloves are capable of accurately recording finger joint motions, in real time via flex sensors in a tightly fitting glove. Although the data glove is very precise hand posture estimation device, it is very expensive for most commercial applications. Moreover, it is cumbersome to wear and potentially obtrusive. Because of its fixed setting, the data glove does not adapt to the size of the hand, resulting in variant posture data across different users, which is particularly noticeable in multiple user gesture recognition applications. Another fundamental problem with the data glove is that, because it uses forward kinematics to visualize the hand posture, it is error prone to detect finger tips touching one another; whereas inverse kinematics start from the end-effector position, which facilitates neighborhood detection.

Research in vision-based hand tracking started in the mid nineties and has been gaining interest ever since; motivated by the cheaper camera peripherals and the increased capabilities in terms of memory storage and processing power of modern computers. The quest for vision-based hand tracking took diverse approaches to the problem, but the main approaches can categorized as appearance-based and model-based hand tracking [35]. Appearance based hand tracking utilizes pattern recognition techniques to classify the extracted hand features to one of, usually distinct, hand postures. In other words, appearance based hand tracking sacrifices the detailed description of the hand posture in favor of direct inference of the posture as a member of a discrete posture set. While it provides faster tracking and requires less processing than model-based hand tracking, the approach does not scale nicely with a large posture set. As tracking the dynamics of hand postures requires detection of small changes of posture overtime, we realize that the appearance based approach is not the best approach for this type of application. Model based hand tracking aims at finding the best fit of a hand model to the segmented hand image, given some extracted image features. In that sense, it represents a multi-dimensional optimization in the posture attributes space, given an optimization function in the form of a resemblance or fitting metric.

The questions we are trying to answer in this part of the dissertation are:

- Can we use vision-based hand tracking as a substitute for the traditionally data glove?
- What is the accuracy of the vision-based hand tracking in comparison with the data glove?
- What are the best and worst features to track using vision-based hand tracking?
- What are the potential advantages and limitations of the vision-based hand tracking over the data glove?
- What are the future directions and challenges to realize a usable vision-based hand tracking in an unconstrained environment?

In the next section, we are trying to answer the above questions.

3.1. Overview

The overview of the proposed hand tracking and posture estimation framework is shown in Figure 2. First, 2D features are extracted from a single camera. The system can be extended to utilize multiple cameras by processing the 2D features from each camera separately; we do not require stereoscopic processing. 2D features are processed by the 2D/3D conversion module, which uses perspective geometry to extract the hand orientation in 3D and inverse kinematics to detect possible finger postures, and outputs multiple possible postures. The tracking algorithm filters the posture hypotheses provided by the 2D/3D module using visual cues in the form of proximity between the projected fingertips and detected ones, detected hand silhouette and hand color, as well as the deviation of the posture hypothesis from expected. Posture expectation is derived from a historical repository of estimated postures and an optional canonical probability of posture.

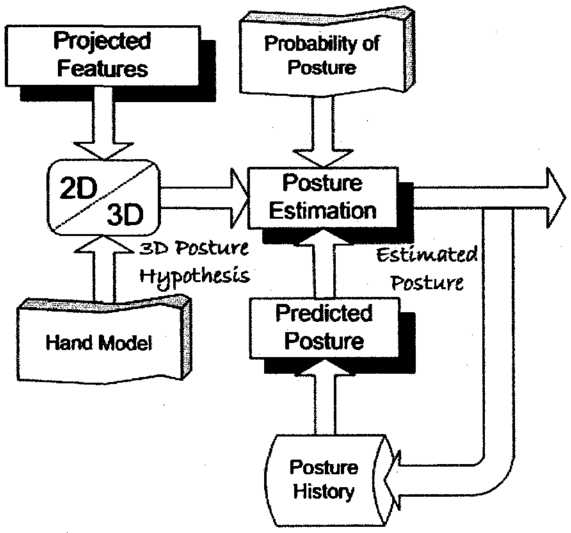


Figure 2: System Overview – Hand Tracking Simulation

The 2D/3D converter module utilizes a 26 degrees-of-freedom (26-DOF) hand model used primarily for forward and inverse kinematics. We set to build a hand tracking system prototype, including a colored marker glove (Figure 3).

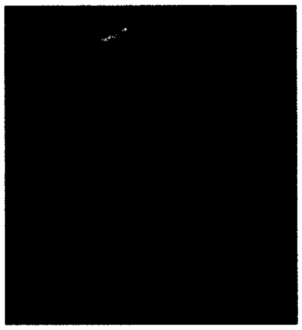


Figure 3: Marker Glove

The marker glove contains a palm marker, which is used to estimate the hand position and orientation in 3D, and fingertip markers, which are used for finger posture estimation. The process of building the hand tracking system went through three phases, namely graphic simulation, single camera, and multiple camera tracking.

This chapter is organized as follows: First, we explain the tracking simulation phase, including detailed explanation of the main modules of the tracking system, namely 2D/3D, the hand model, posture prediction, posture estimation, and inverse kinematics modules. We also compare the simulation results with the CyberGlove, and use the comparison to assess the accuracy of the system in estimating the different posture attributes. Then we explain the single camera phase, where we extend the posture estimation module to include an observation model providing a probabilistic measure off the posture hypothesis, given the acquired image. We also explain how the framework can be extended to accommodate multiple cameras. Finally, we show the results of calibrating our model to a Cyber Glove and the corresponding error models.

3.2. Tracking Simulation

To be able to compare the posture estimated using vision-based hand tracking to postures detected using the CyberGlove, we used real-time graphics simulation. The input to the simulator was the 2D projection of the hand model's fingertips, and the corners of a hypothetical 2 cm x 2 cm square centered at the hand model's root. The CyberGlove was used to drive the intrinsic posture attributes (20 DOF in total) of the 26 DOF hand model. The extrinsic 6 DOF representing the hand position and orientation were controlled by changing the camera viewpoint in the graphics scene by panning, zooming and scaling the model.

The minimum requirements of 2D features, required by the vision-based tracker to be able to provide a tangible estimate of the hand posture, are a reference to the palm and the fingertips. The palm reference facilitates the estimation of the extrinsic posture attributes and the reference to the fingertips facilitates the estimation of the intrinsic posture attributes.

The minimum requirements of image features we are proposing are:

- The projection of three non-co-linear points on the palm and their corresponding location in 3D with respect to the hand model
- The projection of the finger tips, or equivalent points on the distal phalanx, and their corresponding location on the hand model

Both of the above features were detected from the CyberGlove data. The overview of the simulated 3D hand tracking system is shown in Figure 4.

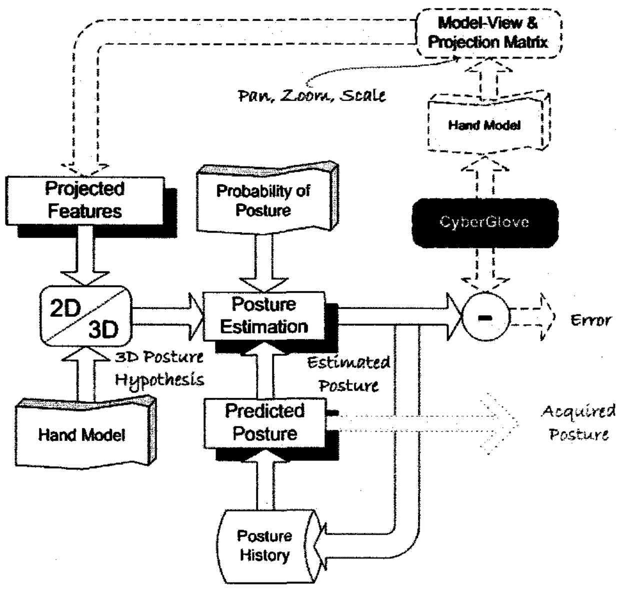


Figure 4: System Overview – Hand Tracking Simulation

The modules, which can be reused in vision based hand tracking as shown in Figure 2, are shown in solid lines, whereas the simulation specific modules are shaded in light color and dashed lines. The system *configuration parameters* include the *articulated hand model*, which may vary from one user to another, and the *canonical probability distribution of the posture attributes*, which may vary from one application to another. We consider all external parameters of the system as part of the context, which can be defined explicitly or recognized automatically by other subsystems. These parameters are manages by a context-management middleware layer, as will be explained later. These context parameters are emphasized in Figure 4 by data chunk blocks, which can be loaded dynamically using a special software loader and can be written in special format, e.g. XML. Context information may vary based on external circumstances of the system, for example the hand model may vary based on the system operator; it may be specific for each user or loosely dependent on its attributed such as age and/or gender. On the same token, the probability distribution of the hand posture may be affected by the setting and the application utilizing the hand posture. The processing modules are: the 2D-to-3D (2D/3D) transformation, the posture estimation and the posture prediction modules.

The 2D/3D module iterates feasible hand postures, given the simulated (image) features, by using perspective projection and inverse kinematics constraints. The posture estimation module provides a probabilistic model to estimate the hand posture from the given posture hypothesis, given the previous estimated postures. Posture prediction is a software interface, protected by a mutual exclusion (MUTEX), which can be utilized by the tracking system and other systems acquiring the estimated hand postures, such as gesture recognition modules; it estimates the current hand posture from the previous postures.

The CyberGlove is also used as a reference for vision-based hand tracking. Both the reference postures and the derived simulated 2D features are derived from the same source and are perfectly synchronized. The difference between the two posture-acquisition sources is used to validate the accuracy of vision-based posture estimation. Moreover, it is used to provide the stimulus for the tracking simulation. Figure 5 shows a visualization of the hand model in an Open Scene Graph; the viewpoint can be varied in the scene using the mouse, and fingertip projection is calculated using the perspective projection. The projected features are then used by the tracking system to estimate the hand posture; the difference between the CyberGlove measurement and the estimated posture is recorded and analyzed.

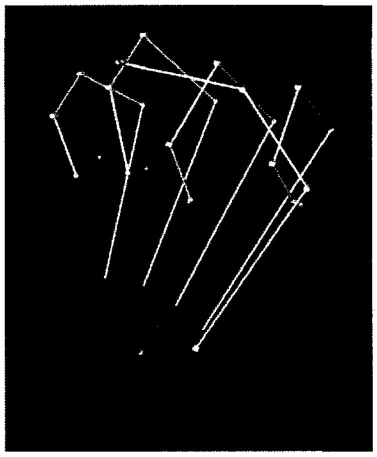


Figure 5: Hand Model Visualization in Open Scene Graph

In the following sections we will give a detailed description of the hand model, the 2D/3D transformation module, the posture prediction and the posture estimation modules.

3.2.1. Hand Model

The human hand is a masterpiece of mechanical complexity, able to perform fine motor manipulations and powerful work alike (cf. [25]). Designing an animate-able human hand model that features the abilities of the archetype created by Nature requires a great deal of anatomical detail. Model based hand tracking utilize an articulated hand model to fit estimated postures to the acquired hand image, whereas hand animation utilize additional models for muscle and skin deformation to provide a realistic animation. The anatomy of the hand is shown in Figure 6.

The human hand skeleton consists of 27 bones, divided to three groups: carpals – are eight wrist bones, metacarpals – are five palm bones, and phalanges – are fourteen finger bones. Joints connecting carpals together and carpals to metacarpals have limited freedom of movements and thus are usually ignored in modeling or animating the human hand. Joints connecting the finger phalanges contribute the most to finger motion. Hand models of 26 ~ 27 DOF provide realistic animation of the human hand, as we acquired from the literature survey. Our 26 DOF hand model provides forward and inverse kinematics transformation utilities, which are used extensively by the 2D/3D features to posture transformation and the 3D/2D posture projection on the image plane.

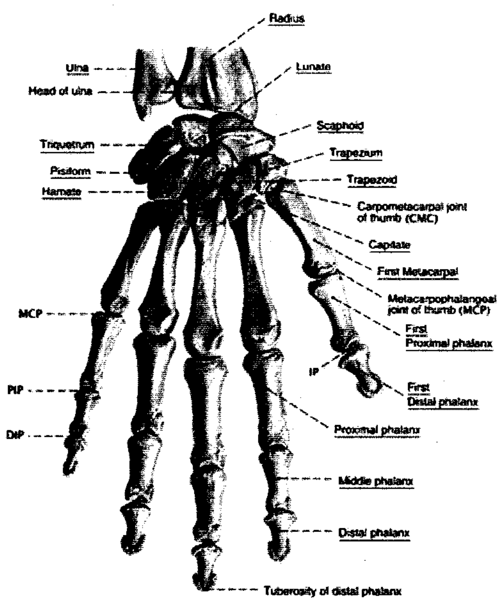


Figure 6: Anatomy of The Human Hand

Using a hierarchy of coordinate systems, we can model the degrees-of-freedom for each joint. The only joints that we ignore are the joints between the individual wrist bones (cf. Figure 6). This is justified, since their contribution to the overall movement is negligible. The Proximal Interphalangial joint (PIP) and the Distal Interphalangial joint (DIP) joints of the fingers and the Interphalangial joint (IP) of the thumb have one DOF each for flexion /

extension, while the Metacarpophalangeal joint (MCP) joints of the fingers have a second DOF for adduction (towards the middle finger) and abduction (away from the middle finger) – see Figure 7 for finger motion terminology. In addition, depending on the current amount of flexion or extension, the finger MCP joints exhibit some rotation around their long symmetry axis, but we ignore this minute motion. Likewise, the Carpometacarpal joint (CMC) joint of the thumb is sometimes modeled with two or three degrees of freedom, thus the 26/27 DOF in some hand models.

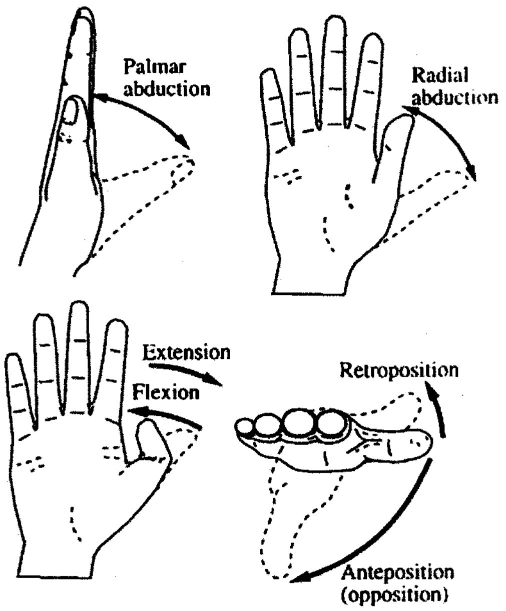


Figure 7: Fingers Motion Terminology

Our hand model (cf. Figure 8) consists of five kinematics chains, modeling the five fingers. All fingers, except for the thumb, consist of 4 links representing the fingers' bones; the first link, the metacarpal phalanx, is fixed to the root and is part of the palm; the second link, the proximal phalanx, is connected to the metacarpal by 2-DOF joint, which is modeled by two 1-DOF revolute joints connected by an abstract zero-length link for flexion and abduction movements. The order of the two joints is not arbitrary, as we will show later when we solve the finger inverse kinematics. The flexion joint supersedes the abduction joint for more realistic modeling of the finger movement. The outer most two links, the middle and distal phalanxes, are connected to the chain through a 1-DOF revolute joint providing flexion movement. The thumb has only three links; the first link, the first metacarpal, is connected to the root by 2-DOF CMC joint that we model using two 1-DOF revolute joints, for opposition/antiposition and radial abduction movements respectively, which are connected to each other by an abstract zero-length link. The other two links, namely the proximal and distal phalanxes, are connected to the chain through 1-DOF revolute joint.

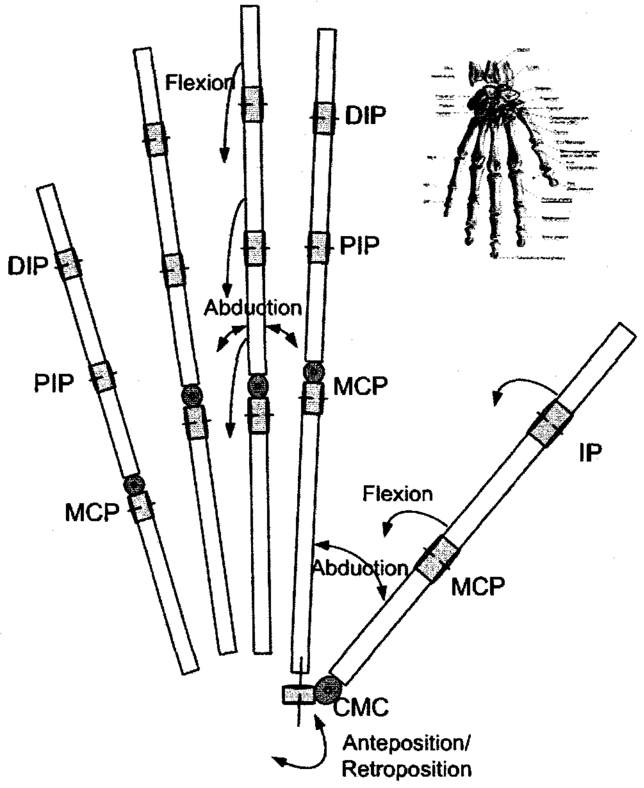


Figure 8: 26 DOF Articulated Hand Model

Modeling the Finger Constraints

Constraints play a major role in vision based hand tracking. It is found that even though the hand fingers possess 4 degrees-of-freedom, they are practically closer the 3-DOF than 4-DOF due to physical constraints and joints correlation. Kuch [58] divided the finger constraints into two main categories: static and dynamic constraints.

Table 3: Finger Constraints (cf. [35])

Static Constraints	
Finger	Thumb
$0 < \Theta_{MCP} < 90^o$ $-15^o < \Theta_{Abd} < 15^o$	
Dynamic Constraints	
$\Theta_{PIP} = 3/2\ \Theta_{DIP}$ $\Theta_{MCP} = ½\ \Theta_{PIP}$	$\Theta_{IP} = \Theta_{MCP}$ $\Theta_{CMC} = 1/3\ \Theta_{MCP}$ $\Theta_{Abd} = ½\ \Theta_{MCP}$

Static constraints are constant and represent the absolute physical limits of the finger joints. Dynamic constraints are dynamically imposed depending on other joint angles, and represent inter dependency between the finger joints. Table 3 shows the static and dynamic constraints proposed by Kuch.

Experimental measurements performed by Burdea et al. [59] showed that the general coupling between Θ_{PIP} and Θ_{DIP} can be represented as a quadratic function – equation (1), where *a*, *b*, and *c* are constants evaluated by fitting equation (1) to the sampling data..

$$\Theta_{DIP} = a - b\,\Theta_{PIP} + c\,\Theta_{PIP}^{2} \tag{1}$$

Other researchers (e.g. [27], [28]) use the approximation $\Theta_{DIP} = 2/3\ \Theta_{PIP}$ to reduce the finger IK problem complexity from 4-DOF to 3-DOF allowing the IK problem to be properly constrained using the finger end-effector. Others, (e.g. [31]) find the $\Theta_{DIP} = 2/3\ \Theta_{PIP}$ constraint too restrictive for intricate control of the hand, they prefer to use an acquired domain of possible hand postures using a data glove or a motion capture device.

Lee and Kunii [16] introduced a set of constraints when they tried to track a human hand using color marker and stereo camera. Firstly the four fingers (except the thumb) are planar manipulators with the exception of the MCP joint. Secondly, the joint angles of the PIP and DIP joints have dependency represented by $\Theta_{DIP} = 2/3\ \Theta_{PIP}$. Thirdly, the joint angle limits of the MCP joints depend on those of the neighboring fingers according to equation (2), where *dyMax*, *dyMin* are the dynamic and static bounds of the joint angles, respectively.

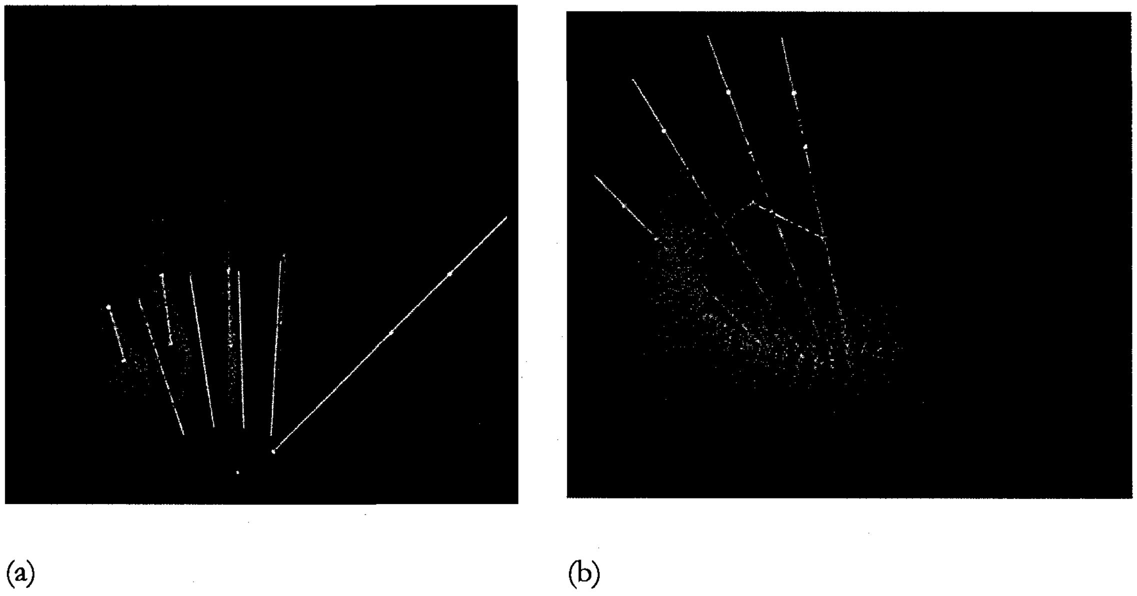
$$\begin{aligned}
dyMax(\Theta_{MCP|index}) &= \min\{\Theta_{MCP|middle} + 25,\ stMax(\Theta_{MCP|index})\}\\
dyMin(\Theta_{MCP|index}) &= \max\{\Theta_{MCP|middle} - 54,\ stMin(\Theta_{MCP|index})\}\\
dyMax(\Theta_{MCP|middle}) &= \min\{\Theta_{MCP|index} + 54,\ \Theta_{MCP|ring} + 20,\ stMax(\Theta_{MCP|middle})\}\\
dyMin(\Theta_{MCP|middle}) &= \max\{\Theta_{MCP|index} - 25,\ \Theta_{MCP|ring} - 45,\ stMin(\Theta_{MCP|middle})\}\\
dyMax(\Theta_{MCP|ring}) &= \min\{\Theta_{MCP|middle} + 45,\ \Theta_{MCP|pinky} + 48,\ stMax(\Theta_{MCP|ring})\}\\
dyMin(\Theta_{MCP|ring}) &= \max\{\Theta_{MCP|middle} - 20,\ \Theta_{MCP|pinky} - 44,\ stMin(\Theta_{MCP|ring})\}\\
dyMax(\Theta_{MCP|pinky}) &= \min\{\Theta_{MCP|ring} + 44,\ stMax(\Theta_{MCP|pinky})\}\\
dyMin(\Theta_{MCP|pinky}) &= \max\{\Theta_{MCP|ring} - 48,\ stMin(\Theta_{MCP|pinky})\}
\end{aligned} \tag{2}$$

Fourthly, the middle finger displays limited abduction and adduction movements. Finally, the finger abduction bounds are modeled according to equation (3).

$$\begin{gathered}
dyMax(\Theta_{Abd}) = k(\Theta_{MCP})\ stMax(\Theta_{Abd}),\\
k = (1 - 1/stMax(\Theta_{MCP}))\ \Theta_{MCP}
\end{gathered} \tag{3}$$

Wu et al. [56] use principal component analysis (PCA) to capture the correlation between finger joints from a large sample set, assumed to cover all possible postures, acquired using a CyberGlove. A parameter space subset of R^7 was found to cover 95% of the hand postures in the posture set. Moreover, using a set of 28 basis configurations, they found that natural hand articulation lies largely in the linear manifolds spanned by any two basis configurations. Same results were also confirmed by Stenger [14].

Figure 9 shows a pictorial illustration of the finger end-effector positions derived by spanning the finger's static constraints.



(a) (b)

Figure 9: End-Effector Positions, (a) fingers, (b) thumb

Figure 9(a) shows the fingers' end-effector positions produced by spanning the MCP, PIP and DIP joint angle intervals at 0.1 radians granularity. The finger abduction was ignored for clarity purposes, but will be shown later in the inverse kinematics section. Figure 9 (b) shows the thumb end effector positions produced by spanning the CMC, Abduction, MCP and IP joint angles intervals at 0.1 radians granularity.

In summary, the two main approaches to define the domain of all possible hand postures are:

- Acquisition of a large sample of all possible postures and then analyze the above set using statistical methods (e.g. PCA), or approximation methods (e.g. curve fitting), or map the solution to the closest posture in the sample (e.g. Nearest Neighbor).

- Utilization of the static and dynamic finger constraints to prune the posture attributes space (a.k.a. joint angles).

We prefer the second approach, because it is user independent. I other words, the static and dynamic joint constraints provides a closed form solution for the finger constraints problem and are independent from the user, although sometimes specific constraints can be added for special users. It can also be quickly adapted for new users, whereas the sampling data may vary significantly from one user to another, requiring a significant amount of preprocessing.

Interdependency between the finger joints is better studied at the whole fingers level as opposed to the sub-finger level, a.k.a. the fingers-joints level. One of the most important attributes that can be used to impose dynamic constraint is the finger curvature. The finger curvature is derived from the finger joints and can be used as parameter to model the fingers' dynamic constraints. Finger curvature is modeled as the reciprocal of the radius of a hypothetical circle having the finger as part of its circumference. This is calculated by allowing the finger's MCP, DIP and fingertip to lie on the circumference of the curvature circle, as shown in Figure 10. Using the finger curvature, we can model the fingers constraints as follows:

First, the finger curvature is always positive, i.e. the finger is allowed to curve in one direction in front of the palm, by eliminating finger joint angles combinations resulting in negative curvature (a.k.a. by fitting the curvature circle on the back side of the finger).

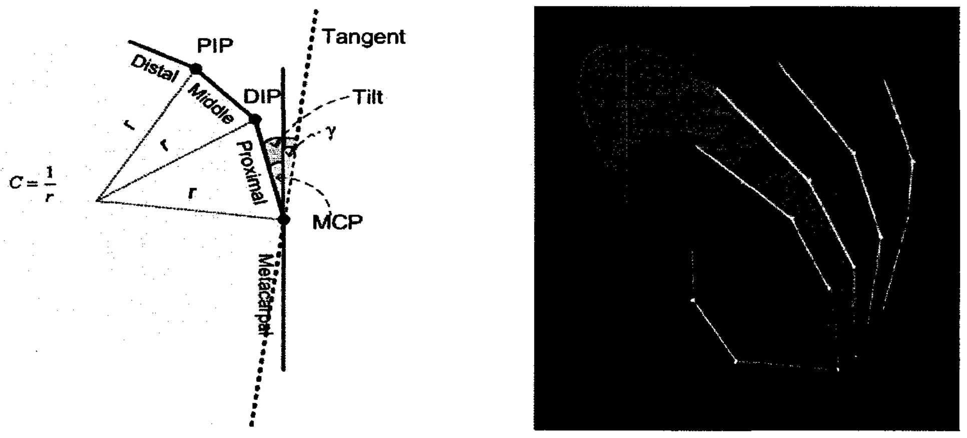


Figure 10: Curvature and tilt calculations

Second, we define another finger posture attribute, the finger tilt, which is defined as the angle between the curvature circle's tangent at the fingers base (MCP joint) and the line passing through the finger's proximal phalanx (cf. Figure 10). It compensates the finger's MCP joint by the curvature induced component γ – equation (4).

$$\text{Tilt} = \text{MCP} - \gamma \tag{4}$$

The curvature induced component γ is the angle between the circle tangent at the MCP joint and the Metacarpal phalanx, as shown in Figure 10. The tilt attribute can be used to model finger constraints by maintaining the range of tilt throughout the finger's posture space, resulting in constraining the MCP angle, particularly at high curvature. Figure 11 shows the finger tilt as it varies with the finger curvature for the attribute region bounded by the static constraints; it also shows the effect of pruning based on finger tilts at high curvatures.

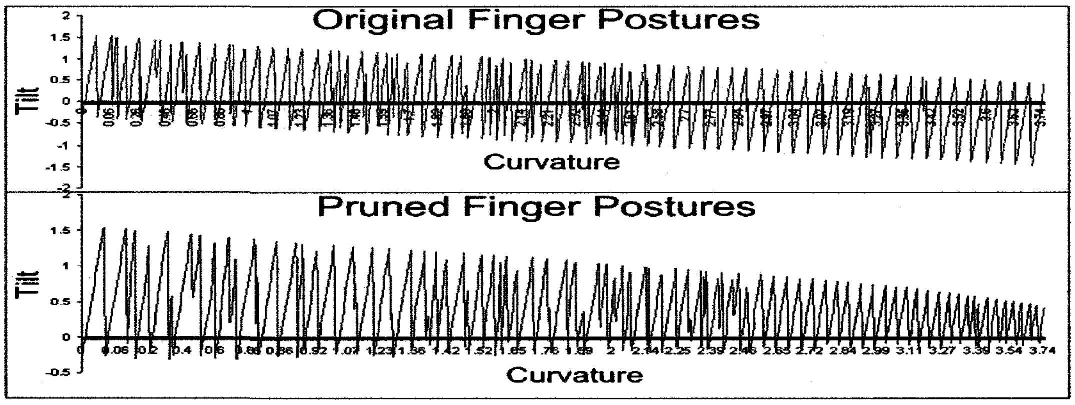


Figure 11: Posture pruning based on tilt

Third, we use the correlation between the finger's PIP and DIP joints as a range bound by two quadratic functions, instead of fixing them to a particular relation. We found the above constraint model easier to use and not requiring data acquisition using the data glove. In our quest for finger inverse kinematics we used both methods of generating the hand posture set, namely acquisition based and parametric modeling based, with comparable results.

3.2.2. 2D/3D Transformation

The 26 DOF of the hand model are estimated in two stages (as illustrated in Figure 12). In the first stage, the 6 DOF associated with the extrinsic posture attributes (the wrist location and orientation in 3D) are solved using the perspective geometry of detected 2D palm marker features. In the second stage, finger postures are determined using inverse kinematics (IK) from the detected fingertip positions.

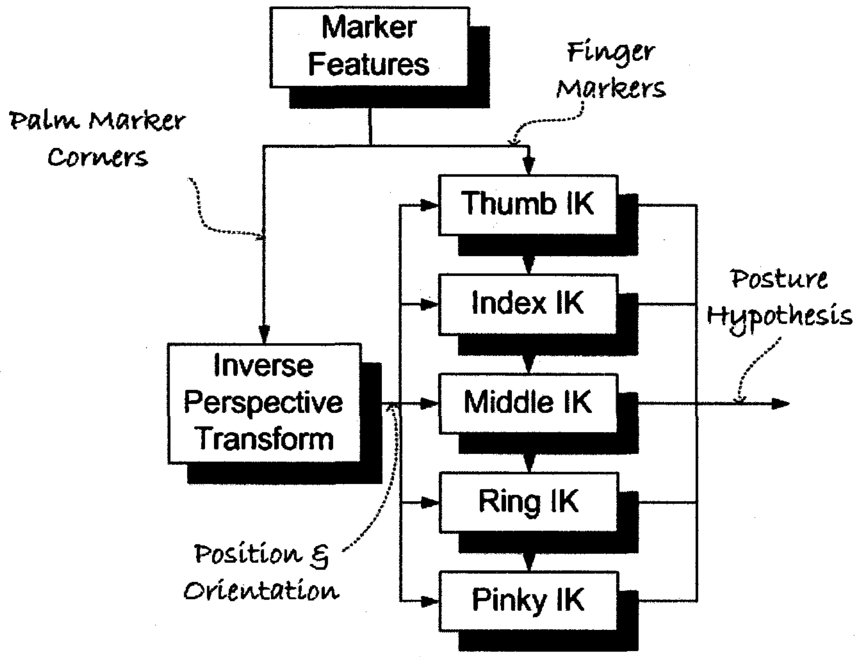


Figure 12: Outline of 2D/3D Conversion

3.2.2.1. Hand Position and Orientation

A 2cm x 2cm square marker is used to reference the hand position and orientation in 3D. We use a pin hole camera model and perspective geometry in our derivation. Hand position and orientation in 3D is formally defined by the camera to the hand model coordinate transformation *T*, which is expressed in homogenous coordinates form by a 4x4 matrix that includes an ortho-normal 3x3 rotation matrix *R*, representing the hand orientation in 3D and a translation vector *h*, representing the hand location with respect to the camera (in the camera coordinate system). The camera-to model-transformation is calculated as the product of the camera to palm marker transformation T_1, which is calculated on a frame by frame basis and the palm marker to hand model transformation T_2, which is considered constant and is set to the identity matrix in the simulation.

$$T \equiv \begin{bmatrix} R & h \\ 0 & 1 \end{bmatrix} = T_1 T_2 \tag{5}$$

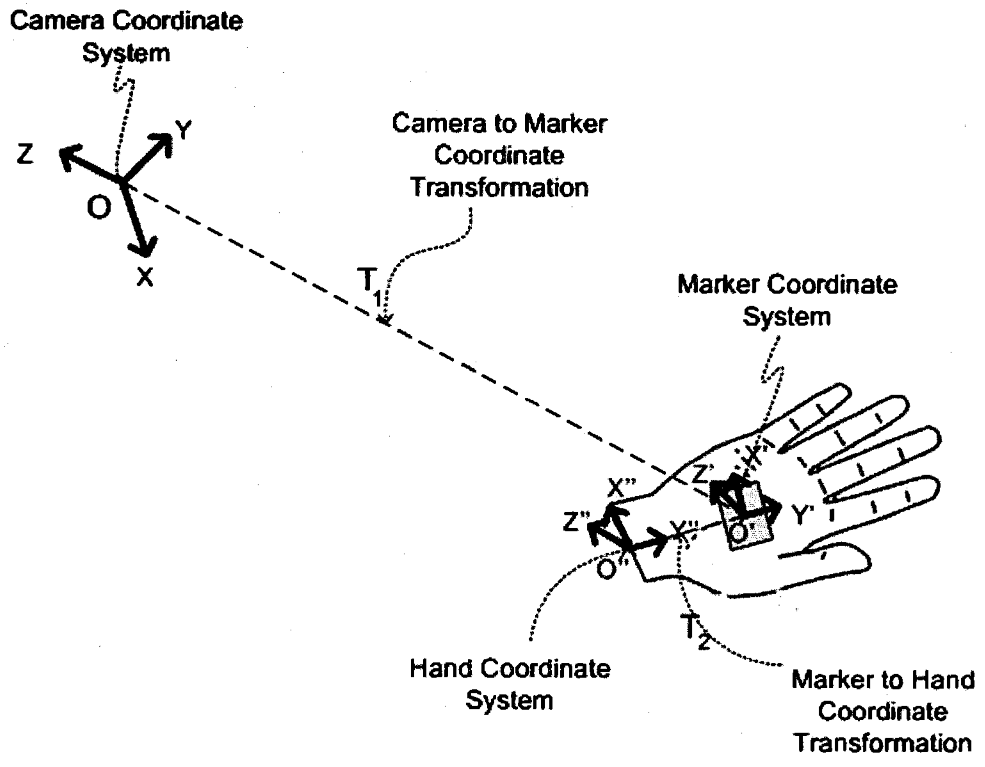


Figure 13: Camera to Marker Coordinate Transformation

Knowing the palm marker dimensions and the camera's calibrated focal distance (associated with pin-hole model), it is possible to determine the camera to marker transformation T_1 from the location of the marker corners on the image plane [26].

The geometry of the hand marker orientation is illustrated in Figure 14.

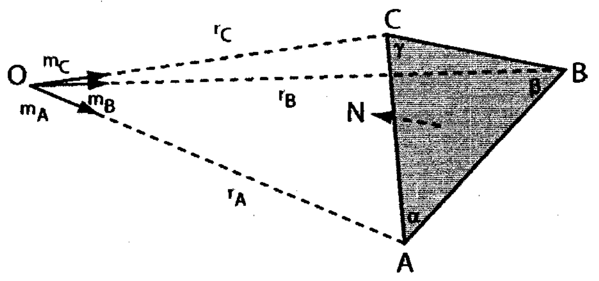


Figure 14: Hand Extrinsic Attribute Calculation

Hereafter, we use the operator ⊗ to denote a vector cross product operation, and the operator • to denote a vector dot product operation. The unit vectors m_A, m_B, m_C, called N-vectors [26] represent the trace of labels corners A, B, C as seen from the camera viewpoint and are given by (6)

$$m_A = \frac{(x_A, y_A, f)}{\|x_A, y_A, f\|}, m_B = \frac{(x_B, y_B, f)}{\|x_B, y_B, f\|}, m_C = \frac{(x_C, y_C, f)}{\|x_C, y_C, f\|}, \tag{6}$$

, where *f* is the camera focal length and *(x, y)* is the pixel coordinates of the point with respect to the camera principal point. The position of the corners *A, B, C* in 3D is given by (7).

$$OA = m_A r_A,\ OA = m_B r_B,\ OA = m_C r_C \tag{7}$$

The plane normal N is given by the cross product $AB \otimes BC$ (8).

$$N = AB \otimes BC = (r_B m_B - r_A m_A) \otimes (r_C m_C - r_B m_B) \tag{8}$$

The orientation problem is solved by determining the depths r_A, r_B, r_C using the geometrical constraints of the marker, given in equations (9) and (10).

$$\begin{aligned} \|AB\|^2 &= r_A^2 + r_B^2 - 2r_A r_B (m_A \bullet m_B) \\ \|BC\|^2 &= r_B^2 + r_C^2 - 2r_B r_C (m_B \bullet m_C) \\ \|CA\|^2 &= r_C^2 + r_A^2 - 2r_C r_A (m_C \bullet m_A) \end{aligned} \tag{9}$$

$$\begin{aligned} \cos(\beta) &= r_B r_C (m_B \bullet m_C) + r_A r_B (m_A \bullet m_B) - r_A r_C (m_A \bullet m_C) - r_B^2 \\ \cos(\alpha) &= r_A r_B (m_A \bullet m_B) + r_C r_A (m_C \bullet m_A) - r_C r_B (m_C \bullet m_B) - r_A^2 \\ \cos(\gamma) &= r_C r_A (m_C \bullet m_A) + r_B r_C (m_B \bullet m_C) - r_B r_A (m_B \bullet m_A) - r_C^2 \end{aligned} \tag{10}$$

The geometry is simplified using a square marker. In this case *AB* = *BC* = L and $\beta = 90°$. Substituting in (9) and (10) we get:

$$L^2 = r_A^2 + r_B^2 - 2r_A r_B (m_A \bullet m_B) = r_B^2 + r_C^2 - 2r_B r_C (m_B \bullet m_C) \tag{11}$$

$$0 = r_B r_C (m_B \bullet m_C) + r_A r_B (m_A \bullet m_B) - r_A r_C (m_A \bullet m_C) - r_B^2 \tag{12}$$

Two quadratic equations in r_A/r_B, r_C/r_B can be obtained; and two possible solutions for r_A, r_B, r_C are obtained. We can assume, without loss of generality, that *AB* is the x-axis, *BC* is the y-axis and *B* is the origin of the features coordinate system, the feature to camera transformation T_1 is given by equations (13), (14)..

$$\begin{gathered} T_1^{-1} = \begin{bmatrix} \hat{X} & \hat{Y} & \hat{Z} & OO' \\ 0 & 0 & 0 & 1 \end{bmatrix},\ h = OO' = r_B m_B, \\ \hat{X} = \frac{(r_B m_B - r_A m_A)}{\|r_B m_B - r_A m_A\|},\ \hat{Y} = \frac{(r_C m_C - r_B m_B)}{\|r_C m_C - r_B m_B\|},\ \hat{Z} = \hat{X} \otimes \hat{Y} \end{gathered} \tag{13}$$

$$T_1 = \begin{bmatrix} R^T & -R^T OO' \\ 0 & 1 \end{bmatrix}, R = \begin{bmatrix} \hat{X} & \hat{Y} & \hat{Z} \end{bmatrix}, RR^T = R^T R = I \qquad (14)$$

The marker to the hand model root transformation T_2 is considered the identity matrix during the simulation and is determined during the camera system.

3.2.2.2. Finger Posture

The detected fingertip markers are used to determine a linear segment reachable by the finger, along the camera view direction through 2D feature on the camera plane, as shown in Figure 15.

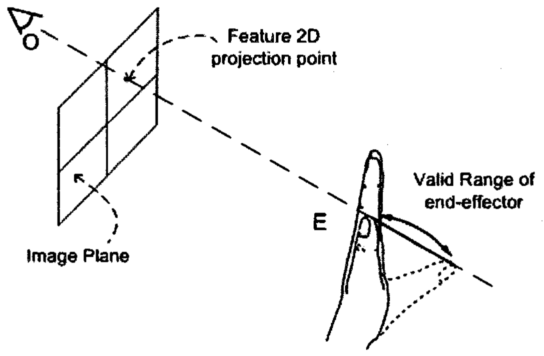


Figure 15: Detecting Feasible Fingertip Range

The camera's intrinsic parameters, namely the focal length and the principal point, are used to define the hypothesis of the fingertip position in 3D (end-effector *E*) from the projected marker position. The finger length determines the start and end of the linear segment. Features which are not reachable by the finger are rejected as noise. The reachable linear segment is sampled at constant lengths to calculate a finger posture hypothesis at each sample point using inverse kinematics. Inverse kinematics of the thumb is performed by a binary search of a lookup table of all feasible end-effector positions using 0.1 radians granulation of the thumb 4-DOF, sorted by the distance of the end-effector to the thumb's base. The other fingers IK is solved using an error model analysis technique.

3.2.2.2.1. Finger Inverse kinematics

The finger inverse kinematics (IK) is a key part of our vision-based hand tracking technique. An efficient and highly accurate finger inverse kinematics is required to achieve real-time performance. Generic approaches for solving under constrained inverse kinematics usually require an iterative optimization technique and an auxiliary objective function [27]. The kinematics of the fingers is defined by equation (15), where X is the 3D end-effector and Θ is the 4-DOF vector.

$$X = F(\Theta) \tag{15}$$

A first order approximation of the derivative of X w.r.t. Θ is given by the Jacobean matrix *J*, which is the multidimensional extension to differentiation of a single variable.

$$\delta X = J(\Theta)\delta\Theta \tag{16}$$

Since inverse kinematics are solving for Θ, given *X*, they can be solved iteratively using (17), given an initial guess Θ_0 and converging displacement δX.

$$\delta\Theta = J^{-1}(\Theta)\delta X \tag{17}$$

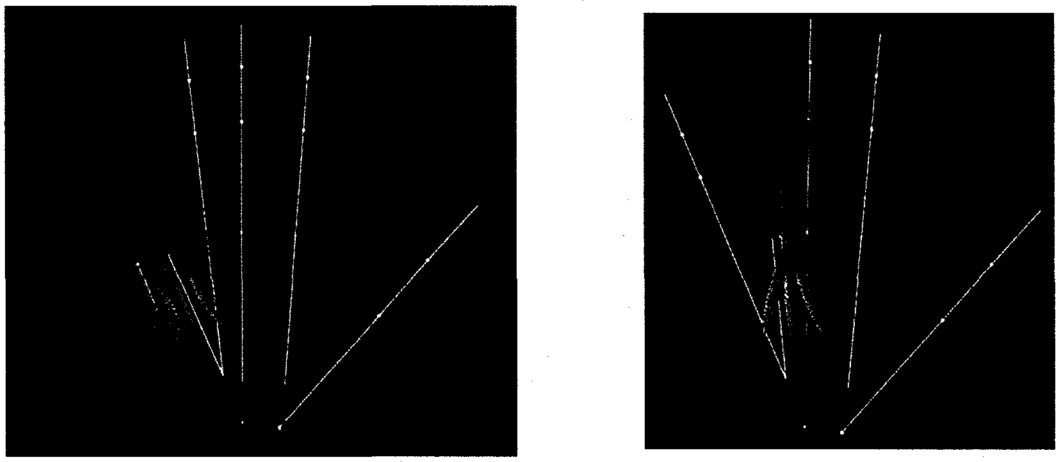
In cases where the Jacobean matrix is not square, a pseudo inverse may be required. The disadvantage of the Jacobean method is that response time depends on the initial guess, and the Jacobean matrix can be singular for some values of Θ.

A more direct approach is to use a table lookup ([30], [31]), where all possible postures are stored together with their associated end-effector vector. The table is searched for the nearest N end-effector vectors in the table and the average, of the corresponding postures, is provided as a solution. The table lookup method is very efficient but it does not scale well as the table granulation is decreased. In this case the table size grows exponentially and the search time increases accordingly.

Some researchers utilize the correlation between the PIP and DIP joint values, particularly in the free hand motion case, to reduce the finger DOF from 4 to 3, making the finger IK problem properly constrained. Initially, a simple factor ($\Theta_{DIP} = 2/3\ \Theta_{PIP}$) was used, (e.g. [27], [28]). Later, a better approximation of the correlation relationship was expressed as an implicit function (18), where d denotes the end-effector distance normalized by the finger length.

$$\frac{\Theta_{DIP}}{\Theta_{PIP}} = 0.23 + 1.73d + 1.5d^2 \tag{18}$$

Some researchers (e.g. [28]) isolate the finger abduction from the flexion motion, assuming that the fingers are flexing in a plane. This assumption requires that all flexion joint are arranged consecutively preceded by the abduction DOF in the articulated chain. Our analysis of the finger articulation (cf. Figure 16) showed that a more realistic joint configuration will require that the MCP flexion precede the abduction in the articulated chain.



(a) (b)

Figure 16: MCP joint order, (a) Flexion, Abduction, (b) Abduction, Flexion

Figure 16(a) shows the case when abduction follows the MCP joint in the link chain. It shows that the end-effectors do not lie on the same plane, for a given abduction angle, unlike the assumption of most researchers in the literature survey. Figure 16-b shows the case when MCP follows the Abduction joint in the link chain. Although the end-effectors lie on a plane, for a given abduction angle, the postures are unnatural. This means that the abduction and MCP joint angles are coupled and should be determined simultaneously.

We propose a direct approach for solving finger inverse kinematics by utilizing the domain specific constraints of the human fingers. Our approach does not isolate the finger abduction from the other 3-DOF of the finger. In other words, it solves the finger's 2-DOF MCP joint simultaneously avoiding the unnatural hand model configuration. The 4-DOF finger inverse kinematics problem is split into two consecutive and directly solvable 2-DOF problems. The corresponding error model of the simple problems is analyzed and used to compensate for the error. Particularly, we introduce a companion IK problem by assuming the finger has only one link with length equal to the end-effector distance and 2-DOF for MCP flexion and Abduction. The error due to the above assumption is analyzed a priori and the error model is used to provide accurate solutions to the MCP and abduction. The PIP and DIP values are then calculated, given the determined MCP and abduction values from the previous step in the same way. In other words, we solve the 2-DOF MCP joint using the direction of the end-effector and we parameterize the error of the first estimate by the end-effector distance. We use the parameterized error model to correct our first estimate. We then solve for PIP and DIP using the direction of their end-effector vector and again parameterize the solution error by the end-effector length and use the parameterized error model to correct our estimate. Table 4 shows the pseudo code of the IK algorithm.

Table 4: Finger IK Algorithm

```
1. Calculate r, the length of the end-effector vector
2. If r > finger length
     Set $\Theta_{PIP} = \Theta_{DIP} = 0$
     Set r = finger length
     Calculate $\Theta_{MCP}$ and $\Theta_{Abd}$ using their Error models
   else if r < min finger reach
     Set $\Theta_{MCP}$, $\Theta_{PIP}$ and $\Theta_{DIP}$ to their maximum limits
     Set $\Theta_{Abd} = 0$
   Otherwise
     Calculate $\Theta_{MCP}$ and $\Theta_{Abd}$ using their Error models
  3. Let the vector $R_1$ between PIP joint and the fingertip
  4. Calculate $r_1$ = length ($R_1$)
  5. if $r_1$ > (finger length - proximal phalanx length)
       Set $r_1$ = (finger length - proximal phalanx length)
       Set $\Theta_{DIP} = 0$, and calculate $\Theta_{PIP}$ using its Error model
     else if $r_1$ < min reach value
       Set $\Theta_{PIP}$ and $\Theta_{DIP}$ to their maximum limit
     Otherwise
       Calculate $\Theta_{PIP}$ and $\Theta_{DIP}$ using their Error models
```

The error model is created using all feasible postures obtained using motion capture or by using a model for the physical constraints. The MCP and abduction angles, resulting from solving the companion problem, are used as an estimate for required MCP and abduction angles. The estimation error is calculated by subtracting the estimate from the reference values used to generate the end-effector. The estimation error model is created by analyzing the relation between the value of the error as a function of the distance between the end-effector position and MCP joint position.

In our notation, $\hat{\Theta}$ indicate an estimate value, Θ indicates the corresponding reference value, and $\bar{\Theta}$ indicate the corrected value. Figure 17 shows the MCP error in relation to the end-effector distance (r) from the MCP joint. It can be seen that the error is highly correlated with the distance r and it can be approximated by a function *F(r)*, as shown in equation (19). A trend line is used to represent the function *F(r)*.

$$\hat{\Theta}_{MCP} - \Theta_{MCP} = \mathrm{F(r)} \tag{19}$$

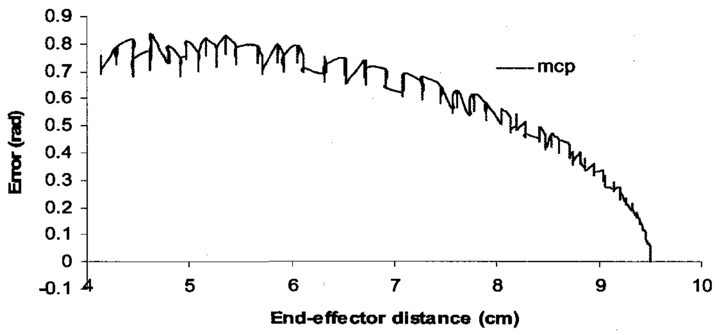


Figure 17: Error model of MCP estimate

Figure 18 shows the error model of the abduction angle for all end-effectors generated at different abduction values. It can be seen that the error is highly correlated with the distance *(r)* and we also realized that it is proportional to the abduction angle. We thus modified the error model as shown in equation (20).

$$\hat{\Theta}_{Abd} - \Theta_{Abd} = \Theta_{Abd} G(r) \tag{20}$$

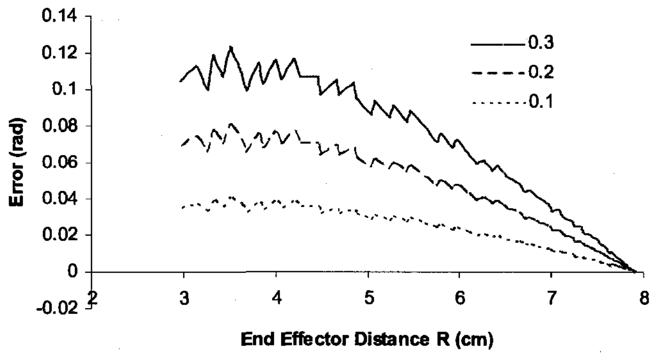


Figure 18: Error model of Abduction estimate at 3 values of reference abductions

The calculated statistics of the MCP and Abduction angles are given, from equations (19) and (20), by equations (21) and (22).

$$\overline{\Theta}_{MCP} = \hat{\Theta}_{MCP} - F(r) \tag{21}$$

$$\overline{\Theta}_{Abd} = \frac{\hat{\Theta}_{Abd}}{1 + G(r)} \tag{22}$$

After applying the above statistics, we get more accurate results. Figure 19 shows the probability distribution (histogram) of the MCP error. It can be shown that the error is in the range [-.06, 0.4] radians with 90% confidence. Figure 20 shows the probability distribution (histogram) of the abduction error. It can be shown that the error is within ±0.018 radians with 90% confidence.

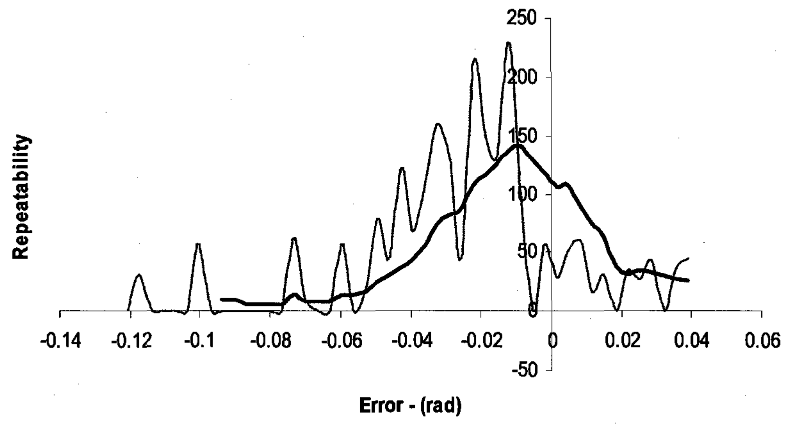


Figure 19: MCP Error Distribution function

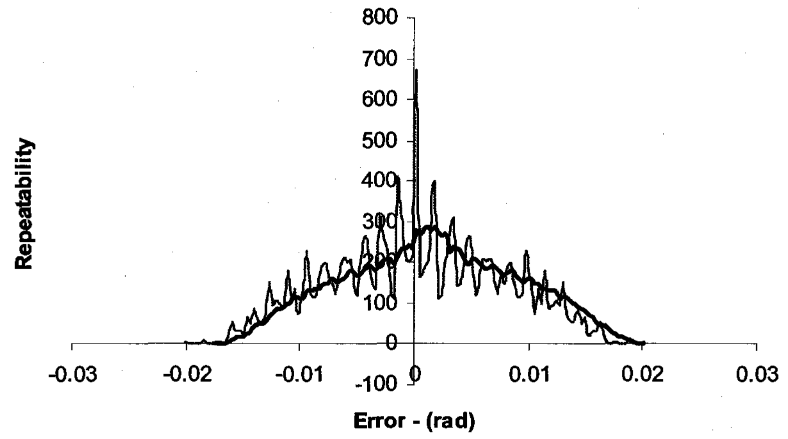


Figure 20: Abduction Error Distribution function

The PIP and DIP are calculated using a similar approach by considering the distance r_1 between the PIP joint and the end-effector position. Error models for PIP and DIP are derived as a function of r_1 and used to correct for the estimation error. Figure 21 shows the resulting error models for PIP and DIP joints. We noticed an increase in the error as the joint gets further in the articulated chain, but this effect is expected since the effect of the error on the overall structure decreases in the same way.

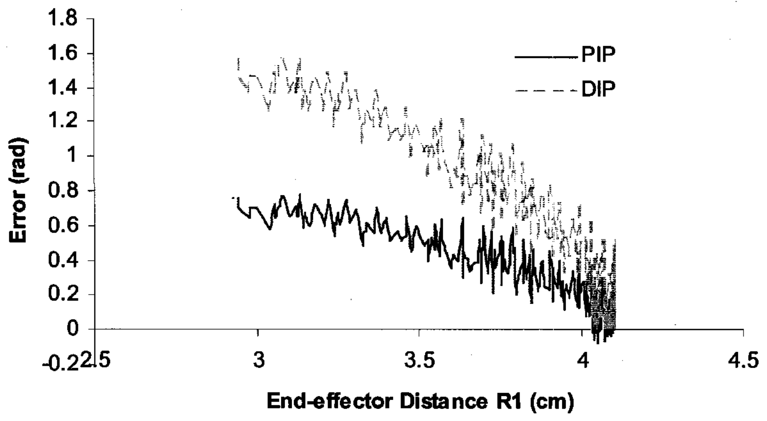
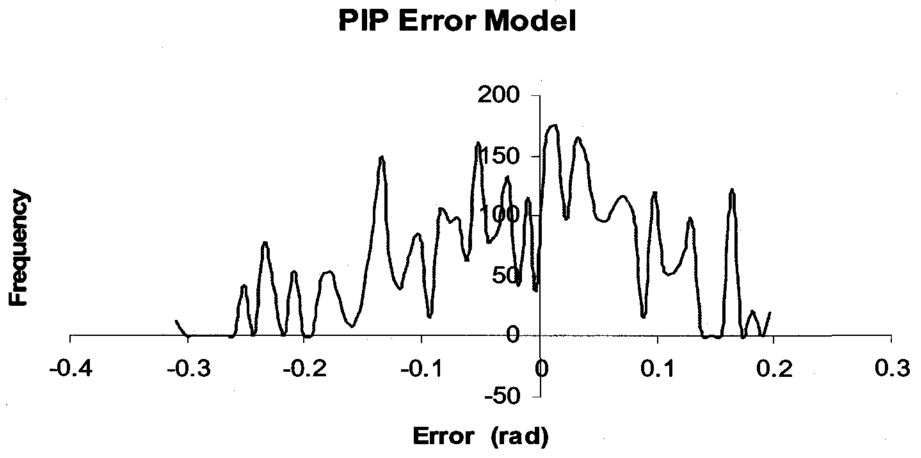


Figure 21: Error model of PIP & DIP estimates

The error distribution for the PIP and DIP joints are shown in Figure 22.



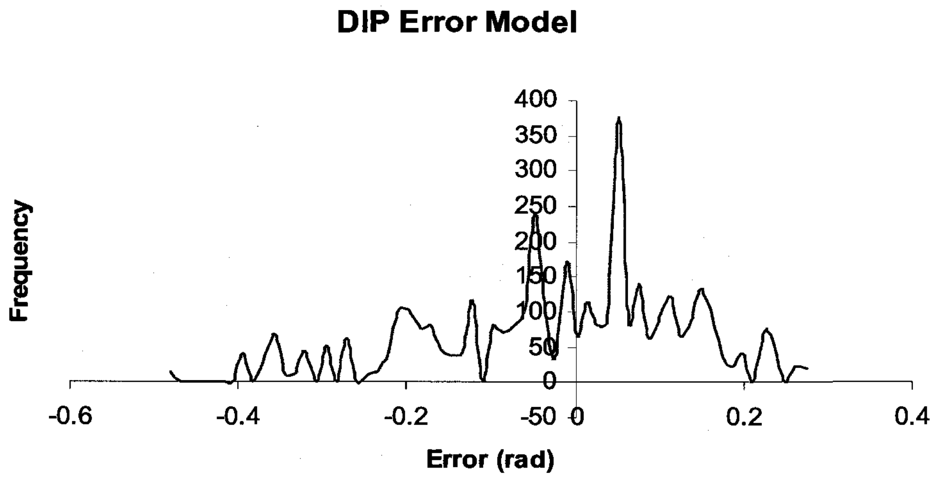


Figure 22: Error Distribution for PIP and DIP joints

It can be shown that the error of the PIP and DIP estimation is in the range [-0.26, 0.2] and [-0.4, 0.23] radians with 90% confidence, respectively. Observing the error distribution of the different joints, we noticed that the error distribution of the MCP and abduction lies in a much smaller range than the PIP and DIP joints. This is explained as the later calculation inherits the error incurred in the first one. This is, in fact, an efficient error distribution since the effect of error in the joints closer to the finger base is higher than the joints farther from the finger base.

We also tested the performance of our algorithm against the trivial table lookup approach. We assert that the error model is used only within the valid range of the model parameters by asserting that the end-effector distance lies between a maximum and minimum range, otherwise the algorithm tries to estimate a suitable value for the joints.

We used a table joint granulation of 0.1, 0.05, 0.03 radians.

Table 5: Hash table record format

MCP	PIP	DIP	u	v	r

Table 5 shows the record format of the lookup table. The parameters u and v are the planar coordinates of the end-effector with respect to the MCP joint, and the distance r is given by the Euclidean distance between the points (0, 0) and (u, v). The abduction angle is calculated separately as the angle of rotation of the finger plane. The table is sorted by the distance r and is searched, using binary search, for a range of records with in the range of r $\pm\ \varepsilon$, where ε is set to 0.2, 0.1, 0.06 cm, respectively. The record providing the minimum Euclidean distance from the end-effector is the resolved IK solution. The performance measures are summarized in Table 6.

Table 6: Summary of performance tests

	Err Model	Table_1	Table_2	Table_3
Granulation	0.015 *rad*	0.1 *rad*	0.05 *rad*	0.03 *rad*
$\text{Time}_{\text{Average}}$	104 *ms*	38.4 *ms*	102.6 *ms*	253 *ms*
$\text{Time}_{\text{Median}}$	100 *ms*	38.4 *ms*	75 *ms*	369 *ms*
File Size	0.224 *KB*	50 *KB*	373 *KB*	1640 *KB*

The average and the median processing times, angle granulation or accuracy, and the table size are shown. Our method is comparable with Table$_2$ (shown in the third column of Table 6) in terms of processing times, but it provides better accuracy and a huge storage gain. The trade off between processing speed, accuracy/granulation and storage size is evident in the table.

Our finger inverse kinematics method is suitable for applications that require high accuracy and limited storage capacity. Figure 23 shows that the error-model based IK algorithm outperforms the trivial table lookup.

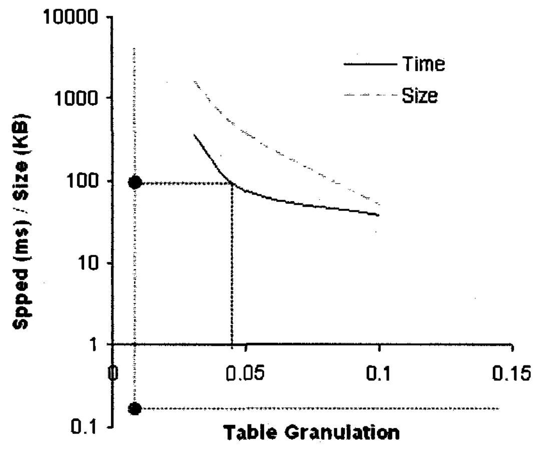


Figure 23: Performance Measure of the Error Model Based IK Algorithm

The solid line shows the average processing time, in milliseconds, of the trivial table lookup at different granulation levels. It shows that for the same processing time, our algorithm provides better estimates of the solution, and for the same accuracy, our algorithm is much faster. The dashed line shows table size in KB as a function of the table granulation. Our algorithm does not require storing large amount of table data.

Although the fingers have 4-DOF, they are in fact closer to 3-DOF than 4-DOF due to the static and dynamic constraints. The thumb, on the other hand, has fewer constraints than the ordinary finger and can reach farther. The thumb's inverse kinematics is thus under constrained and there are many joint configurations reaching the same end-effector position. The error model analysis provides an interval bound for the CMC and abduction joint angles as function of the end-effector distance from the CMC joint as shown in Figure 24. As in the case with fingers, the feasible thumb postures can be obtained using motion capture or by modeling its constraints. The error bounds, as function of r, can be used to speed up the table lookup for the IK solution.

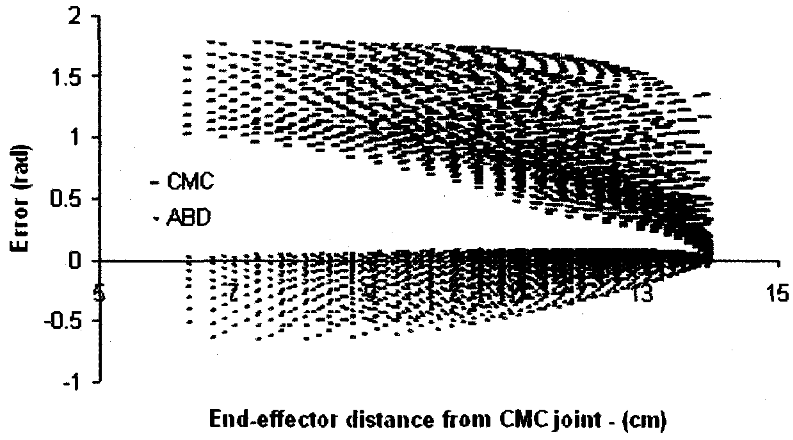


Figure 24: Error model of thumb CMC & ABD estimates

3.2.3. Posture Prediction

The prediction of future postures from the previous ones is the main task of the posture predictor. This is designed as a critical section in the software system, which means multiple processes can access the module asynchronously. It is used particularly by the gesture recognition module which may acquire posture data at a higher rate than the tracking speed. Pattern based predictors, such as Kalman filters, Hidden Markov Models, Bayesian Networks, Linear and Nonlinear Regression, and Neural Networks undergo a training phase, where the underlying patterns are learned from the training samples. These methods tend to generalize the prediction based on an assumption of an underlying model. For example HMM assumes the existence of some hidden states and that the transition from one state to another is bound to a fixed probability. The Kalman filter assumes normal distribution of the observation noise. But we would like to remain faithful for the measured data and be able to predict future ones. Since the hand posture is unpredictable and can undergo large variations in small time, we propose a damped polynomial extrapolation to estimate future postures from previously calculated ones. The memory of the system is modeled using exponential damping factors to avoid unstable extrapolation, especially in higher order polynomials. We allow polynomial extrapolation, up to the N^{th} degree, by storing the latest (N+1) calculated values for each DOF in a queue. The damped polynomial of ith degree is given recursively, from damped polynomial of lower degree by equation (23).

$$\begin{aligned} \wp_i(t) &= e^{-\frac{(t-t_0)}{T_i}} P_i(t) + (1 - e^{-\frac{(t-t_0)}{T_i}})\wp_{i-1}(t) \\ \wp_0(t) &= v_0 \end{aligned} \tag{23}$$

v_0 is the latest value in the queue, which is detected at time t_0. $P_i(t)$ is the extrapolating polynomial of the ith degree, and $\wp_i(t)$ is the corresponding damped polynomial of the same degree. T_i is damping period for the ith degree polynomial. It determined empirically and obeys the condition (24).

$$T_i < T_j \quad \forall \quad i > j \tag{24}$$

The damping coefficient $\delta_m(t) = e^{-\frac{t}{T_m}}$ satisfies the following characteristics:

- $\delta_m(0) = 1 \rightarrow \wp^m(0) = v_0 \;\; \forall \;\; m$
- $\delta_m(\infty) = 0 \rightarrow \wp^m(\infty) = v_0 \;\; \forall \;\; m$
- $\delta_m(t) < \delta_n(t) \;\; \forall \;\; m > n$ (Asserts that higher order polynomials are damped faster).

In other words, the damped extrapolating polynomial, of any degree, has the following characteristics, respectively:

- It honors the initial condition set by the last calculated posture,
- It converges back to the last calculated posture if tracking was terminated,
- It forgets previous postures faster than more recent ones.

Utilizing posture prediction in the posture estimation can be error prone in the case when erroneous postures find their way into the posture history queue. In other words, an erroneous record in the history table could bias the DOF-estimator to generate more erroneous values causing the tracking algorithm to diverge. When this happens we call the event a "temporary visual illusion". We found that the hand tracker quickly recovers when the camera has a better viewpoint to the object. Arguably, visual illusion happens to all humans, but they recover from it, given better view point to the observed object.

Kalman filters traditionally used to filter noise in similar circumstances can be used but is ill posed, *in our opinion*, because we acknowledge that hand gestures, which represent paths in the highly dimensional posture space, are spontaneous and unpredictable. Any attempt to predict hand postures based on previous experiences (static) will restrict the posture prediction and render spontaneous postures irrelevant.

3.2.4. Posture Estimation

The posture estimation module is the final step in the hand tracking system, where the most probable posture, from the set of posture hypotheses, is selected. It provides an objective function for posture estimation based on three factors, namely the canonical probability of the posture, the deviation of the estimated fingertip position from the observed feature position, and closeness of the posture estimation to the posture prediction.

In the absence of any artifact to qualify a posture hypothesis, the system must rely on intrinsic parameters. The assumption is that the best hypothesis will conform to the canonical probability of the postures, which are measured in similar situations. It will also conform to the motion dynamics as estimated by the posture prediction module. Finally, it will also conform to the only observation available to the system, the projection of the fingertip on the image plane.

Firstly, the canonical probability of the posture is obtained dynamically during simulation. It is initialized to equal probability of all postures and dynamically updated using an accumulated histogram of the estimated postures. It is considered an external parameter of the tracking system that belongs to the context of use of the system.

Secondly, the inverse kinematics algorithm filters the feasible posture within the Feasible Fingertip Range (cf. Figure 15). We allow some tolerance for error in the hand orientation calculations. For that reason, we use the distance between the projected fingertip hypothesis and the detected marker location as another factor in the hypotheses qualification. It is modeled using a Gaussian distribution with zero mean. The variance is set inversely proportional to the distance from the camera – to accommodate errors in hand orientation.

Thirdly, the predicted posture provides a reference to the hand dynamics that ensures smooth posture estimation. The closeness of the posture to our prediction, assures that the estimated posture timely conforms to the posture dynamics as modeled in the Posture

Prediction module. It is modeled as a normal distribution with mean at the predicted posture and variance as a percentage of the DOF dynamic range.

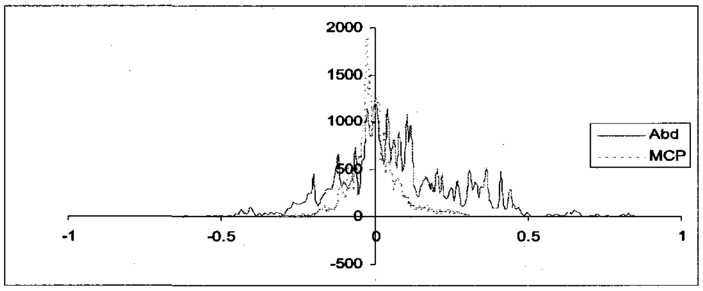
Finally, the estimated posture is thus given by equation (25), where p is a posture in the hypotheses set, Prob(p) is the canonical probability of the posture, N(d(p),σ_p) is a normal distribution of mean zero and variance σ_p, evaluated at d(p) – the 2D distance between the estimated and detected fingertip locations on the image plane. The deviation of the expected posture from the predicted posture is evaluated per attribute using normal distribution with mean zero and variance associated with the attribute.

$$p_{estimate} = \underset{p \in Hypotheses}{\arg\max} \left\{ \Pr ob(p) N(d(p), \sigma_p) \prod_{A \in postureAttributes} N((A_{hyp} - A_{predicted}), \sigma_A) \right\} \tag{25}$$

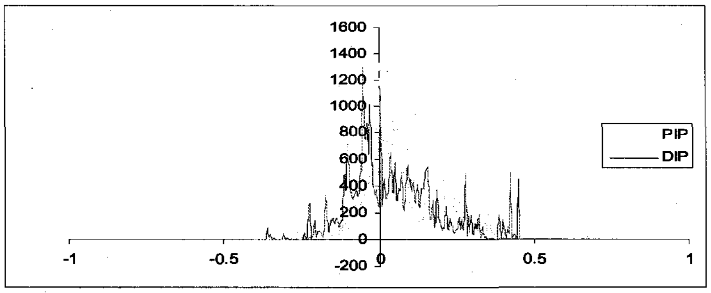
3.2.5. Results of Tracking Simulation

With reference to Figure 4, the difference between the CyberGlove acquired posture and vision-based estimated posture provides an indication about the feasibility of the vision-based approach using a single camera and a measure of accuracy of the approach in comparison to the CyberGlove.

The posture acquired from the CyberGlove was used to drive the hand model. The 3D position of palm marker corners in the model's coordinate system is constant, and the fingertips positions were calculated using forward kinematics. The projection of the 3D features on the computer display was calculated using the Model View Matrix and the (perspective) Projection Matrix, acquired in real-time, and controlled directly by zooming, panning, and scaling the model in the scene during the simulation. The histograms of the posture estimation error in comparison to the CyberGlove measurements are shown in Figure 25 and Figure 26 for the fingers and the thumb, respectively.



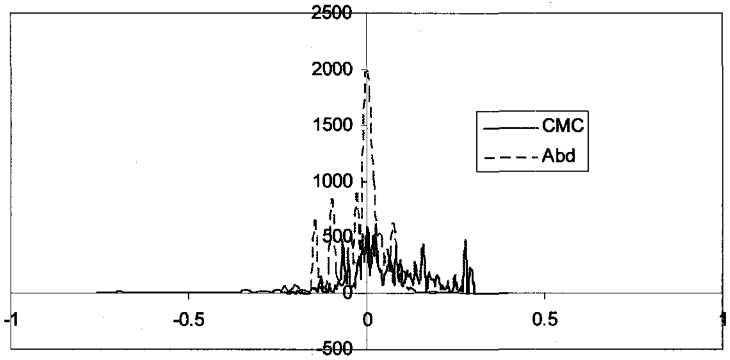
(a)



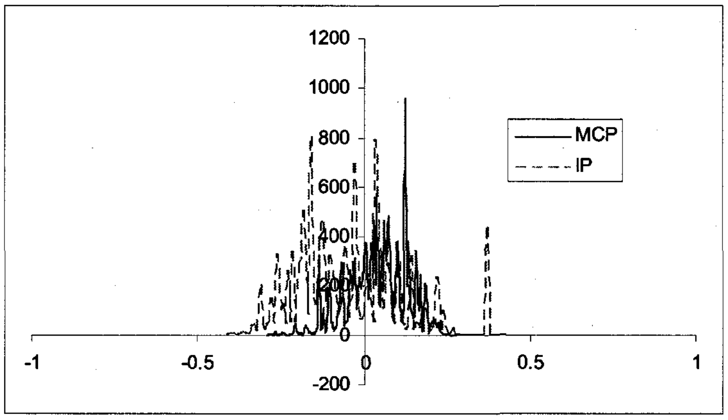
(b)

Figure 25: Statistical Error of Estimated finger DOF

The MCP flexion joint was the best joint angle estimated, whereas the MCP abduction was the worst one. The reason is the MCP joint provides the maximum impact on the fingertip position and on the same token abduction can only be reliably estimated when the fingers are mot curved when it makes the maximum impact on the fingertip position.



(a)



(b)

Figure 26: Statistical Error of Estimated thumb DOF

We argue that posture attributes that correlate multiple finger joints and work on the finger level can be better estimated using vision-based posture estimation, as explained in modeling the finger constraints. Although finger joint angles uniquely define hand postures, they suffer from interdependency and do not provide a direct means of imposing the physical constraints of the human hand.

We propose the finger curvature as an alternative posture attribute to some of the flexion finger joints. Finger curvature is defined on the whole finger and is derived implicitly from the joint angles. It can be easily visualized or made sense of its measure, and it is less error prone than the joints because it correlates multiple joints. We model the finger curvature by the reciprocal of the radius of a circle having finger as part of its circumference.

The derivation resorts to the relationship between the circumference of a circular sector and its width, which is given in terms of its radius by equation (26)

$$\mathrm{L} = \Theta\, \mathrm{r} \tag{26}$$

By definition, the curvature C = 1 / r, is the reciprocal of the radius. To relate the curvature to the fingertip distance, we calculate the linear segment, representing the distance between the fingertip and the finger base as given by equation (27) (cf. Figure 27).

$$\mathrm{d} = 2\, \mathrm{r} \sin (\Theta / 2) \tag{27}$$

We normalize the curvature measure by the finger length L, i.e. we consider $C \equiv L / r = \Theta$ – according to equation (26). The ratio between the ends of the circular sector (the end-effector distance) and the sector's circumference (the finger's length) is given, in terms of Θ, by equation (28).

$$\frac{d}{L} = \frac{\sin(\Theta/2)}{(\Theta/2)} = Sinc(\Theta/2) \tag{28}$$

Thus the normalized curvature is given, in terms of the finger length and fingertip distance, by equation (29).

$$C \equiv \Theta = \frac{L}{r} = 2Sinc^{-1}(\frac{d}{L}) \tag{29}$$

Figure 27 shows two finger curvature measures, that we found to effectively describe the finger posture and yet can be accurately estimated in vision-based hand tracking.

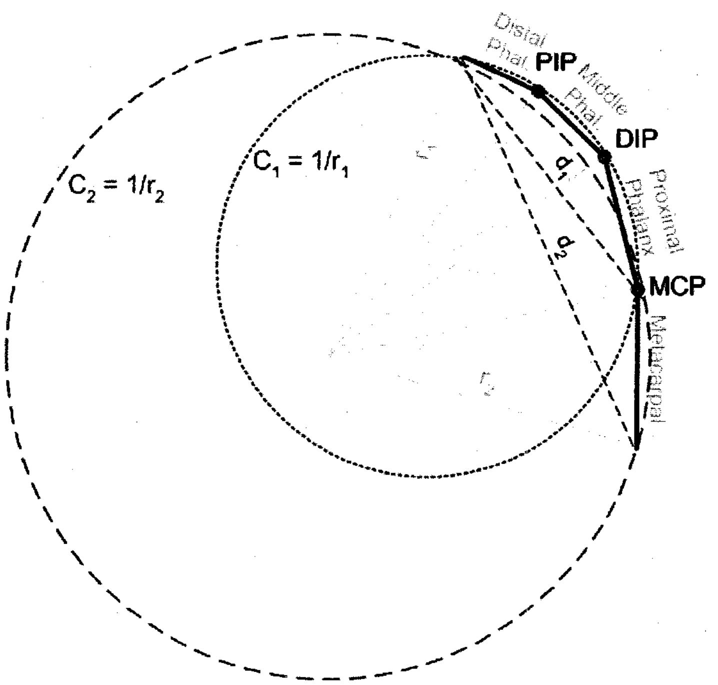


Figure 27: Fingers Major and Minor Curvatures

The major curvature (C_2) includes the finger and the palm, and the minor curvature (C_1) includes only the finger. The estimation error of the curvature errors provides better estimates than the MCP, as shown in Figure 28 – curvature is normalized by a practical maximum of 4.0 radians (vs. the theoretical maximum of 2π).

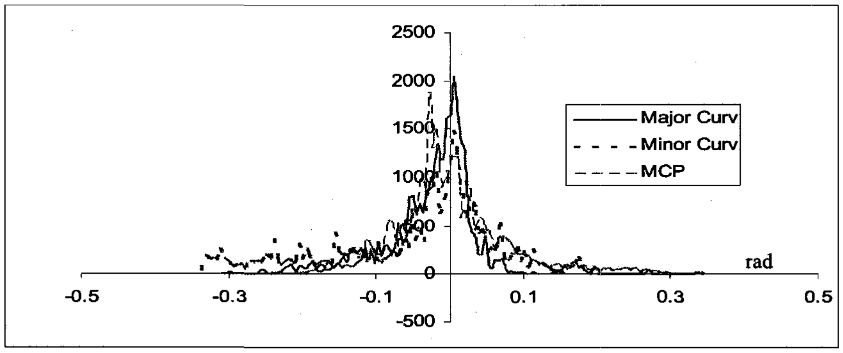


Figure 28: Statistical Error of Finger Curvature vs. MCP

Similar results for thumb are shown in Figure 29.

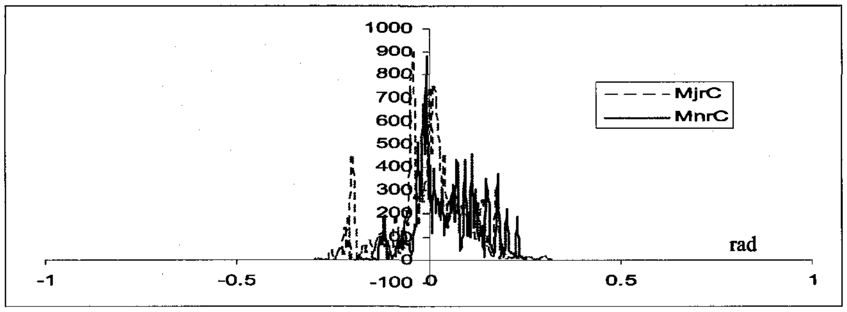


Figure 29: Statistical Error of Thumb Curvature

3.2.6. Conclusion of the Tracking Simulation

The main cause of error in simulation is the loss of depth information due to projection and the lack of visual cues to validate the posture hypothesis. The tracking problem is ambiguous with multiple possible solutions available, and hypothesis qualifications must rely on heuristic inference from previous estimations. This causes an intermittent visual illusion when the camera viewpoint is ill posed, but the system recovers quickly when a better viewpoint of the hand is provided.

The input to the tracking simulation is:

- The 2D projections of the fingertips derived from the 3D model, which is driven by the CyberGlove
- The 2D projection of the corners of a square marker positioned at the hand model's root, which is derived from the current transformation matrix, i.e. Model View matrix and Projection matrix that is changed by panning, zooming and scaling the scene during the simulation.

From these 2D features an estimate of the 3D hand posture is made.

The tracking simulation answers some of the questions we were bound to answer at the beginning of the chapter. Firstly, it shows that the vision-based hand tracking, *using a single camera*, can provide tangible measurement of the hand posture. Secondly, it provides a sense of the accuracy of the vision-based posture estimation, in comparison to the data glove. The standard deviation of the different posture attributes estimated using the vision-based hand tracking system is given in Table 7. *The fingers measurements are found very similar, thus we grouped them in one group. The thumb measurement is essentially different and is displayed separately.*

Notice improvement in the accuracy of the major and minor curvatures over the MCP and PIP joints respectively. The finger abduction estimation is not reliable, unless the finger is straight and the hand plane is almost parallel to the camera plane. The thumb's attributes accuracy is relatively higher than the fingers because its motion is less constrained, better accuracy can be obtained using another marker close the thumb's MCP joint.

Table 7: Posture Attributes Variance

Finger Type	Posture Attribute	Standard Deviation
Fingers	Abduction	0.040234
	MCP	0.005869
	PIP	0.019052
	DIP	0.023109
	Major Curvature	0.004361
	Minor Curvature	0.015301
Thumb	CMC	0.015651
	Abduction	0.006732
	MCP	0.00992
	IP	0.026515
	Major Curvature	0.008879
	Minor Curvature	0.009278

Tracking simulation provides a proof of concept of the system but suffers from the following limitations:

- Firstly, the position of the marker features, calculated from the projection of 3D features on the graphic scene, is very accurate and assumes a pin hole camera model with well defined camera intrinsic parameters; which is different from the real situation where the camera parameters are calculated by a calibration procedure and suffers from approximation errors.
- Secondly, the simulation does not include background clutter.
- Thirdly, the simulation uses the same hand model in forward and inverse kinematics and does not test the approximation error of the hand model with a real hand.
- Fourthly, the simulation does not test for extrinsic parameters such as skew and lens distortion.

- Finally, the simulation does not require a palm marker to model's root transformation, since the marker was centered on the root. In the real tracking this transformation must be calculated, during the system calibration, and this transformation may vary over time and will cause tracking error. For these reasons, we use a video camera in the next phase.

3.3. Tracking with Real Cameras

We extend our tracking procedure to the real world. We still use fingertip markers and palm markers to derive the posture hypothesis generation. We reviewed the state of the art in hand tracking, including marker-less approaches to hand tracking, and we realized that it includes a number of constraints that restrict the usability of the tracking approach in dynamic gesture recognition. These restrictions include:

- Wearing special clothes and Long sleeved clothing
- Using Colored gloves
- Uniform background
- Complex but stationary background
- Restricted movement
- Continuous movement
- Vocabulary restriction
- Unnatural signing

For example, Wu et al. [24], [98] use a divide and conquer approach , where they use an iterative algorithm to estimate the extrinsic posture attributes and then use it to estimate the intrinsic attributes and use the estimated intrinsic attributes to enhance the extrinsic attributes estimation and so forth. Stenger [14] estimates extrinsic hand attributes, given the intrinsic attributes, using a Hierarchical Bayesian Filter and the image silhouette and color likelihoods. The two approaches were demonstrated using a marker-less hand, but the iterative nature does not provide real-time response. Using color markers, we can decouple palm position and orientation estimation from finger joints estimation. Knowledge about the fingertip positions can facilitate using kinematics constraints in the posture estimation process. Solving inverse kinematics can be very efficient and requires less processing time than using an image based observation model, which requires edge detection and hand segmentation, followed by evaluating the probability of the estimated posture given the detected features. Since our paramount goal is to reduce the restriction on the hand movement, we decided to use the color markers approach as a step in the right direction, bearing in mind that many of the algorithms used can be reused in a marker-less environment. Figure 30 shows the overview of the single camera based hand tracking and posture estimation system.

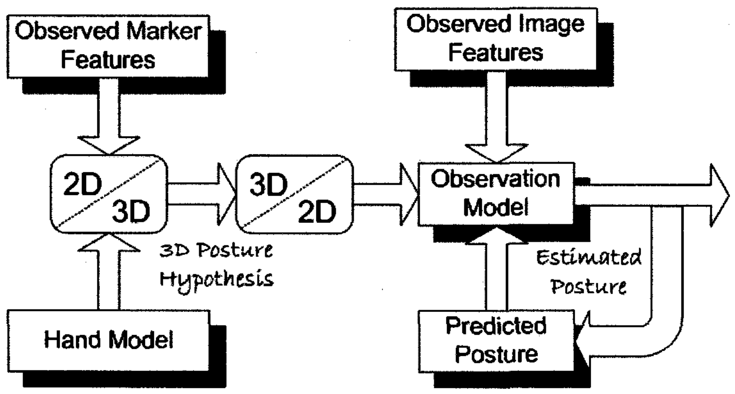


Figure 30: Posture Estimation System Overview

We reused the hand model, 2D/3D, and posture prediction modules. We added a marker feature extraction module a 3D-to-2D projection module and replaced the posture estimation module by an observation model that utilizes the acquired video frame to provide a probability measure of the posture hypothesis, given the hand silhouette and color in the image. The following sections will provide a detailed explanation of the new modules.

3.3.1. Feature Extraction

We use a dark glove with color-coded ring markers to detect the fingertips, and 2 cm square markers to derive the palm's position and orientation with respect to the camera (cf. Figure 3). We use two palm markers, one for the front and the other for the back of the palm, but more markers can be added as required. The palm markers are identical and can be distinguished using a color-coded tag.

Motion tracking systems, such as Vicon, use light emitting or infrared markers to identify important features in the image. Multiple cameras are used to detect the location of these markers in 3D space. Light emitting markers are easier to track than color markers, but require battery power. On the other hand, marker correspondence between cameras is more complicated, thus we prefer working with single cameras independently. The number of markers used and the position of the markers used in a motion tracking system like Vicon is an open question, and our experiments can provide a clue to the answer to these questions.

As part of the initial phase, we convert the color-space into the HSV (or Hue, Saturation, and Luminosity) space with 255 values for each plane. This provides the best performance in terms of identifying a very narrow variance (usually 3 or 4 levels) of color (Hue), with a small acceptable variance (larger than Hue, usually around 10) for shading (Saturation), and an even larger acceptable variance for the lighting conditions (luminosity). We initially identify all the dark colored (black glove) elements in the room. We process the identified pixels for noise by counting the number per segment (determining if a mass of pixels exist in one place, a segment being 8 by 8 pixels) and segment clustering (determine if segments join with each other). We then search for a specific marker color (in this case a specific red), passing a pixel if it falls within the specified range. We segment the candidate marker region using connectivity analysis and identify the marker using a color ID (blue for front of the hand, and yellow for the back of the hand). The aforementioned technique basically identifies the tracked hand within the image, and provides information as to its orientation. Markers of different color markers are used on the fingers in order to recognize the fingertip positions.

The marker on the back of the hand is 0.02m by 0.02m square, and we search for the marker's corners (as shown in Figure 31) in order to pass this information for transformation into the 3D domain. As the information pertaining to corner's positions is essential for an accurate interpretation of the hands orientation and distance from the camera, and as the corners as appreciated by the camera are rather fuzzy, we process this information in order to obtain the most accurate positions possible. To do this, we track the edge of the corner sequentially in the direction perpendicular to the gradient of the pixel illumination [106] (the dark glove and the bright red marker provide very high contrast in the illumination channel V). The corner is roughly determined by a pulse in the contour curvature, this information here is only used to determine the sides of the marker (i.e. each sides start and end point).

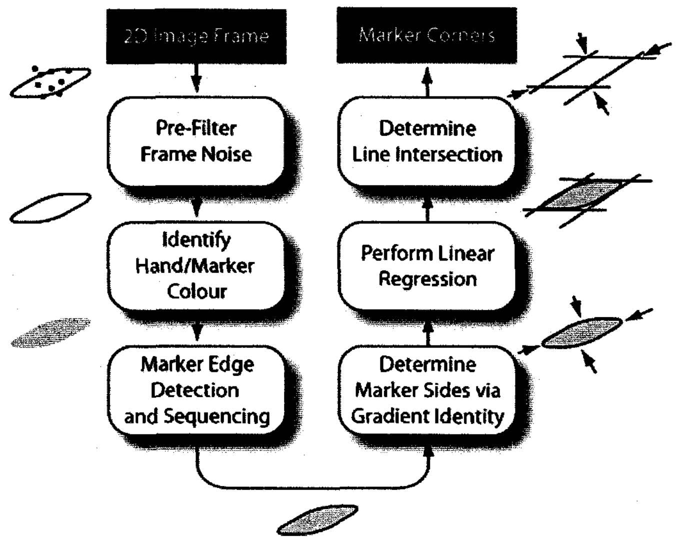

2D Image Frame
Marker Corners
Pre-Filter Frame Noise
Determine Line Intersection
Identify Hand/Marker Colour
Perform Linear Regression
Marker Edge Detection and Sequencing
Determine Marker Sides via Gradient Identity

Figure 31: Hand/Marker Identification

We then pass each side as a single implicit linear equation (A.X + B.Y + C = 0) using linear regression in order to determine A, B, and C (whereby B is fixed to -1 when A $\neq$ 0). Using the four linear equations (one for each side) the line intersect points are found (within a bounding region, as opposite lines are very rarely parallel) and used to determine the true marker corners. The corners are used to calculate the hand position and orientation in 3D space using perspective geometry [26]. Five colored ring markers are used to mark the finger tips.

The locations of the finger markers are determined as holes in the segmented hand. The holes are derived by a morphological closing operation. First, the holes in the segmented hand are closed using a repeated dilate operation; then the hand is restored using a repeated erode operation. To avoid using a large kernel size, dilate and erode operations are performed iteratively using a 3x3 kernel, where the number of iterations is set in proportion to the hand marker size. The holes are detected by masking the original hand segment from the closed hand segment. The markers are identified by their colors, where we use Hue-Saturation-Value (HSV) color space, as it effectively detects the color hue. The 3D color histogram is segmented using multiple thresholds to determine overlapping color ranges for each marker. The number of pixels of each range, normalized by the palm marker area, is used as an attribute vector for the detected color blobs. An SVM classifier is trained and used to classify the detected finger tip features using its color attributes. Figure 32 shows an example of the detected markers and the overlaid hand model.

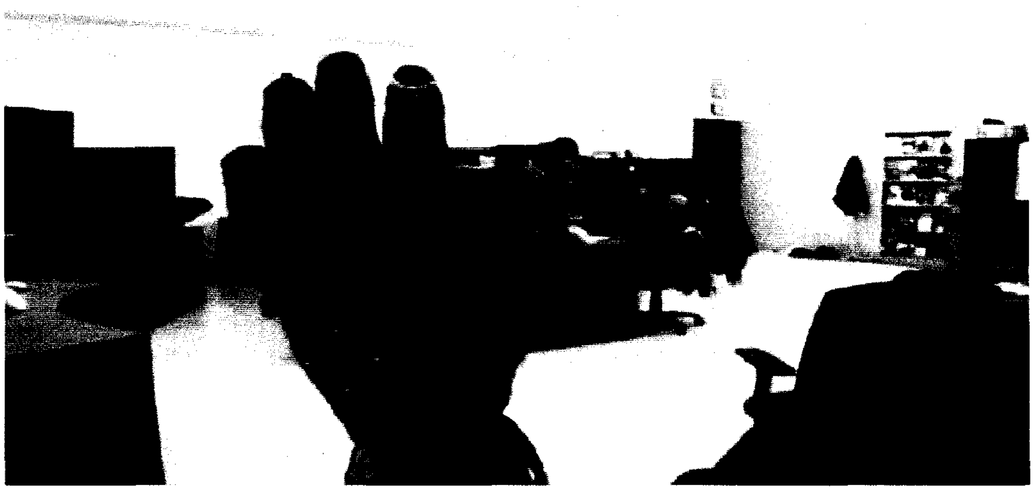


Figure 32: Tracking Glove

Table 8 shows the pseudo code of the feature extraction process.

Table 8: Pseudo Code for Feature Extraction

```
1. Segment the candidate hand ROI using the hand/glove color
2. Segment palm marker region, within the hand region using the marker color
3. Identify the palm marker using the color ID
4. Detect the position of palm marker corners, from the marker edge
5. Determine the expected thickness of the finger markers
6. Close the holes in the hand region of at most width as the expected thickness
7. Use unclosed image as a mask to isolate the marker region candidates
8. Identify the marker regions using the region color and size attributes
```

3.3.2. System Calibration

As part of the real system setup, we had to put palm markers at a location which is least deformable on the palm, yet it can provide a large work space from the camera perspective. Since the palm location does not confine to the hand model's root, a geometric transformation must be derived to translate the marker position and orientation to the hand model's position and orientation. A procedure to detect the Marker-to-Model transformation is devised using 2 markers, one is the hand marker and the other is placed at the root's position. Multiple captures (Figure 33) are used and the camera to model transformation is calculated for both markers.

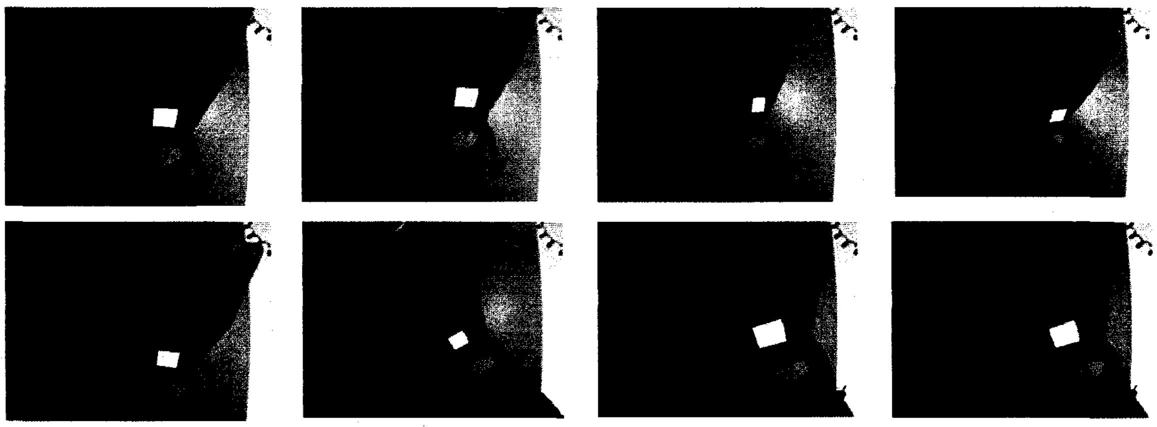


Figure 33: Hand Marker-to-Root calibration captures

At each capture, we evaluate the markers orientation w.r.t. the camera. The marker-to-marker transformation is then calculated by multiplying the inverse of the first transformation matrix by the second transformation matrix.

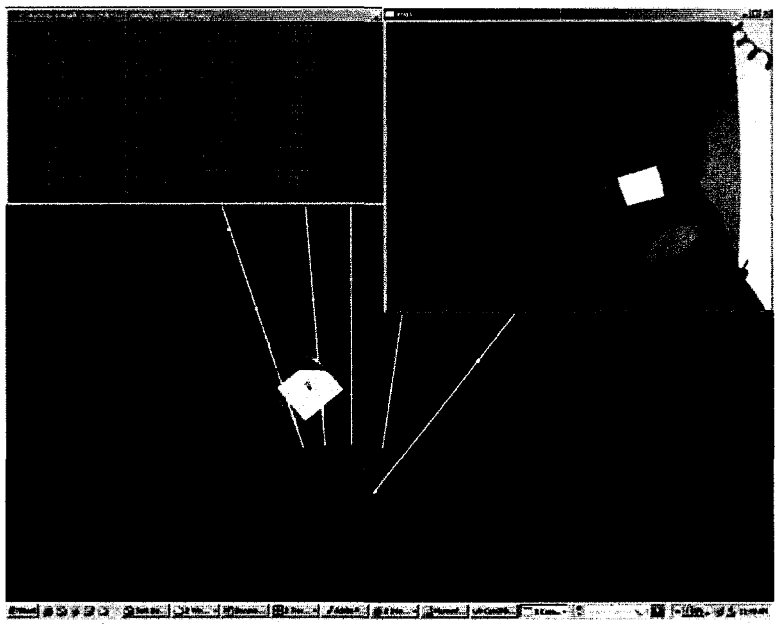


Figure 34: Camera to Marker calibration utility

While this method is ill posed, because small errors in one transformation can be amplified in the resulting transformation, we use visual correction by showing the marker position in graphics (Figure 34). Since our method can only provide 2 possible solutions for each transformation, there are 4 possible solutions for the combined transformations. Irrelevant solutions are rejected visually and multiple consistent solutions are used to filter the errors.

3.3.3. Posture Hypothesis Validation – The Observation Model

In the presence of 2D feature noise and due to the under constrained nature of the finger IK, especially the loss of depth due to camera projection, a qualifying process must be used to produce the most probable posture. During simulation, the qualification process was based primarily on our prediction of the current hand posture and how close the projected finger tips are to the detected fingertip locations. Given the images acquired using the tracking video camera; we can improve the qualifying process by utilizing color and edge features extracted from the image. We propose a probabilistic observation model utilizing the segmented hand image and its silhouette. The hand posture hypothesis is applied on the hand model using forward kinematics, and then the resulting 3D model is projected on the image plane as in Figure 35. The projected model is sampled at N points.

Figure 35: Projecting the Hand Model and Finger Edges

At each point, we calculate the probability of the pixel being a correct observation, given the color of the pixels in it neighborhood, using the glove color histogram. The 2D observation model along N sample points constructs a binomial density function – equation (30).

$$P_1 \propto p^n(1-p)^{N-n} \qquad (30)$$

Where n is the number of pixels projected on skin/glove color and p the probability of pixels projecting on skin color ($p \approx 1$).

Moreover, the corresponding finger edge, see Figure 35, is qualified using an observation functional, adopted from [33], by assuming that clutter is a Poisson process along the direction perpendicular to the edge with spatial density λ, that all edges have the same probability of being the true observation, and that the projected edge accuracy is a normal distribution with standard deviation σ. The resulting probability at a sample edge point is given by equation (31), where *q* is the probability that the true edge is not detected and *d* is the distance between the edge hypothesis the detected edge.

$$p(hyp|obs) \propto 1 + \frac{1}{\sqrt{2\pi}\sigma q\lambda} \sum e^{\frac{-d^2}{2\sigma^2}} \tag{31}$$

The summation in equation (31) is performed for all detected edges along the direction perpendicular to the model edge. The fingers are approximated by cylinders with a constant diameter and the edges of the fingers are projected parallel to the fingers kinematic links with a distance from the links corresponding to the finger thickness. Since the function is exponentially decreasing, only a small distance range is considered, beyond which the probability is insignificant. The 2D observation model along N sample points is given by the product of point probabilities as given by equation (32).

$$P_2 = \prod_{i=1}^{N} p_i(hyp|edge) \tag{32}$$

As we may detect multiple possible targets for the end-effector of a finger, and we do not know the true target feature, we must include all detected targets in one functional. We use a similar functional P_3 to equation (31) to determine the probability of the end-effector hypothesis, given all the detected 2D end-effector features. In this case, λ is the density of the 2D feature clutter, σ is standard deviation of the detected feature accuracy, q is the probability that the true feature is not detected or is occluded, and d is the 2D distance between the projected end-effector position and the detected feature position; and the summation is performed on all detected features.

The probability of the posture attribute hypothesis, given the previous estimated posture attributes P_4 is calculated as a normal distribution with mean at the predicted posture and variance as a percentage of the DOF dynamic range, as described earlier in the simulation.

Since IK is applied on the finger level and the observation features are applied on the pixel level, it is essential to include a hand-level multi-dimensional probability distribution of the hand posture P_5. This distribution can be determined from the context of use, i.e. from the gesture training data in this case, using parametric or non-parametric density estimation.

Finally, the overall observation model is given by the product of the above mentioned factors, *namely segmented hand color*, *hand silhouette*, detected *end-effector position*, previous *posture estimates*, and the context-based *posture probability*, $P = P_1 P_2 P_3 P_4 P_5$. The posture which gives optimal probability according to the observation model is provided as the best estimate.

The assumption of mutual independence of the above probability measures is justified, since every factor correspond to independent random variable: Firstly, the hand color is independent from the detected edges in a cluttered image. Secondly, in case of detecting multiple candidate fingertip markers a probabilistic model to measure the closeness of the posture hypothesis to the target end-effector is required. Thirdly, the posture history, a.k.a. the posture prediction, is independent from the detected features. Finally, the canonical probability is an external factor that filters the posture hypothesis. In other words the above probabilistic measure is proportional to the following independent variables:

- The color artifacts
- The edges and clutter
- The proximity to the right fingertip marker candidate
- The proximity to our prediction

3.3.4. Multiple Cameras

We extend the observation model to include multiple cameras. Multiple cameras are used to improve the reliability of the tracking system. Stereoscopic cameras are used to infer the depth by relating the detected features between two cameras with slightly different fields of view. In our case we use multiple cameras to improve the accuracy of the posture estimation and to improve our posture hypothesis creation.

Camera sensor fusion is done at different levels. At the first level, we do not assume knowledge of the geometrical transformations between the cameras coordinate systems, which is useful when the cameras are moving. In this case, the posture hypothesis generated using 2D features from each camera is validated using the observation models of all the cameras. The overview of the two camera system is shown in Figure 36.

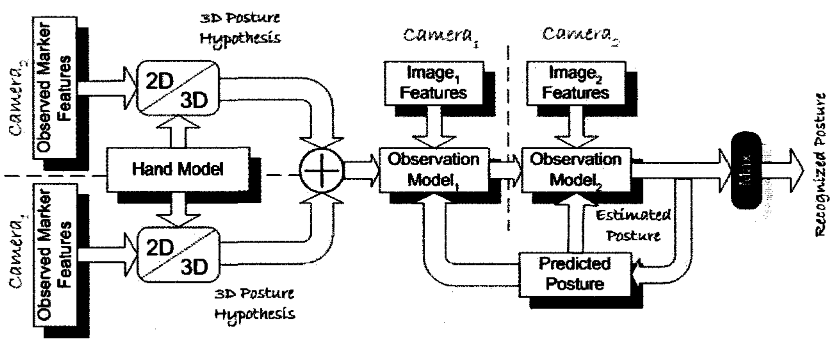


Figure 36: Two-camera extension of the observation model

At the second level of camera sensor fusion, we use the geometrical transformation between the camera coordinate frames. In this case, the detected hand position and orientation is validated and the best orientation is used by both models. The overview of the system is shown in Figure 37.

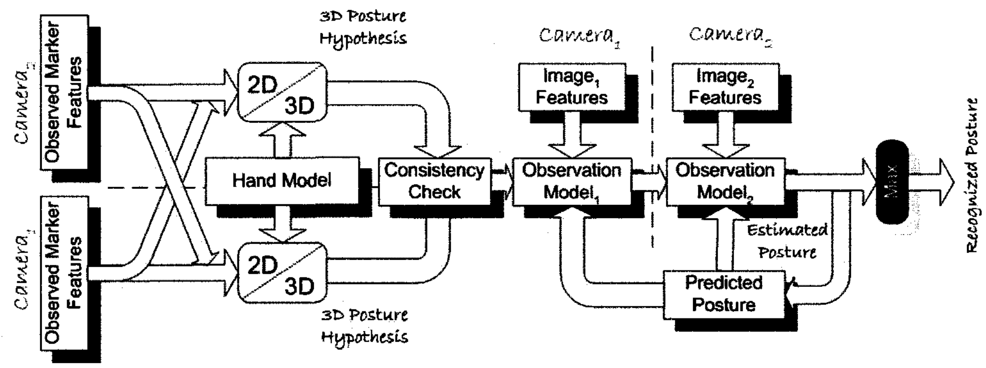


Figure 37: Two-camera system

3.3.5. Implementation issues

Figure 38 shows the class diagram of the hand tracker. The main interface is the tracker interface, which controls the process of feature extraction and posture estimation. The different classes implementing the tracker interface include the single camera tracker, and the multiple camera trackers with and without inter-camera geometric transformation. The hand model consists of five finger models, which include the finger's joints and kinematic links. The palm marker retrieves the coordinate transformation between the camera and the hand model and the finger markers are used to retrieve the fingertip position.

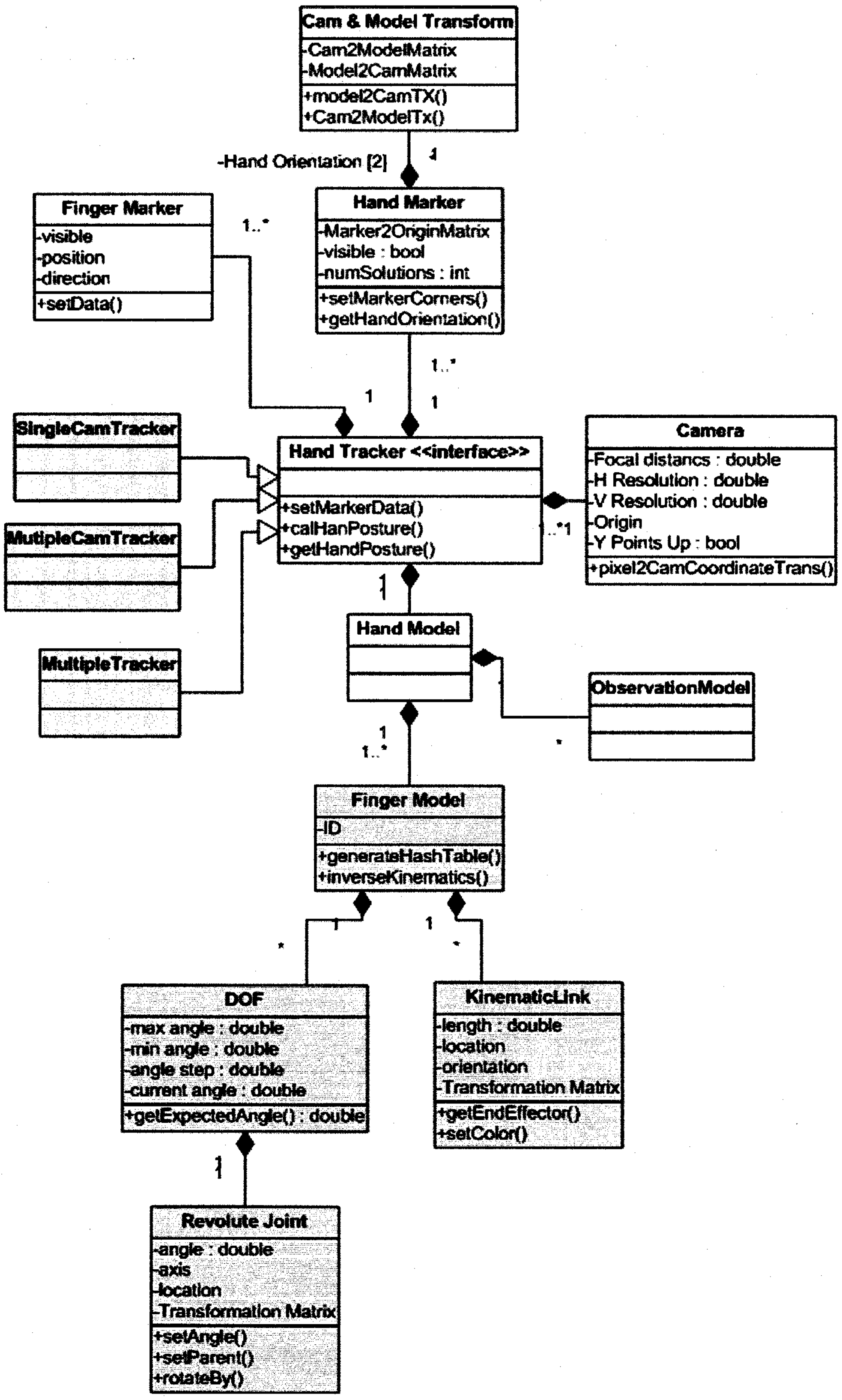


Figure 38: Hand Tracker Class Diagram

3.4. Tracking Results

Global posture attributes include the hand position and orientation in 3D. To test the calculation of hand position in 3D we used a Point Grey Dragonfly video camera with 3.5 mm lens to track the trajectory of an operator hand using a palm marker as shown in Figure 39. The operator moves his hand in predefined trajectories and the calculated hand trajectories matched the input trajectories.

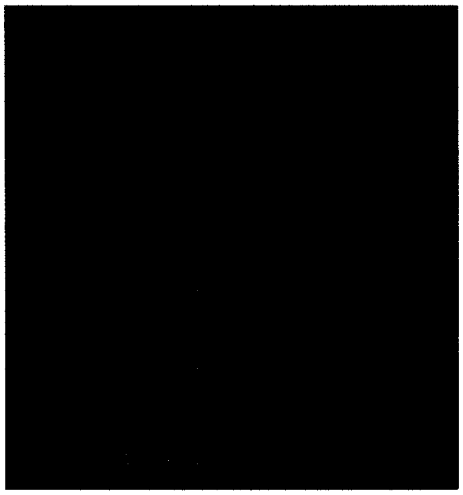
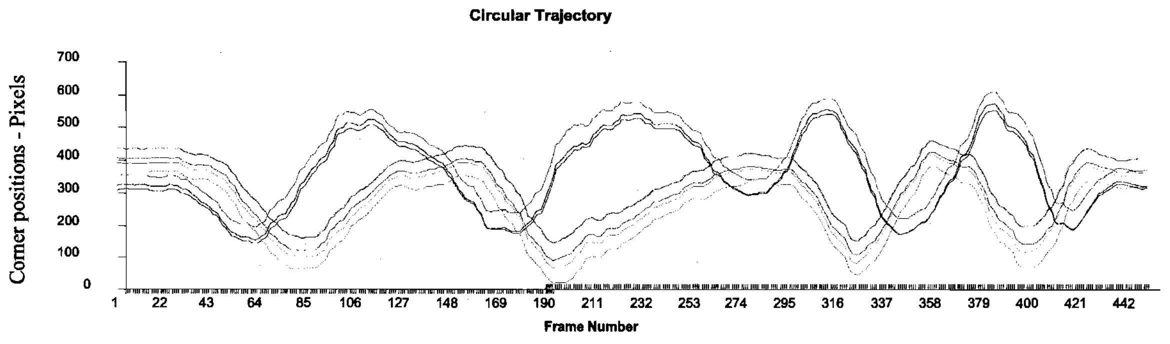


Figure 39: Hand Tracking Experiment

Figure 40 - Figure 42 show the results of the calculated trajectories. It starts with the marker's *2D corner features in pixels*, followed by the *3D trajectories* calculated by the tracker in centimeters (cm), where x-y is the projection of the trajectory on the horizontal plane parallel to the camera plane and z is the distance from the camera plane.



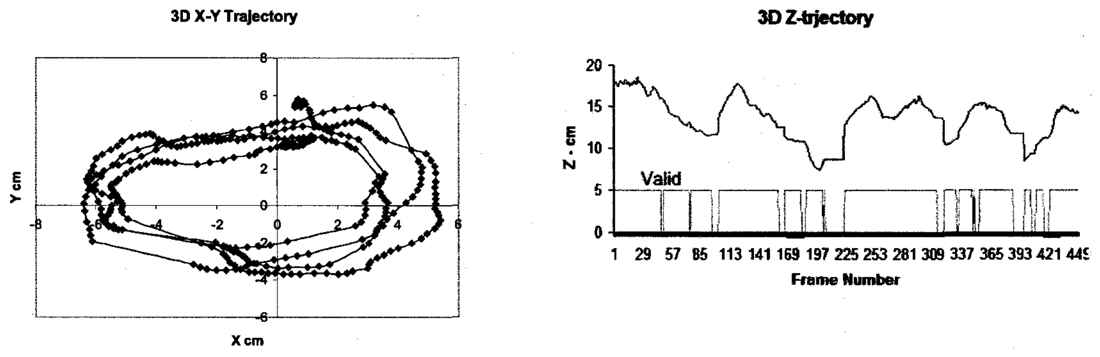
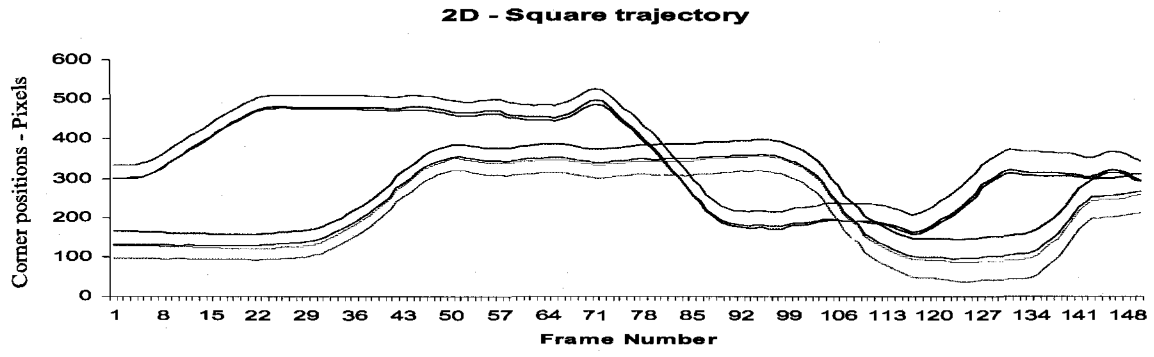


Figure 40: Circular trajectory, while moving up and down



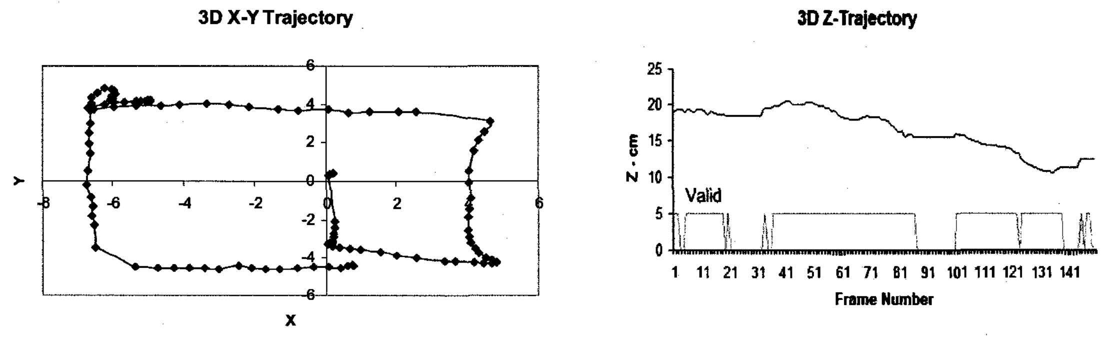
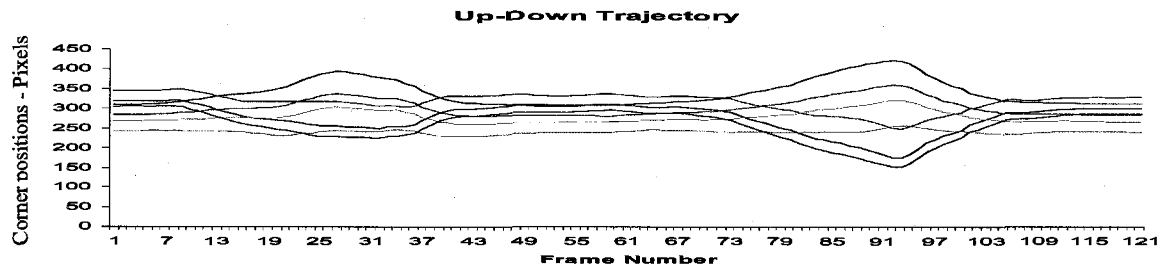


Figure 41: Rectangular Trajectory



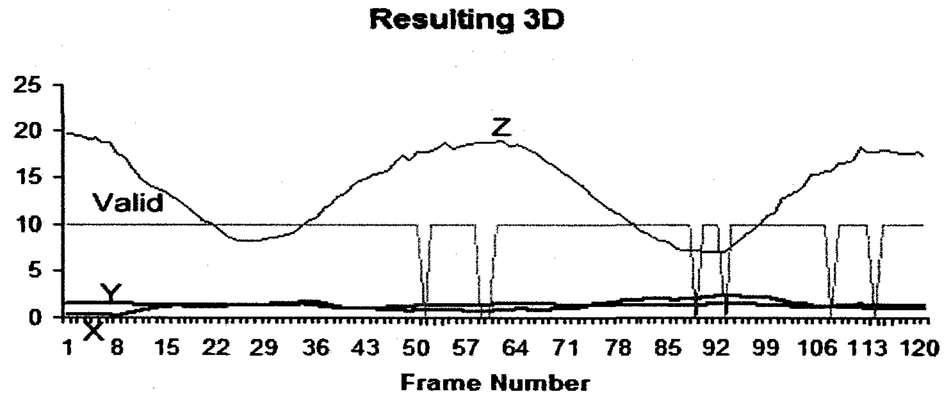
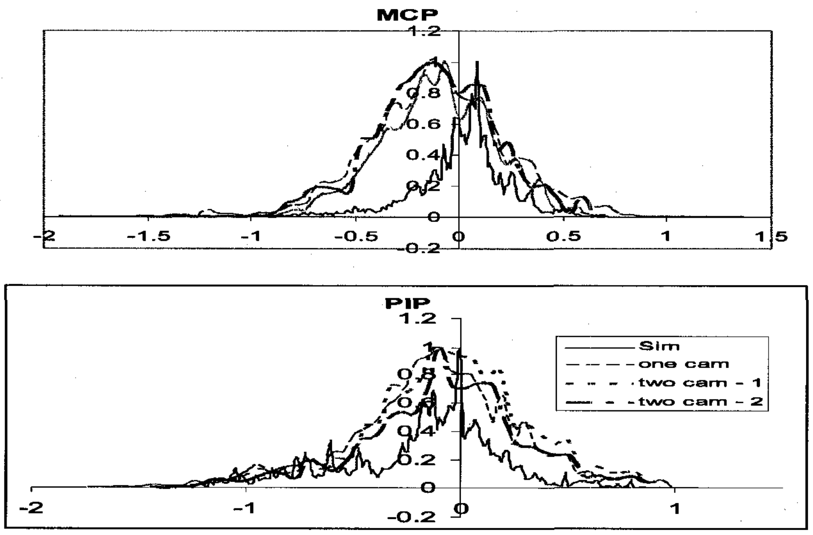


Figure 42: Vertical Trajectory

Occasionally, the hand orientation calculations fail due to lens distortion or perturbation of the location of the extracted features. The system responds by freezing the hand model momentarily. The binary signal, termed "valid", in the figures above depicts when happens. The associated frames are dropped and do not contribute to the hand tracking.

The intrinsic posture attributes includes the fingers articulation in the form of joint angles associated with the hand model. We acquired 6 video clips using two Point Grey Dragonfly cameras. The associated reference hand postures were acquired simultaneously using a CyberGlove. The error analysis of the fingers and thumb articulation using the different system configurations are shown in Figure 43 and Figure 44 respectively. Please note that all errors are shown in radians.



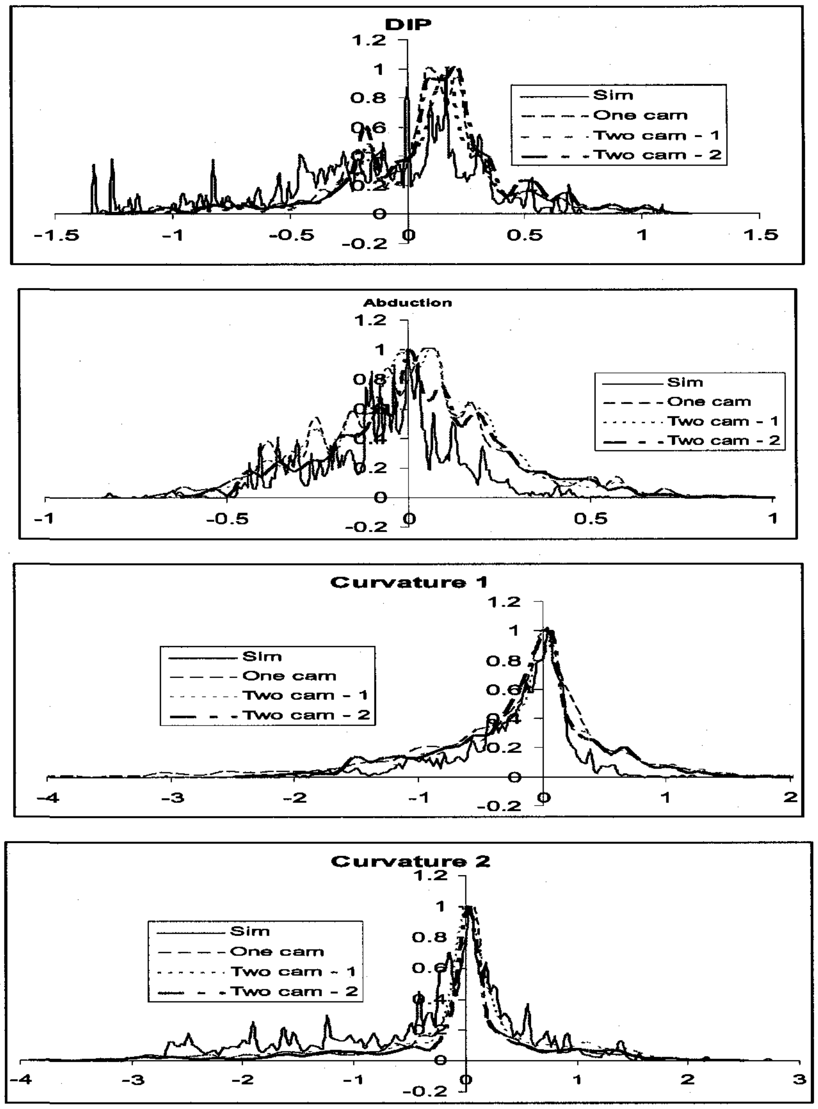
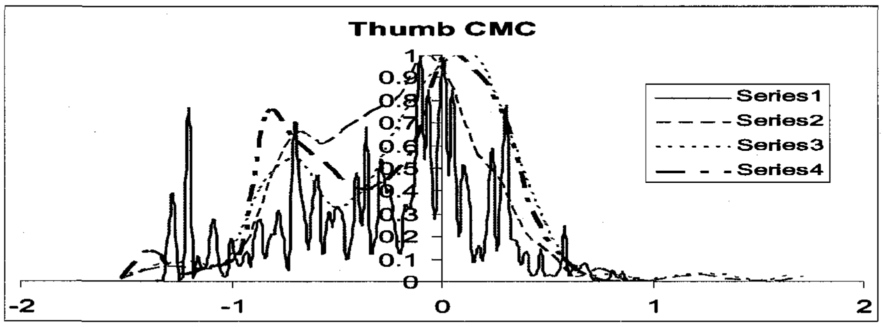
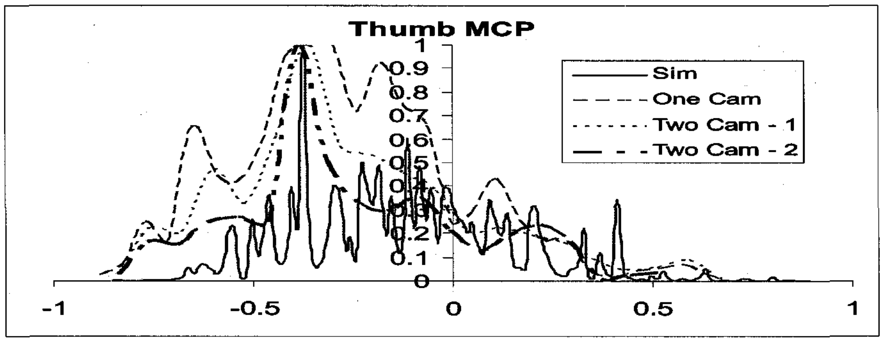
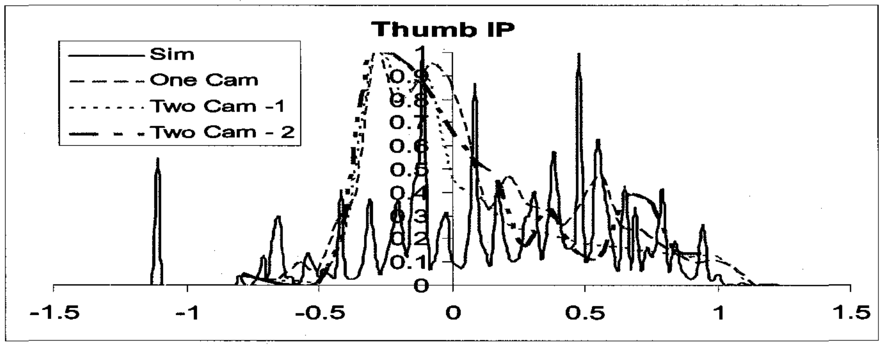
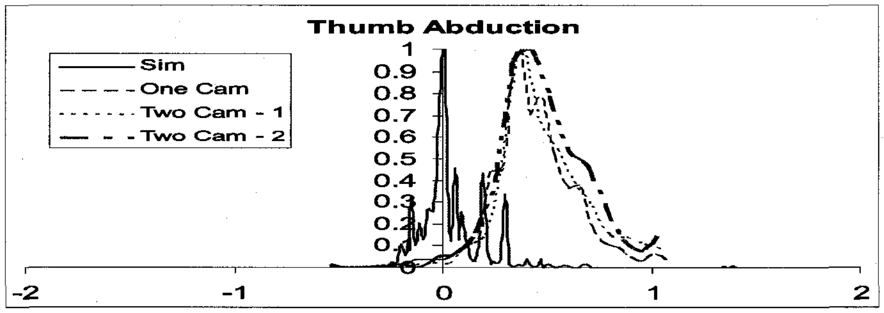


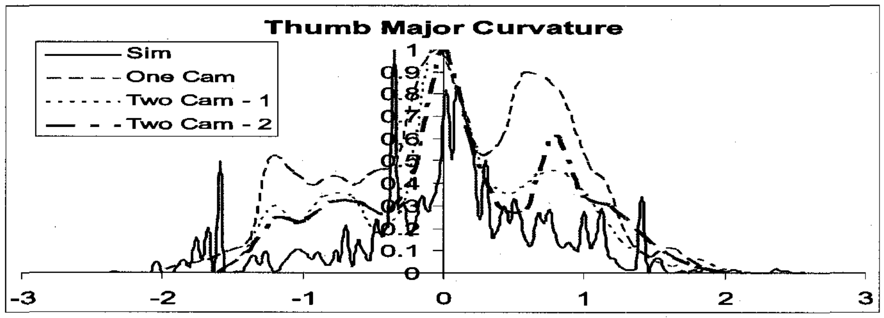
Figure 43: Fingers Error Model




Thumb MCP
Sim
One Cam
Two Cam - 1
Two Cam - 2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
-1
-0.5
0
0.5
1


Thumb IP
Sim
One Cam
Two Cam -1
Two Cam - 2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
-1.5
-1
-0.5
0
0.5
1
1.5


Thumb Abduction
Sim
One Cam
Two Cam - 1
Two Cam - 2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
-2
-1
0
1
2


Thumb Major Curvature
Sim
One Cam
Two Cam - 1
Two Cam - 2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
-3
-2
-1
0
1
2
3

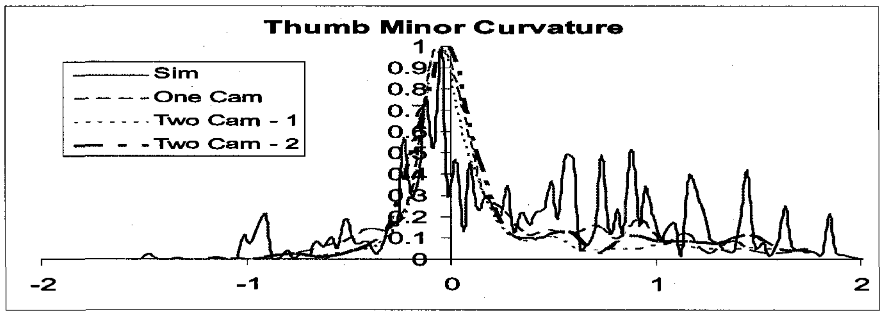


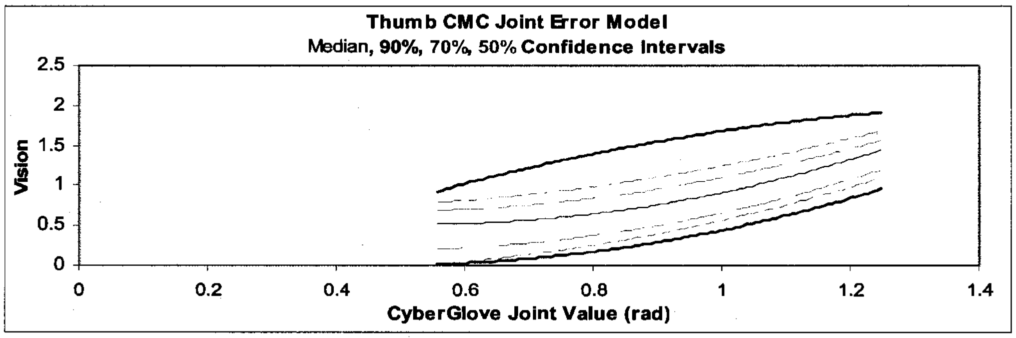
Figure 44: Thumb Error Model

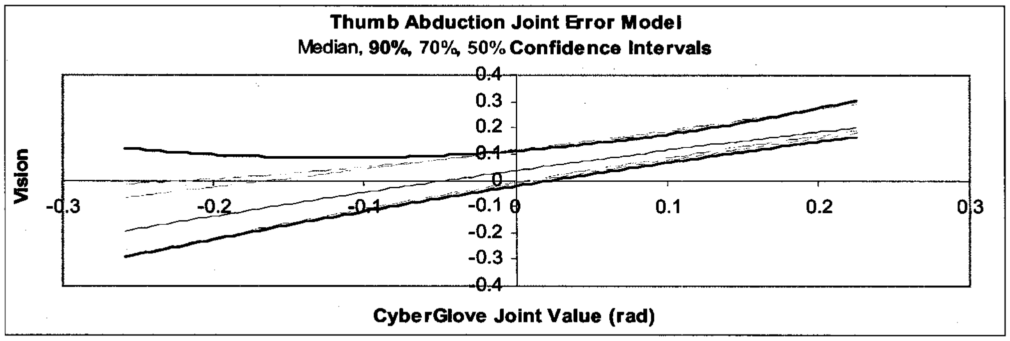
All figures show the collective error from simulation, single camera, and two cameras types one and two. In general the simulation results provide more accurate posture estimation due to accurate detection of the 2D features, and the absence of occlusion, hand model approximation and lens distortion artifacts; although in some cases, e.g. the thumb minor curvature, the vision based tracking provided better results than the simulation; the reason for that is the enhancement of the observation model used in the vision-based tracking, which utilizes the hand color and silhouette. Mean error such as in thumb abduction estimation can be rectified by system calibration as discussed next. In general finger DOF estimation is more accurate than the thumb due the natural constraint of the fingers motion. We recommend using a second marker for the thumb positioned close to the base for better DOF estimation.

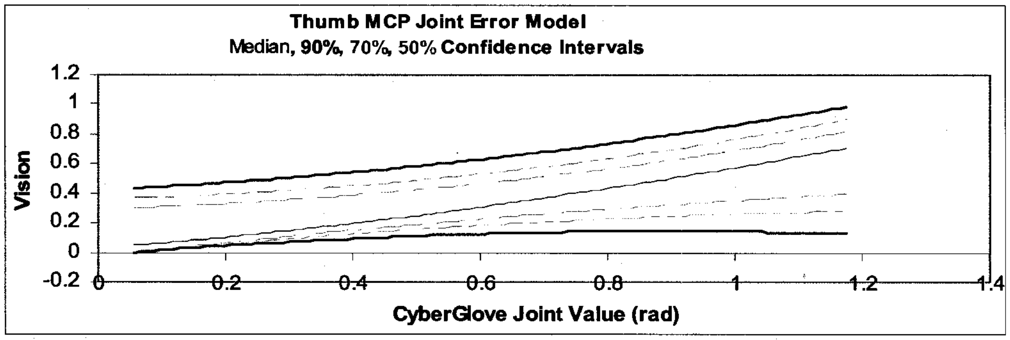
We noticed an obvious zero-error (offset type error) in thumb abduction calculation. We also note that the vision-based posture estimation beat the simulation results in the case of the thumb minor curvature; this is explained by analyzing the thumb inverse kinematics. The thumb has truly 4 degrees-of-freedom with minimum natural constraints. Thus the inverse kinematics of the thumb is obviously under constrained using solely the position of the fingertip. In simulation, another objective constraint is the form of the distance between the target and actual finger tip position is used. In the vision-based tracking the additional objective function is the probability of the posture, given the observed features – as explained in the observation model. Thus, the lack of a proper observation model in simulation resulted in less accurate estimation, even though the detected 2D features are more accurate than in the vision-based tracking.

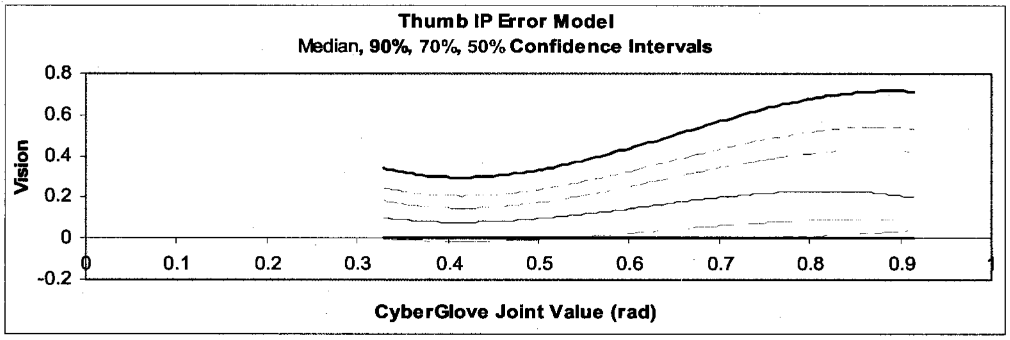
3.5. Vision to Data-Glove Calibration

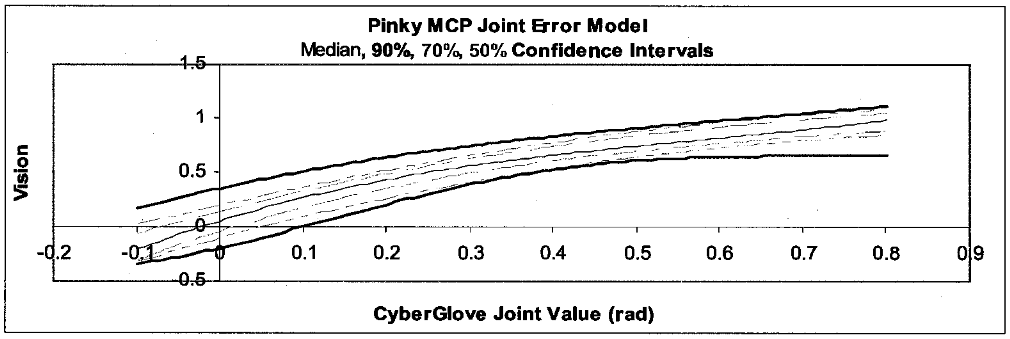
To measure the accuracy of the vision-based posture estimation, we gathered several video clips of hand postures at different hand orientations while wearing the CyberGlove. Although the CyberGlove does not qualify as a true reference because it undergoes a calibration phase of its own, it provides a measure of confidence of the vision system output. The CyberGlove was covered by the color marker glove and then calibrated. To eliminate synchronization errors between the camera and the CyberGlove, each clip contains a single hand posture. The joint angles from both CyberGlove and the model based posture estimation are statistically analyzed. First the scatter plot of the data acquired for each joint is plotted. The histogram generated from every video clip is used to calculate the median and the 90%, 70%, and 50% confidence intervals for each DOF, which are then interpolated to derive the confidence interval values for the whole DOF dynamic range. Figure 45 shows the results we obtained for the thumb and pinky. The median curve can be used as a CyberGlove to vision system calibration, and the confidence intervals can be used as a measure of signal to noise ratio. Both measures are required to be able to use both devices interchangeably for dynamic gesture recognition. The accuracy of the vision system is reduced by the following factors: The loss of depth information due to image projection, the lens distortion, the feature noise due to the unconstrained background and variable lighting conditions, the missing or occlusion of features, the difficulty to extract finger edges when they are overlapping, and the under constrained inverse kinematics in solving 4-DOF using only 2.5 constraints, particularly in the case of the thumb.




Thumb Abduction Joint Error Model
Median, 90%, 70%, 50% Confidence Intervals
Vision
CyberGlove Joint Value (rad)
-0.3
-0.2
-0.1
0
0.1
0.2
0.3
0.4
-0.4


Thumb MCP Joint Error Model
Median, 90%, 70%, 50% Confidence Intervals
Vision
CyberGlove Joint Value (rad)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
1.4


Thumb IP Error Model
Median, 90%, 70%, 50% Confidence Intervals
Vision
CyberGlove Joint Value (rad)
0.8
0.6
0.4
0.2
0
-0.2
0.1
0.9
1


Pinky MCP Joint Error Model
Median, 90%, 70%, 50% Confidence Intervals
Vision
CyberGlove Joint Value (rad)
1.5
1
0.5
0
-0.5
-0.2
-0.1
0.9

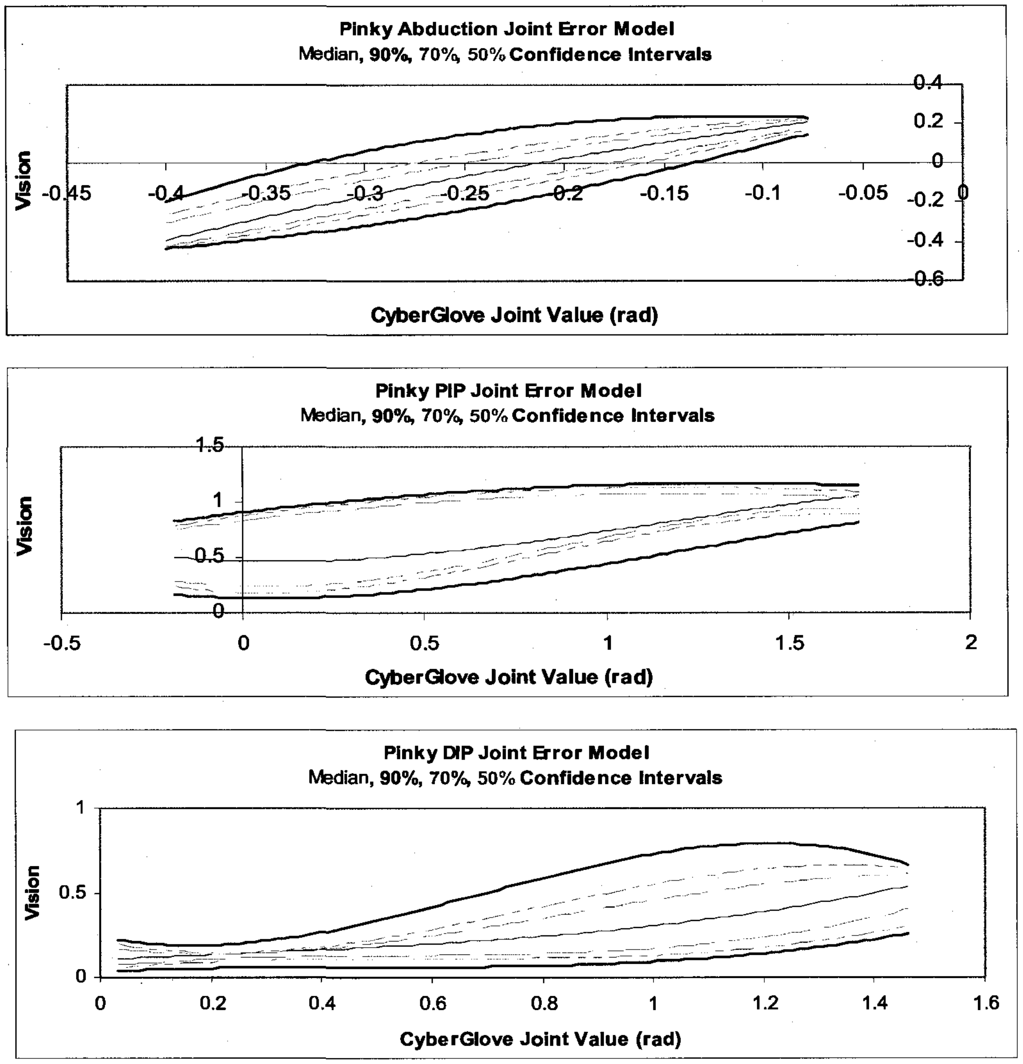


Figure 45: Calibration and Error model Analysis

3.6. Conclusion

We presented a framework for vision based posture estimation using a single camera, which can be extended to multiple cameras. The framework is a two step process, namely posture hypothesis generation and validation. We validated the framework by comparing its outcome to a CyberGlove and shown that it provides reasonable results. The results are systematically enhanced by using multiple cameras at different levels of sensor fusion. The results obtained from the simulation provide an upper bound of the system accuracy, although this can be improved by using the observation model. Using multiple cameras slightly enhanced the accuracy, but mainly improved the continuity of the data by increasing the working area. The system handles intermittent occlusion by relying on posture prediction, but it does not handle occlusion for extended periods of time. This must be handled, in our opinion, by understanding the context in the form of objects (grasped or manipulated) or conversations (hinting to particular gestures). Another challenge is 3D hand tracking of a marker-less hand. In this case edge detection can be used to detect the fingertips and the hand orientation may be estimated from the palm's silhouette. In all these cases, the concept of hypothesis generation and validation using the observation model can be reused.

Chapter 4

4. Hand Gesture Recognition

Dynamic pattern recognition problems can be stated as an optimization problem of a characteristic functional defined over a time sequence of vector attributes. Pattern recognition approaches reportedly successful in recognizing dynamic patterns include Hidden Markov Models (HMM), Time-delay Neural Networks (TDNN), Time-delay Radial Basis Function Networks (TDRBF), and Dynamic Bayesian Networks (DBN).

Hidden Markov Models are based on the assumption that the observed attribute vector sequence is caused by a sequence of states hidden from the observer and conditionally dependent on one another in Markov chain form. The assumption is that the transition between states, which is described by a stochastic matrix governing the trend of state transitions, is stable and time invariant. In our opinion, the stability assumption is too generative, i.e. relying on previous experiences that restrict spontaneous motions, thus not suitable for the emergent nature of improvised gestures. The structure of the HMM is arbitrary and is usually performed experimentally to find the best network topology and number of states.

Time-delay neural networks and time-delay radial basis functions extend the fundamentally sound concepts to handle temporal sequences.

A Bayesian network is a graphical model for representing conditional interdependency between a set of random variables. A Dynamic Bayesian Network is simply a Bayesian network for modeling time series data [39], where the assumption is that dependency trends are in one direction, i.e. states depend on previous states and not vice-versa. The advantage of DBN over the foreseen approaches is its *flexibility*, particularly in terms of relaxing some of the assumptions inherent in building the model, and consequently the *simplicity* of learning and recognition, thus achieving *minimum recognition latency* required for real-time human-computer interaction. In particular, in DBN we relax the assumption of time-invariant state transition; we alternatively consider the state transition matrix as a function of time, as opposed to a fixed state transition in HMM. The transition is not expressed in a closed form, but is expressed as a time sequence, where every element represents a point in time. In other words, the state transitions are evaluated at constant sampling points, emphasizing the dynamic nature of state transition over time. This particular characteristic allows our model to ***distinguish dynamic gestures based on their timing as well as their attributes***, which is an important factor in the gesture interpretation – as mentioned earlier.

In this chapter, we are presenting a novel dynamic gesture recognition model. The model utilizes sequences of posture attributes, assumed acquired at near constant sampling frequency. The postures attributes are generic, but we emphasize the intrinsic hand posture attributes in the form of the finger joints and their derivatives. The model also makes the timing of the dynamic gesture explicit, and thus it can differentiate between similar gestures based solely on timing differences. In the following sections, we first present the derivation of the model, and then present a pragmatic model creation method, followed by the gesture recognition algorithm. We present software design and development details, that we see relevant for discussion. Finally, we present some experimental results.

4.1 Dynamic Bayesian Model for Dynamic Gesture Recognition

Let Θ be a sequence of posture attribute vectors sampled of a time period T, where $\Theta \equiv \{o_1, o_2 \ldots o_T\}$ and o is an observed posture attribute vector. And let our gesture recognition characteristic function be defined as the probability of the gesture, given the observed postures sequence Θ, i.e. $F = P(g_i \mid \Theta)$, where g_i is a dynamic gesture and is a member of a dynamic gesture set of cardinality N, $\Omega = \{g_1, g_2, \ldots, g_N\}$. The dynamic gesture recognition problem can then be described as an optimization of the probability of the gesture, given the observed hand postures over the time period T, over the set of possible gestures Ω.

$$\hat{g} = \arg\max_{g \in \Omega} \{P(g|\Theta)\} \tag{33}$$

Figure 46 gives a pictorial illustration of equation (33).

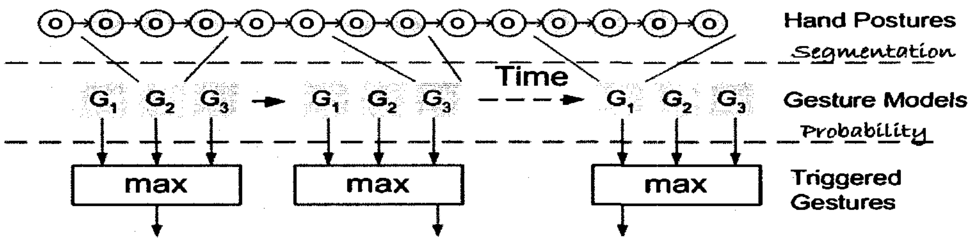


Figure 46: DBN Architecture for Gesture Recognition and Segmentation

The gesture models (shown in the middle layer between the two dashed lines) operate on a sliding window of observed hand postures and the most likely gesture triggers recognition. The probability of a gesture, given the observed posture sequence, can be derived using Bayes rule [40] in terms of the dynamical observation priors, i.e. the probability of a gesture g, given the observed posture sequence Θ P(g|Θ), can be expressed in terms of the probability of the observed sequence Θ, given the dynamic gesture g P(Θ|g), as shown in equation (34).

$$P(g_i|\Theta) = P(\Theta|g_i)P(g_i)/P(\Theta) \tag{34}$$

Analyzing equation (34) closely, we notice that the canonical probability of the posture sequence P(Θ) is a common factor amongst all possible gestures. Similarly, the canonical probability of a gesture $P(g_i)$ is independent from the observation Θ and is likely to be dependent on external forces that we will call the context of the application. The dynamic gesture model thus mainly depends on the dynamical observation priors P(Θ|g), which can be learned from examples of dynamic gesture observation sequences.

Table 9: Terminology for objects

Object	**Name**	**Subscript**	**Subscript Meaning**	**Range**
Gesture set	Ω	g_i	Gesture identification	$i \in [1 \ldots G]$
Posture set	Γ	ϱ	Classified Posture identification	$i \in [1 \ldots S]$
Posture Sequence	Θ	o_i	Observed posture at time sample i	$i \in [1 \ldots T]$
Posture Attribute Vector sequence	$\overline{X}$	$\bar{x}_i \in \Re^N$	Attribute vector of dimension N, observed at time sample i	$i \in [1 \ldots T]$
Stochastic Matrix set	$M_{1..T}$	M_t	Posture transition stochastic matrix at time sample t	$t \in [1 \ldots T]$
Stochastic Matrix	M_t	$m_{ij}(t)$	The probability of transition from posture j to posture i at time sample t	$i,j\varepsilon \in [1 \ldots M]$, $t \in [1 \ldots T-1]$
Initial state Probability	V_1	V_{0i}	The probability of posture i at the initial sample 0	$i \in [1 \ldots M]$
Terminal state Probability	V_T	v_{Ti}	The probability of posture i at the last sample in the posture	$i \in [1 \ldots M]$

We model the posture attribute vector sequence using a Bayesian network, which is a generic model for time series data - Figure 47.

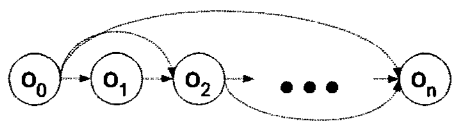


Figure 47: DBN model of Posture Sequences

The conditional dependence of the observed sequence Θ over the dynamic gesture P(Θ | g) is learned by the gesture models from supervised training sequences associated with the dynamic gesture. The model's characteristic function can now be derived using the DBN model assumption by equation (35)

$$P(\Theta|g) = P(o_1, o_2, \ldots, o_T|g) = P(o_1|g)P(o_2|o_1, g)\ldots P(o_T|o_{1\ldots T-1}, g) \tag{35}$$

From Nyquist–Shannon sampling theorem, we can derive a proper sampling frequency to preserve the dynamic information of the gestures based on the bandwidth of the dynamic gestures in question. Assuming that we are using the proper sampling frequency for our band limited dynamic gestures; we can assume a first order Markov chain causality model for our observed posture sequence, where every posture is solely dependent on the previous sample and not on any former posture history - Figure 48.

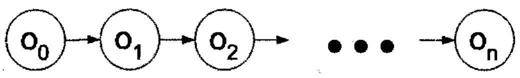


Figure 48: Markov Network model of Posture Sequences

The conditional dependence of the observed posture sequence (Θ) over the dynamic gesture (g) P(Θ | g) using the first order Markov chain assumption is given by equation (36).

$$P(\Theta|g) = P(o_1|g)P(o_2|o_1, g)\ldots P(o_T|o_{T-1}, g) \tag{36}$$

Without loss of generality, higher order Markov chain causality models can be used, in the case when the sampling frequency falls short from the proper sampling frequency, e.g. the tracking system is slow, but we did not anticipate the need, particularly in the CyberGlove case where the system could be running at 50 Hz sampling frequency. Higher order Markov models can provide a smother posture transition, particularly in a highly dynamic system. We overcome the need for higher order Markovian dynaics by using high sampling frequency, where the dynsmics of the system is surely captured collectively by the transition matrices.

The right hand side of equation (36) defines the dynamic gesture model. The probability of initial posture, given the gesture $P(o_1 | g)$ is represented by a vector functional V_0, and the probability of a posture o_i at sample *i*, given posture o_{i-1} at the previous sample and the gesture $P(o_i | o_{i-1}, g)$ is represented by a stochastic posture transition matrix functional M_i whose dimension is equal to the cardinality of the set of all possible postures Γ. A clustering technique is used to transform the domain of possible hand postures from the real R^N domain, where N is the number of posture attributes, to a discrete set of postures of cardinality Γ. Thus, the gesture model derived from equation (36) is given by a probability vector, and a sequence of stochastic matrices representing the attribute variation at every sampling point. A terminal probability vector V_T, *derived from the last posture* in the training sample, is used to detect posture termination – equation (37).

$$G \equiv \{V_0, M_1, M_2, \dots, M_{T-1}, V_T\} \tag{37}$$

4.2. Gesture Model Creation

The model creation, or the training process, is the process of model parameter estimation using supervised training data. The training data are comprised of a sequence of hand postures segmented sampled at a constant frequency and segmented at the beginning and the end of the dynamic gesture. The sampling frequency is calculated using the Nyquist–Shannon sampling theorem from the estimated target bandwidth of the dynamic hand gestures. The training process is shown in Figure 49.

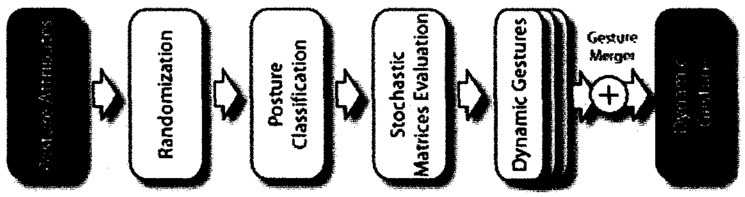


Figure 49: DBN Model Training

Similar to the concept of radial basis functions, we generate an *ensemble X(t)* of gestures samples from every training sample sequence *x(t)*. The ensemble *X* is generated by adding two random vectors α and β corresponding to variations in gesture timing and acuteness respectively - equation (38).

$$X(\alpha, \beta, t) = x(\alpha t) + \beta \quad (38)$$

Figure 50 shows the effect of attribute randomization of a scalar attribute. The combination of all mutually independent attributes comprises the randomized attribute random vector.

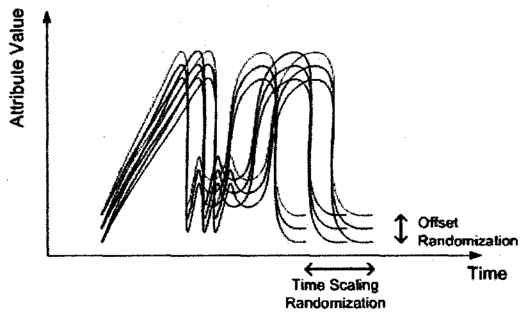


Figure 50: Example Attribute Randomization

The random variables α and β are normalized to the random vectors γ and δ, with elements in the range [-1, 1], as given in equation (39).

$$\begin{aligned} \alpha &= 1 + A\gamma, & \alpha &\in [1-A, 1+A] \\ \beta &= B\delta, & \beta &\in [-B, B] \end{aligned} \quad (39)$$

Let Φ_A and Φ_B be the probability density function of the normalized random vectors γ and δ respectively. Φ_A and Φ_B can be represented in the form: $\Phi_A = \{\varphi_{ai}\}$ and $\Phi_B = \{\varphi_{bi}\}$, where φ_{ai} and φ_{bi} are mutually independent scalar random variables in the range [-1, 1]. The corresponding probability distribution function $F_\varphi(x) = \int_{-\infty}^{x} \varphi(y)dy$.

Table 10: Terminology for Functions

Function	Symbol	Mapping	Description
Posture Classifier	$C(\bar{x})$	$\Re^N \to \Gamma$	A mapping of the posture attribute vector $\bar{x}$ to the hand posture set
Posture Labeling	$L(\varrho)$	$\Gamma \to [0,S-1]$	A mapping from the posture set to an integer in the range [0, S-1]
Probability Density	$p(x)$	$[-1,1] \Rightarrow \Re$	Probability density of the posture attributes randomizing variables
Probability Distribution	F(x)	$\Re \Rightarrow [0,1]$	Probability distribution of the posture attributes randomizing variables
Posture Domain	Λ(i, t)	$[0,S-1]\times[0,T] \to [a,b]\in[-1,1]$	A mapping from a posture label i at sample t to an interval in [-1, 1]

Let C be the posture clustering function mapping the hand postures from the Cartesian R^N domain to the discrete space of clustered postures Γ' - equation (40), and let L be a labeling function mapping the clustered hand posture ϱ from the set Γ' to a subset of the Nonnegative integers, in the range [0, S], where S is the cardinality of Γ' - equation (41).

$$C(X) = \varrho \in \Gamma', \qquad X \in R^N \tag{40}$$
$$L(\varrho) = n \in \{0, 1, 2, 3, \ldots S\}, \qquad \varrho \in \Gamma' \tag{41}$$

For the sake of clarity, an example of the radial basis functions, which can be used as a probability density function φ, is in the form of equation (42), where *n* is a parameter used to modify the characteristics of the probability density function.

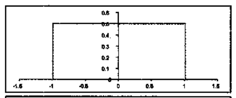
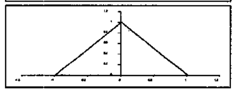
$$\varphi_n(x) = \frac{n+1}{2}\left(1-|x|^{1/n}\right), \quad x \in [-1,1] \tag{42}$$

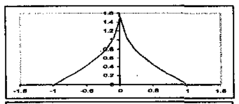
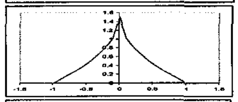
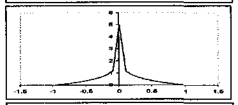
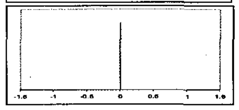
The corresponding probability distribution function F(x) is given in equation (43).

$$F_\varphi(x) = \int_{-\infty}^{x} \varphi(y)dy = \begin{cases} 0 & ,x \le -1 \\ \begin{cases} \frac{1}{2}(1+x) & ,n=0, x \in (-1,1) \\ \begin{cases} \frac{1}{2} - \frac{n+1}{2}\left(|x| - \frac{n}{n+1}|x|^{\frac{n+1}{n}}\right) & ,0<n<\infty, x \in (-1,0] \\ \frac{1}{2} + \frac{n+1}{2}\left(|x| - \frac{n}{n+1}|x|^{\frac{n+1}{n}}\right) & ,0<n<\infty, x \in [0,1) \end{cases} \\ \begin{cases} 0 & ,n=\infty, x \in (-1,0) \\ 1 & ,n=\infty, x \in [0,1) \end{cases} \end{cases} \\ 1 & ,x \ge 1 \end{cases} \tag{43}$$

Table 11 shows the variation of the example probability density function (pdf) Φ as the parameter n varies from 0 to ∞. It can be seen that the density function can be represented as a uniform-density where the emphasis is distributed uniformly in the attribute range to an impulse, at the limit n → ∞ the pdf becomes an impulse, which means that only the training sample is used without any randomization.

Table 11: Parameterized probability density functions

n	Probability Density Function	Graph
	$\varphi(x) = \frac{n+1}{2}\left(1-|x|^{1/n}\right)$	
0	$\varphi(x) = \frac{1}{2}$	
1	$\varphi(x) = \left(1-|x|\right)$	

2	$\varphi(x)=\frac{3}{2}\left(1-\lvert x\rvert^{1/2}\right)$	
3	$\varphi(x)=2\left(1-\lvert x\rvert^{1/3}\right)$	
9	$\varphi(x)=5\left(1-\lvert x\rvert^{1/9}\right)$	
∞	$\varphi(x)=\begin{cases}\infty & x=0\\ 0 & x\neq 0\end{cases}$	

4.2.1. Posture Transition Stochastic Matrices Calculation

The posture transition matrix M_t at time sample *t* is a square matrix of size S x S, where S is the cardinality of the clustered posture set Γ. Every element $m_{ij}(t)$ of M_t represents the probability of a state transition from the posture labelled j to the posture labelled i, given the generalized posture sequence family X(α, β, t) – equation (44). The posture transition matrix M_t is calculated by integrating the *pdf* of the random variables α and β over the ranges where the corresponding posture remains constant. It can be defined more formally by equation (45).

$$m_{ij}(t) = \frac{P(L(C(X(\alpha,\beta,t)))=i, L(C(X(\alpha,\beta,t-1)))=j)}{P(L(C(X(\alpha,\beta,t-1)))=j)} \quad (44)$$

$$m_{ij}(t) = \frac{\iint\limits_{\{L(C(X(\alpha,\beta,t)))=i \cap L(C(X(\alpha,\beta,t-1)))=j\}} \Phi_A(\gamma)\Phi_B(\delta)d\gamma d\delta}{\iint\limits_{\{L(C(X(\alpha,\beta,t-1)))=j\}} \Phi_A(\gamma)\Phi_B(\delta)d\gamma d\delta} \quad (45)$$

We will define Λ as the interval in the posture attribute vector space R^N corresponding to a posture label i at time sample t.

$$\Lambda(i, t) = (\gamma, \delta) \mid L(C(X(\alpha, \beta, t))) = i \quad (46)$$

Substituting from (46) into (45) we get

$$m_{ij} = \frac{\iint\limits_{\Lambda(i,t)\cap\Lambda(j,t-1)} \Phi_A(\gamma)\Phi_B(\delta)d\gamma d\delta}{\iint\limits_{\Lambda(j,t-1)} \Phi_A(\gamma)\Phi_B(\delta)d\gamma d\delta} \quad (47)$$

Λ(i, t) can be expressed in terms of the union of disjoint intervals in the (γ, δ) space.

$$\Lambda(i,t) = \bigcup_j \left(\left[\breve{\gamma}_j, \hat{\gamma}_j \right], \left[\breve{\delta}_j, \hat{\delta}_j \right] \right) \qquad (48)$$

Equation (47) can now be written as a product of the difference of the probability distribution functions of posture attributes.

4.2.2. Initial Probability Vector Calculation

The initial probability vector V_0 represents the probability that the gesture starts at a particular posture, given the training sample family X(α, β, t). Every element v_i of V_0 represent the probability that the gesture starts at the posture labelled i. It can be defined by an integral of the probability density function over the domain of the random variables over which the random vector X(α, β, 0) correspond to a particular posture, as shown in equation (49).

$$V_0 = \{ v_{0i} \}, \quad v_{0i} = \iint_{(\gamma,\delta)|L(C(X(\alpha,\beta,0)))=i} \Phi_A(\gamma)\Phi_B(\delta)d\gamma d\delta, \quad i \in \{0,1,2,3,\ldots S\} \qquad (49)$$

Substituting with time t = 0 in equation (38), we get

$$X(\alpha, \beta, 0) = X(0) + \beta \qquad (50)$$

Thus, the initial probability vector is independent of the timing scale factor at time t = 0. Therefore, equation (49) can be simplified as in (51).

$$V_0 = \{ v_{0i} \}, \quad v_{0i} = \int_{\delta|L(C(X(0)+\beta))=i} \Phi_B(\delta)d\delta, \quad i \in \{0,1,2,3,\ldots S\} \qquad (51)$$

From the assumption of scalar attributes mutual independence, we can perform the integration of equation (51) on each attribute random variable independently and the product of all scalar integrals taken as the resultant value.

$$V_0 = \{ v_{0i} \}, \quad v_{0i} = \prod_{j=0}^{N-1} \int_{\delta|L(C(X(0)+\beta))=i} \varphi_{b_j}(\delta_j)d\delta_j, \quad i \in \{0,1,2,3,\ldots S\} \qquad (52)$$

Finally, the scalar integral of equation (52) can be expressed in the form of a difference between the probability distribution functions $F_{\varphi j}$ evaluated at the interval ends.

$$V_0 = \{ v_{oi} \}, \quad v_{0i} = \prod_{j=0}^{N-1} (F_{\varphi_j}(\hat{\delta}_j) - F(\breve{\delta}_j)), \quad i \in \{0,1,2,3,\ldots S\} \qquad (53)$$

The domain of the integration is constrained by (L(C(X(0)+ β)) = i), which can be expressed by the union of disjoint intervals in R^N defined by $\{[\breve{\delta}_j, \hat{\delta}_j]\}$.

4.2.3. Terminal Probability Vector Calculation

Dynamic gestures terminate with specific postures which may persist over a small period of time. These postures are significant to observe and can be used for gesture segmentation. The terminal posture probability V_T represents the probability that the gesture ends at a particular posture, given the training sample family X(α, β, t). The ending time of the sample X(α, β, t) is inversely proportional to the scaling factor α, and is given by (T/α), where T is the duration of the training sample X(t). Every element v_i of V_T represent the probability that the gesture starts at the posture labelled i. It can be defined by an integral of the probability density function over the domain of the random variables over which the random vector X(α, β, T/α) correspond to a particular posture. Substituting with time t = T/α in equation (38), we get

$$X(\alpha, \beta, T/\alpha) = X(T) + \beta \qquad (54)$$

Thus, the terminal probability vector is independent of the timing scale factor, similar to the initial probability vector. Therefore, the terminal probability vector can be estimated as in initial probability vector case – cf. (51).

$$V_T = \{ v_{Ti} \}, \quad v_{Ti} = \int\limits_{\delta | L(C(X(T)+\beta))=i} \Phi_B(\delta) d\delta, \qquad i \in \{ 0, 1, 2, 3, \dots S\} \qquad (55)$$

Detection of the terminal posture at any time past a minimum period, defines the end of the gesture, even though the gesture model may contain more transition matrices, given that the gesture provides a non-zero probability given all the previous postures. For example, suppose a dynamic gesture is performed faster than average. It may terminate, while the gesture model contains posture transition matrices modeling slower versions of the same gesture. The detection of the terminal posture automatically renders the model's transition matrices beyond it as invalid. In the gesture recognition section however, the gesture probability is calculated, given a fixed sample size. For that reason the gesture recognition is divided into two sections, one prior to the detection of the terminal posture and another past the detection of the terminal posture.

4.2.4. Compound Gesture Models

It is desirable in gesture recognition and interpretation to be able to regroup the gesture models by combining and splitting gestures. Generalizing the training gesture sample as discusses above is not sufficient to accommodate for all the variances of the gesture. Multiple training samples are acquired and combined to generate a more general model. Moreover, dealing with the gesture set as a ring, where the gestures can be grouped and subtracted or split, may be advantageous in both the recognition phase where a more general gesture model is used to infer a concept, and then the more specific models can add specific meaning to the gesture.

The dynamic gesture model created using a single sequence of hand postures is extended to include multiple samples by *merging* the gesture models into a more general or compound model. The merge operator can be used to build a hierarchy of similar gesture models. Merging the probability vectors is performed by a weighting average between the corresponding models. The gesture model of equation (37) is thus extended to include a scalar weight ω, which reflects the commonality of the gesture model and the number of gesture samples used to generate the gesture model. It is initialized to one and grows as the gesture model compounds more gestures.

$$G \equiv \{\omega, V_0, M_1, M_2, \ldots, M_{T-1}, V_T\} \tag{56}$$

The merge operation $\oplus$ is similar to an ADD operation, as shown in equation (57).

$$\begin{aligned} \omega_1 \oplus \omega_2 &= \omega_1 + \omega_2 \\ V_{0,1} \oplus V_{0,2} &= (\omega_1 V_{0,1} + \omega_2 V_{0,2}) / (\omega_1 + \omega_2) \\ M_{i,1} \oplus M_{i,2} &= (\omega_1 M_{i,1} + \omega_2 M_{i,2}) / (\omega_1 + \omega_2) \\ V_{T,1} \oplus V_{T,2} &= (\omega_1 V_{T,1} + \omega_2 V_{T,2}) / (\omega_1 + \omega_2) \end{aligned} \tag{57}$$

Special attention must be paid to the stochastic matrices M_i. According to the model creation procedure discussed earlier, the posture transition matrices are not truly stochastic matrices, because according to equation (47) if a posture j not encountered at time t-1, the elements $m_{ij}(t)$ are not determined $\forall$ i (division of $0 \div 0$) and are set by default to zero. Merging such a row with a non-zero row will violate the definition of stochastic matrices, which indicates that all rows must add to one (definite probability), and will render inversion of the merge operation (i.e. split) infeasible. To resolve this issue, we extend the posture set to include a hypothetical *null posture*, indicating an invalid posture. In this case the row corresponding to the transition from an unreachable posture will be represented by a transition from the *null posture*, and will thus have a value of 0 in all elements except column corresponding to the *null posture* will have a value of 1 – conforming to the definition of the stochastic matrices.

The inverse of the merge operation, the split operation ($\ulcorner$) is defined by equation (58).

$$\begin{gathered}\omega_1 \ulcorner \omega_2 = \omega_1 - \omega_2,\ \omega_1 \neq \omega_2 \\ V_{0,1} \ulcorner V_{0,2} = (\omega_1 V_{0,1} - \omega_2 V_{0,2}) / (\omega_1 - \omega_2) \\ M_{i,1} \ulcorner M_{i,2} = (\omega_1 M_{i,1} - \omega_2 M_{i,2}) / (\omega_1 - \omega_2) \\ V_{T,1} \ulcorner V_{T,2} = (\omega_1 V_{T,1} - \omega_2 V_{T,2}) / (\omega_1 - \omega_2)\end{gathered} \tag{58}$$

4.3. Gesture Recognition

One of the advantages of our model is the speed by which we can recognize dynamic gestures, that is why it is suitable for real-time and interactive applications. The reason is that the data required to evaluate the probability of a gesture, given the sequence of observed postures, are directly available in the model. The label of the classified posture derived from the observed posture attribute vector, i.e. L(C(X(t))), is used as an index in the model's probability vectors and stochastic matrices. The product of the elements corresponding to these indices, comprise the probability of the gesture given the sequence of postures X(t).

According to equation (36), the probability measure is given by the product of an initial probability and a set of conditional probabilities, which is rewritten in equation (59) in terms of elements of the gesture model parameters.

$$P(\Theta|g) = P^{(0)}_{o_1} P^{(1)}_{o_1 o_2} P^{(2)}_{o_2 o_3} \dots P^{(T-1)}_{o_{T-1} o_T} \tag{59}$$

As mentioned briefly in the gesture model training, equation (59) is valid only as long the gesture has not passed its terminal state (i.e. $t \leq \tau$, where τ is the last sample prior to leaving the terminal posture). We would like to evaluate the probability of posture, given the gesture over a fixed period of time for all the gestures, thus normalizing the probability w.r.t. time duration. The probability of a gesture given the observed postures is divided into two parts, one is prior the terminal posture ($t \leq \tau$) and another is past the terminal posture ($\tau < t < t_{max}$), where $t_{max} \geq \tau$ is the size of the observed posture time window. The first part is within the scope of the gesture model, and the probability coefficients are derived from the model parameters, as in equation (59). The second part is outside the scope of the model, and is evaluated using a '*null-model*' stochastic matrix 'Q', which is independent of all gestures and sample time. Thus the probability evaluation is normalized with respect to the observation period, as shown in equation (60).

$$P(\Theta|g) = P_{o_1}^{(0)} \prod_{i=1}^{i\le\tau} P_{o_i o_{i+1}}^{(i)} \prod_{i>\tau}^{i<t_{\max}} Q_{o_i o_{i+1}} \tag{60}$$

The segmentation time τ is defined as the time sample $t > t_{\min}$, where the probability of terminal state of the corresponding posture is greater than zero and the probability of the terminal state of the next posture is zero; this is expressed logically as the event Terminal Posture ($TP = t > t_{\min}$ && $V_{T,L(C(X(t)))} > 0$ && $V_{T,L(C(X(t+1)))} = 0$).

The pseudo code for the evaluation of the probability of the gesture, given the observed postures, is shown in Table 12.

Table 12: probability of gesture pseudo code

```
1.set TP = false
2.set t = 1
3.get label of first sample $O_1 = L(C(X(0)))$
4.set probability p = $V_{0,o1}$
5.while  sample t < $t_{max}$
   a. t = t + 1
   b. classify sample $O_t$ = L(C(X(t)))
   c. if TP (Terminal Posture)
      go to 6
   d. p = p * $M(t)_{o_t o_{t-1}}$
6.while  sample i < $t_{max}$
   a. p = p * $Q_{o_t o_{t-1}}$
   b. t = t + 1
   c. classify sample $O_t$ = L(C(X(t)))
7.return  p
```

Steps one to four are for initialization; steps five and six represent the first and second parts of equation (60), respectively.

4.4. Fuzzy Dynamic Bayesian Model

The dynamic gesture recognition framework described so far works well with noiseless data acquired from accurate sensors such as the CyberGlove. But in the presence of noise acquired from unconstrained environment using vision-based hand tracking, the framework is modified in two ways to handle such noise. First, a larger tolerance for posture attribute offsets in the gesture model creation is used. Second, a fuzzy posture set in the form of multiple posture candidates and their associated observed probability, is utilized in the recognition phase. Figure 51 shows the fuzzy dynamic Bayesian model architecture.

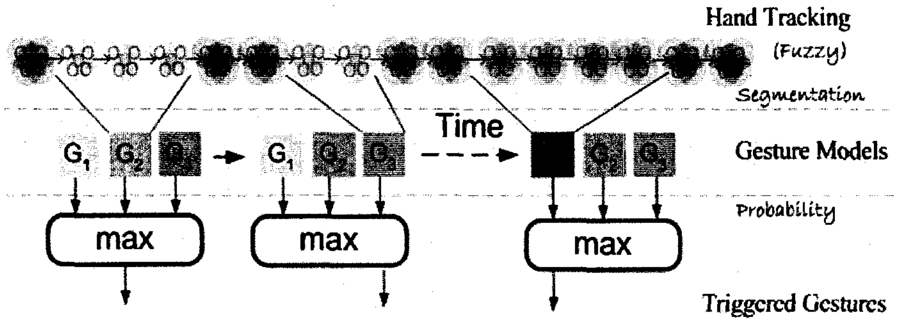


Figure 51: Fuzzy DBN Model architecture

The dynamic gesture model $G \equiv \{\omega, V_0, M_1, M_2, \ldots, M_{T-1}, V_T\}$, of equation (56), remains the same. The gesture recognition algorithm of equation (36) and (60) and Table 12 is modified to process a fuzzy set of postures at every sample point. Borrowing the terminology of genetic programming [64], this problem lends itself to the concept of genetic crossover. Mixing and matching the detected hand postures prioritized using the probability of the observation, provides a candidate population of adequate postures. The posture population grows exponentially over the observed posture sequence, but the assumption here is that there is only one correct solution and the fitness of any other noisy members of the population is negligible. In other words, the probability that a noisy sequence of hand postures triggers gesture recognition is close to zero. Thus, instead of processing the exponentially increasing posture sequence population individually, a weighted sum of the posture population at every sample point is used, where the probability of observation is used as the weight. Moreover, the potential postures at the latest sample point, i.e. the posture with high fitness to the gesture recognition problem, are maintained in the loop by feeding them back in the sample population (Figure 52); insuring their persistence under intermittent occlusion or feature extraction failure.

Equation (36) is modified for a fuzzy posture set as in Figure 51, which optimizes the probability all possible permutations of the fuzzy posture set.

$$P(\Theta|g) = \arg\max_{i_1 \ldots i_T}\{P(o_{i1}|g)P(o_{i2}|o_{i1},g)\ldots P(o_{iT}|o_{i(T-1)},g)\} \tag{61}$$

The processing time for equation (61) is in the order O(T^N), where N is the number of postures in the fuzzy posture ensemble.

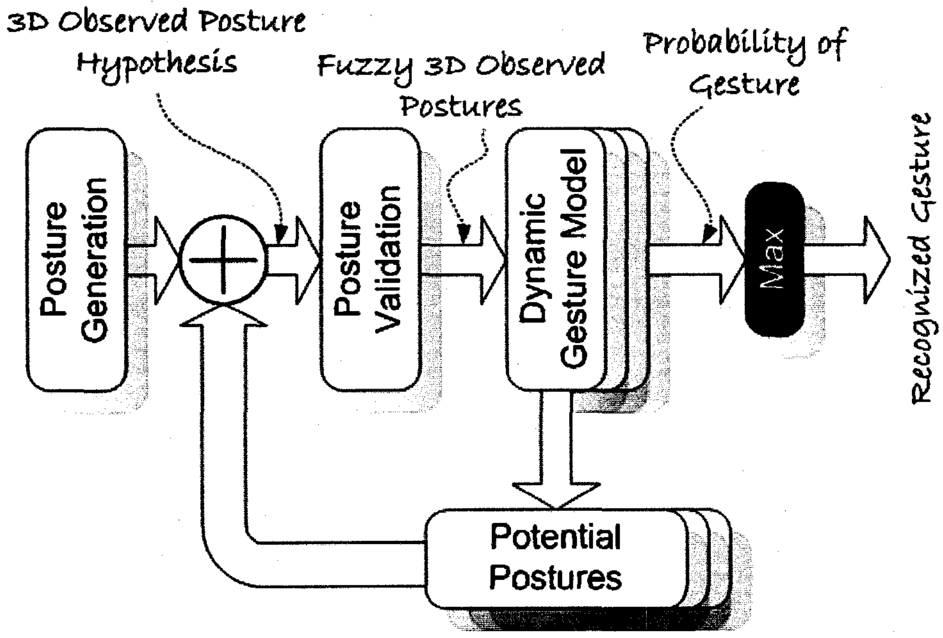


Figure 52: Fuzzy Dynamic Bayesian Model for Gesture Recognition

To achieve polynomial time processing we modify equation (61) to utilize the fuzzy posture set as one entity and we take the waited sum of the probability of all elements in the fuzzy posture set, assuming that the noisy postures will not score any gesture over the gesture period. The probability of gesture, given the fuzzy posture is thus given by equation (62), which has a processing complexity of O(T^2).

$$P(\Theta|g) = \sum_{i_1} P(o_{i1}|g)\sum_{i_2,j_1} P(o_{i2}|o_{j1},g)\ldots \sum_{i_T,j_{T-1}} P(o_{iT}|o_{j(T-1)},g) \tag{62}$$

4.5. Implementation Discussion

Since response time is a major issue in the success of recognition-based human computer interaction, we explain the techniques we used to improve the response time of the dynamic gesture recognition algorithm. The algorithm, explained above, utilizes a sliding window of hand postures. At each sampling time, the window slides one element allowing a new sample to enter at the end of the window and removing a sample from the beginning. Typical implementation of sliding window, or buffer type, applications is set up using the maximum window/buffer requirement allowing the algorithm to process the longest possible gesture. Aligning the gesture model with the start of the sliding window, will result in huge latency in most cases due to buffering unnecessary postures at the end, as shown in Figure 53.

One way to eliminate the latency is to derive the dynamic gesture model traversing backwards in time, where the sliding window is always utilized up to the latest sample. Since the gesture model is structured symmetrically with respect to time, i.e. it is comprised of an initial and terminal probability vectors bounding a sequence of stochastic matrices, it can be trained by traversing the training samples in backward direction. Figure 53 shows the advantage of reverse time traversal mode over the forward time traversal one., where it is evident that in inverse time traversal case the latency due to excess buffer size has been eliminated by shifting it to the past side of the time line

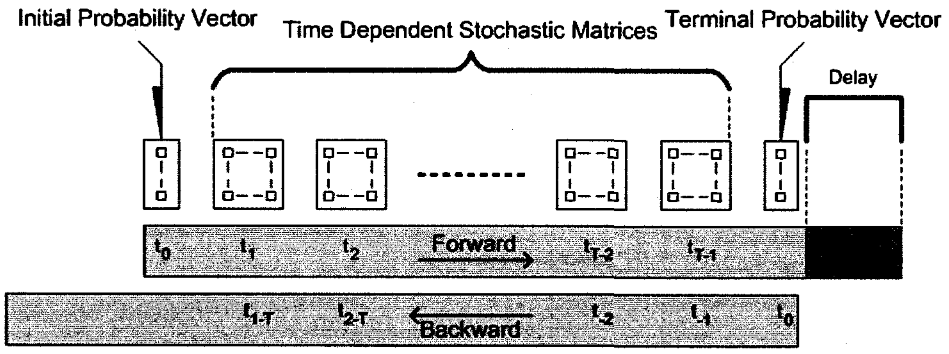


Figure 53: Forward & backward time traversal dynamic gesture models

Another way we to reduce latency, due to excess buffer size, is to keep track of the effective sample count for each gesture in the gesture set. The effective sample count is a state variable associated with every gesture and is initialized to zero. At every sample, the state variable is incremented if the probability of the gesture is greater than zero, or reset to the greatest value providing non-zero probability, or to zero otherwise. The state variable helps align the sliding window with the current time sample, avoiding the latency due to aligning the sliding window fully with current sample.

Figure 54 shows the class diagram of the dynamic gesture recognition library we developed to test the performance of the gesture model. The main class in the diagram is the Dynamic

Gesture class, which is comprised of the initial and terminal probability vectors and the state transition matrices. It also contains the posture classifier. It utilizes other gesture models by merging to them, and it utilized the gesture sample by providing the probability of the gesture, given the posture sequence. The Gesture Sample class has different variants such as the classified-, sliding window-, or the classified sliding window – gesture sample.

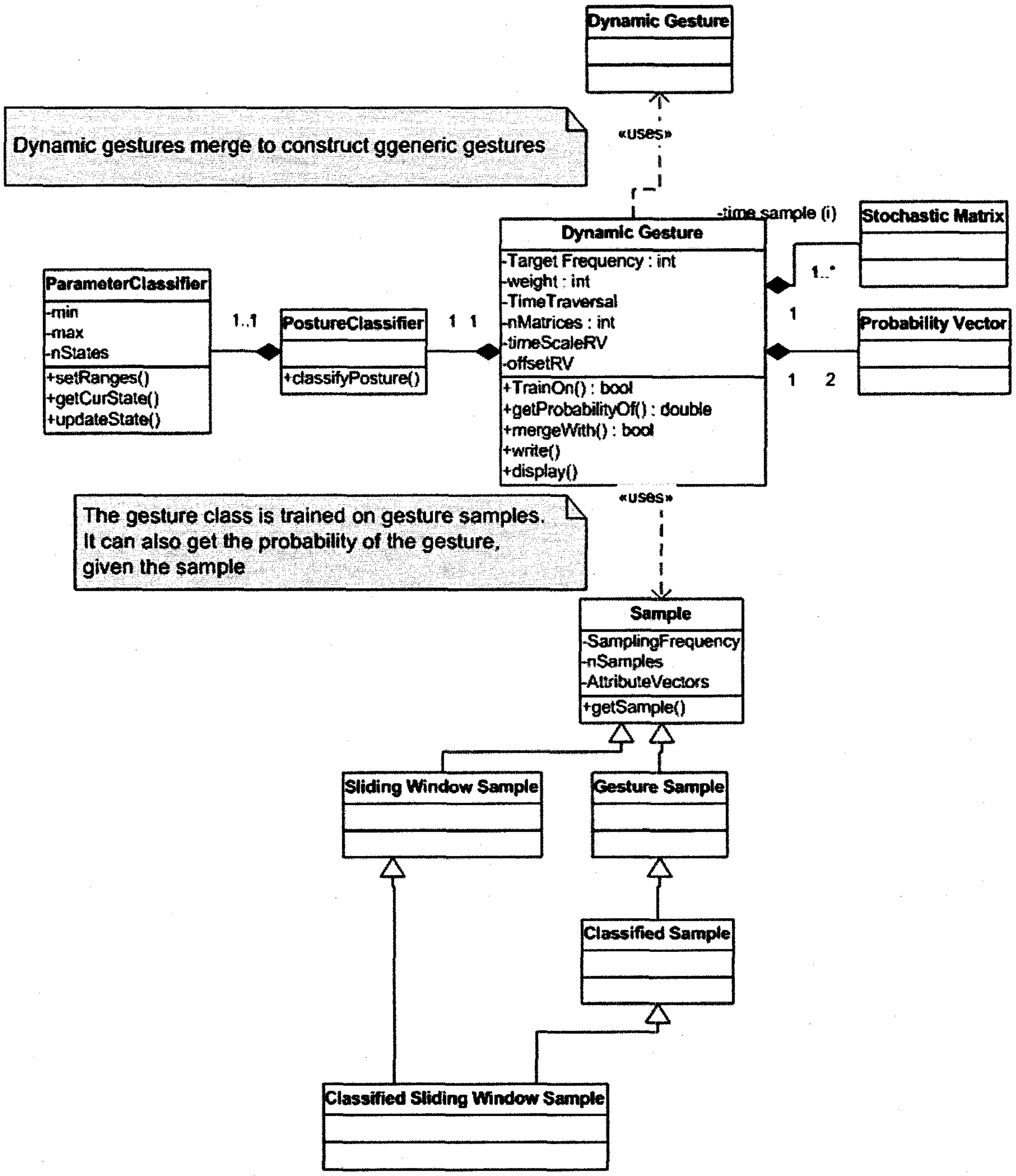


Figure 54: Dynamic Gesture Model Class Diagram

4.5.1. Hand Tracking and Gesture Recognition System Integration

We described the dynamic gesture recognition framework and the vision based hand tracking framework. The data flow in each framework goes through an endless loop of processing. The hand tracking framework continuously extracts the hand features from the acquired images and updates the hand posture estimation accordingly. The gesture recognition framework continuously acquires the hand posture and provides the state of the acquired gestures. One problem of integrating the two frameworks is that they may be required to run at different cycle rates (frequency). For example, the vision-based hand tracking framework can run as fast as the camera frame rate 30 Hz but may be slowed down by processing load, so practically it works in a variant frequency in the range [5, 30] Hz. Meanwhile, the gesture recognition is required to work in a near constant frequency of 50 Hz, in case of the CyberGlove, and 10 Hz in case of the vision-based tracking. Figure 55 shows how we resolved the data rate discrepancy.

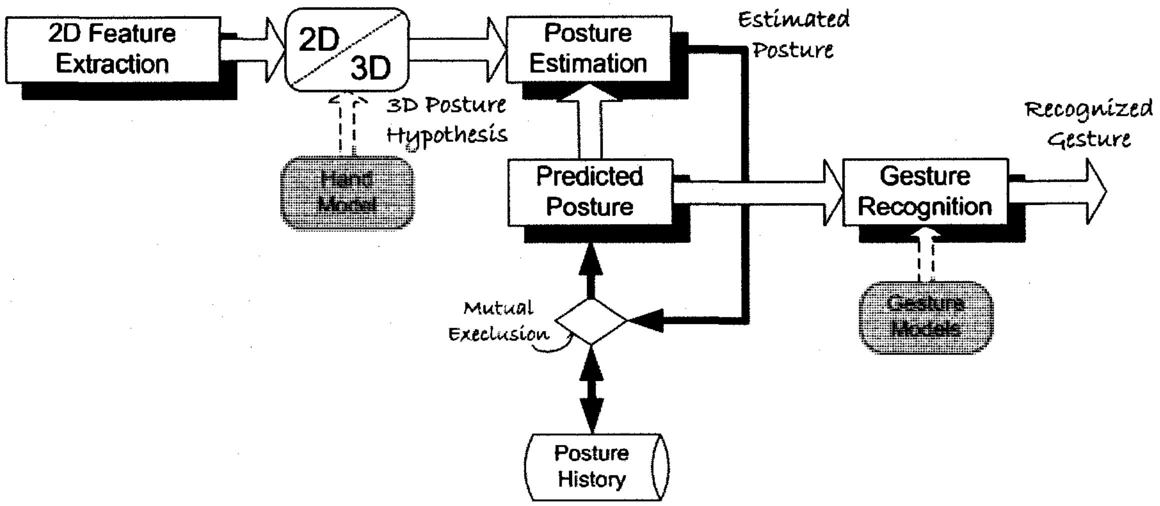


Figure 55: Hand Tracking and Gesture Recognition Integration

The solution to this problem is to make the posture prediction interface a re-entrant mutual exclusion critical section in the hand tracking framework. The interface is acquired periodically by the gesture recognition module to get an updated estimation of the hand posture. In other words, the gesture recognition process is running asynchronously with the hand tracking process, and it utilizes the predicted posture interface to asynchronously acquire the hand posture at high data rate, independently from the hand tracking process, which updates the estimated postured at a slower rate.

4.6. Posture Attributes and Posture Classification

The human hand can be modeled using an articulated structure of 19 links and 20 degrees-of-freedom, plus 6 DOF for the hand position and orientation. Thus, the hand posture space can be fully covered using a 26-dimentional space. In practice, the 20 DOF hand articulation, also called intrinsic posture attributes in this dissertation, contains a great deal of redundancy and can be greatly reduced to a lower dimensional. All hand the posture attributes, except for the 3-DOF position, is bounded within fixed bounds, namely the static finger constraints. Posture classification is a mapping of the high dimensional posture space to a discrete set of postures. By dividing the posture attributes into disjoint sections we can divide the multi-dimensional posture space into discrete set of postures represented by volumes in the posture attribute domain. The number of postures increases exponentially with the number of attributes and the number of sections used to divide each attribute. In other words, we can think of the attribute sub-sections as nodes in a tree structure, where every attribute represent a level deeper in the tree structure. The postures are represented by leaf nodes in the tree. The more levels in the tree, the more number of leaf nodes there is, corresponding to more postures in the posture set – an example will follow.

Excluding the hand position, our approach for posture classification is as follows:

First, use high level posture attributes derived from the finger joint angles, which abstract the inter-joint dependency and relate to visual posture attributes; e.g. the finger curvature. The major curvature provides a measure of the finger bending around the MCP joint, and the minor curvature provides a measure of the finger mending around its PIP joint. Together the two curvatures provide a visual interpretation of the fingers posture.

Second, the posture attributes are divided into groups. The advantage of group-subdivision is to be able to use few small DBN models instead of one large model. As described in the Gesture Model Creation section, the size of the model's stochastic matrices equals the cardinality of the posture set. By grouping the posture attributes into independent groups, the cardinality of the resulting sub-groups is much less than the cardinality of a single group.

For example, suppose we have a total of 15 posture attributes; namely the thumb's curvature, abduction and anteposition /opposition, and the fingers' major and minor curvatures and abduction. Suppose also that every attribute is divided into 4 different intervals. Thus the total number of postures as a result of the Cartesian product of all attribute is equal to $4^{15} = 1{,}073{,}741{,}824$. By regrouping the posture attributes into 5 different groups, one for every finger, the total number of states per finger as a result of the split is $4^3 = 64$, and five independent sets of finger postures will sum up to 320 states. The grouping schemes are arbitrary, but the following three schemes are found to be more meaningful:

- One group: (all fingers),
- Five groups: (thumb, index, middle, ring, pinky),
- Three groups: (thumb, (index, middle), (ring, pinky))

Third, we divide the attribute range into as fine granulation as the application requires. This requires analyzing the training data and the gesture set. The gesture set dictates the posture attribute set and its granulation in order to be able to distinguish the different postures in the posture set. Analyzing the histogram of the posture attributes in the training data, provide good candidate for the attribute segmentation points. It is advised to use valley points in the histogram as segmentation points.

Finally, the classified attribute vectors are mapped to integers using the labeling function L, which is a many to one function, labeling similar postures to the same label and rejecting insignificant postures along the way. For example, the fingers abduction (except the thumb) is insignificant to the overall posture except when the finger is straight, which means that when the finger is curved the abduction variations will be mapped to the same posture label.

4.7. Gesture Recognition Results

To demonstrate the precision of the dynamic gesture DBN model, the framework was tested with three dynamic gestures, namely Grasp, Quote, and Trigger gestures. In the grasp gesture, the hand starts from a resting posture and ends up in a fist posture. The quote gesture emulates a double quote, using the index and middle fingers. The trigger gesture emulates triggering a gun. The gestures were chosen to be visually separable to demonstrate the basic capability of the dynamic gesture models. The gestures were acquired using a

CyberGlove at a sampling frequency of 25 Hz. The target gesture sampling frequency was set to 10 Hz, resulting in over-sampling ratio of 2.5. The posture attribute vector was comprised of the five finger minor curvatures. Each finger attribute was divided into four intervals comprising a different finger state. Figure 56 shows the posture attribute vector used for training the dynamic gestures, where (9, 9, and 8) gestures were used to train the dynamic gesture models, respectively.

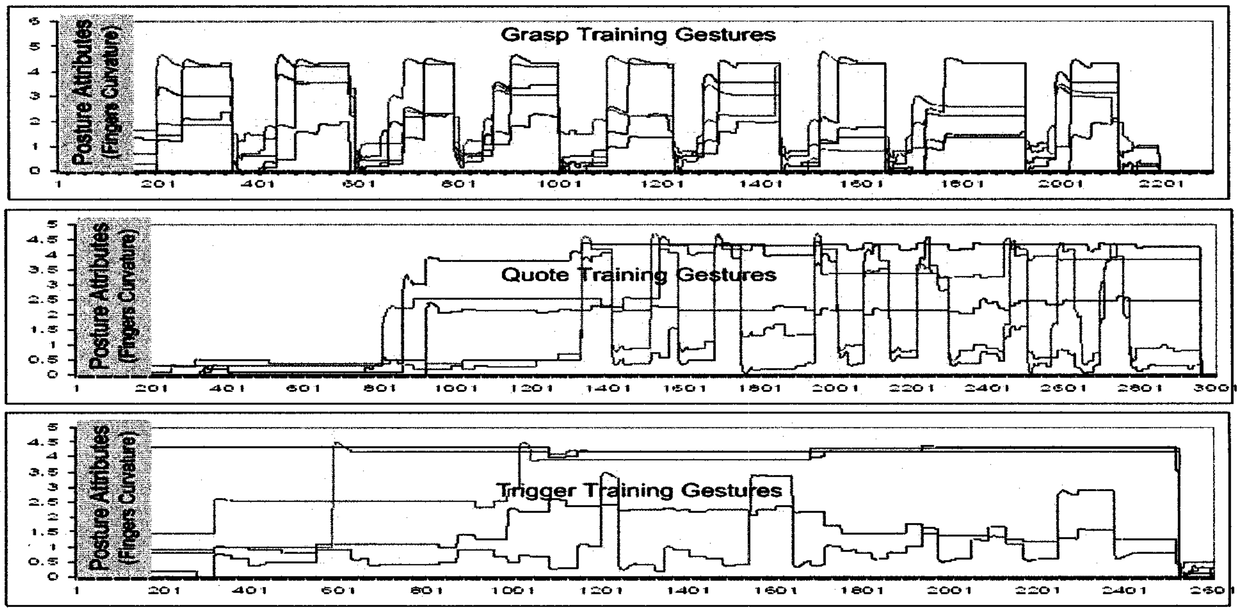


Figure 56: Training samples

The horizontal axis shows the sample number and the vertical axis is normalized curvature measure in the range [0, 2π].

In the grasp gesture case, all fingers start from low curvature and gain curvature as the fingers bend and then move back to resting position. In the case of the Quote gesture all fingers except for the index and middle are bent and the later two fingers stretch and bend as the gesture is performed. The trigger gesture is similar to the Quote with only the index finger stretching and bending.

The cardinality of the posture set Γ was reduced from 4^5 to 5x4 by considering every finger separately, i.e. the dynamic gesture model, $G = \{V_0, M_{1..T}, V_T\}$, is replaced by five independent sub-models of the fingers $G_i = \{V_{i,0}, M_{i,1..T}, V_{i,T}\}$, $i \in [1, 5]$. The advantage/disadvantage of the compound model is that it may allow some inter-finger timing discrepancies. This may play as an advantage in increasing the generality of the model, but can be a disadvantage in reducing the model's ability to distinguish gestures solely on inter-finger timing. In the latter case, different attribute grouping may be required. For example grouping the index and middle finger attributes to catch the two fingers' timing.

A linear probability density functions (n=1, cf. equation (42) and Table 11) were used for both time scaling and attribute acuteness random variables. The maximum time scaling factor (cf. equation (39)) was set to ±30%; i.e. gestures of rates 30% faster or slower than the training sample are considered valid. The posture attribute tolerance was set to half the attribute interval width, which means the attribute probability density function will span the whole attribute interval when the attribute value is centered over the interval, but it will overlap with the adjacent interval otherwise, as shown in Figure 57.

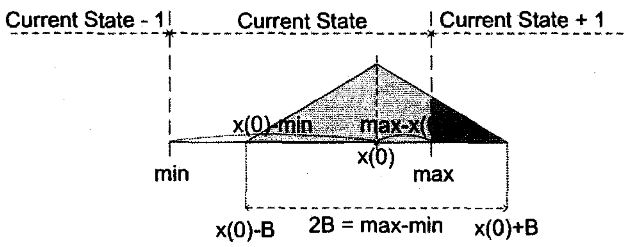


Figure 57: Setting posture attribute tolerance

The posture sequence sliding window size was set 20% larger than the maximum dynamic gesture time period. The posture attributes used in the testing experiment and their corresponding gesture probability measure of three gesture models are shown in Figure 58. The probability of the observed posture, given each gesture using forward time traversal, was evaluated for the three gestures at every sampling period, also shown in Figure 58.

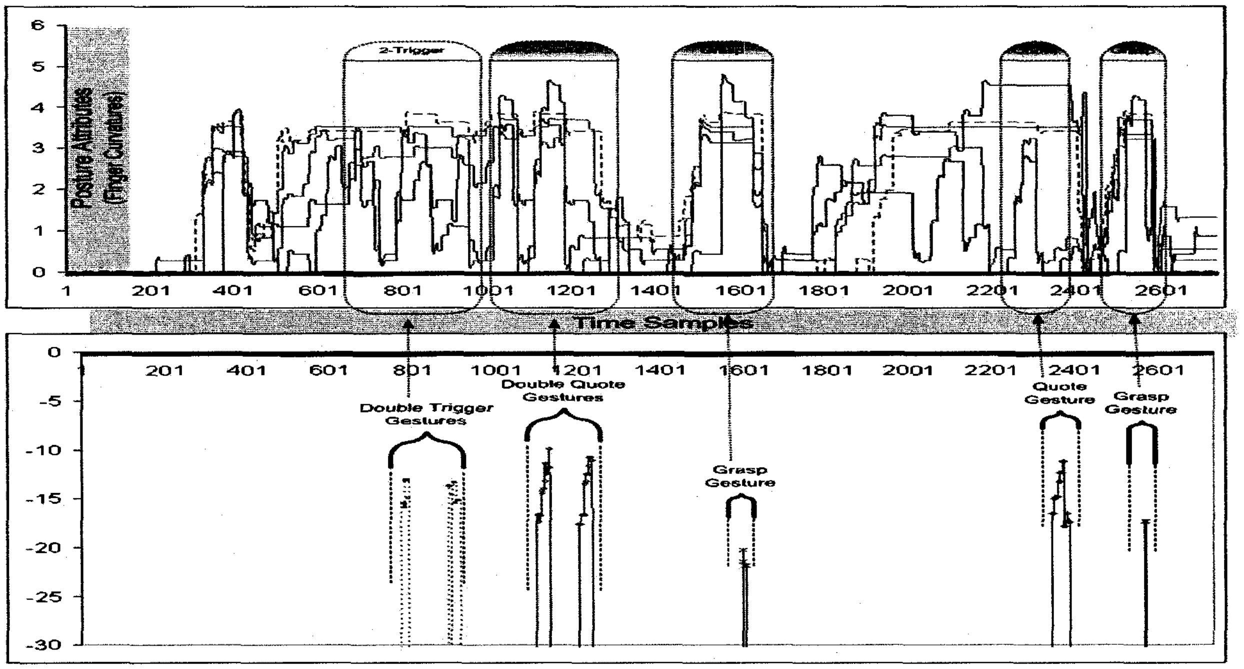


Figure 58: Probability of posture, given gestures, and the segmented gestures detected

A logarithmic scale was used to show the gesture probability for convenience.

We included an association between the gestures detected and the posture attributes that caused the dynamic gesture model to be triggered. The algorithm provided 100% recognition of the input gestures. We recognized that our scheme has a high precision, indicated by the pulsing gesture triggers. We also did not require a threshold for probability measure as the invalid gestures resulted in zero probability. In other words, the gesture model is not based on the nearest neighbor where it continuously triggers the most probable gesture, but it remains idle except when a gesture is encountered.

The advantage of backward time-traversal is demonstrated by another test, shown in Figure 59.

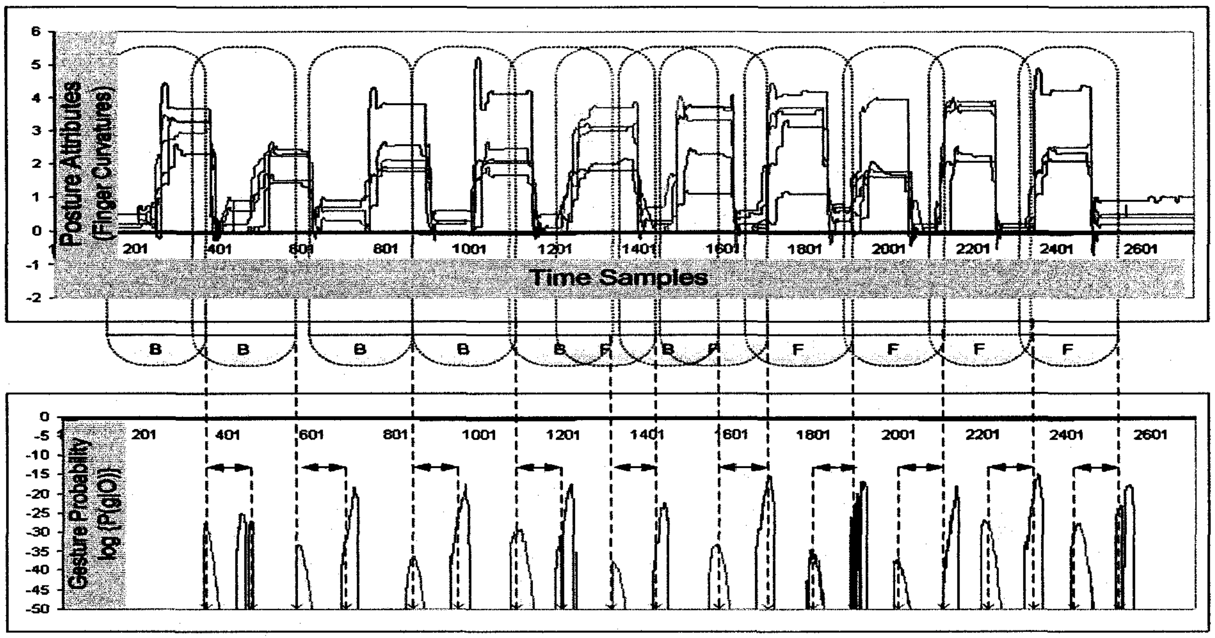


Figure 59: Forward and backward time traversal – recognition latency

Ten gestures were recognized in both forward and reverse time traversal modes. The resulting gesture probability measure is shown in logarithmic scale for convenience. The position of the forward and backward sliding window at the time of gesture recognition is shown, labeled F and B respectively. For convenience, we only show the last six forward sliding windows and the initial six backward sliding windows with only two overlapped. The latency due to oversized sliding window, in the case of forward time traversal, is shown in comparison to the backward traversal case (shaded in gray).

To test the recognition success rate of our scheme over multiple users, we collected dynamic gesture samples from six different operators with different hand sizes – they included a football player and a Chinese girl to expand the hand sized range. We conducted a test using sample sets of 14 gestures, including 7 dynamic gestures and 7 postures. The gesture samples were acquired using a CyberGlove. The gestures trained on the first operator were used to recognize the gesture samples from the rest of the operators; we estimated the recognition rate at 40%. Then, we progressively trained the gesture models on the least successful operator, using the gesture model merge operation described earlier, and then re-performed the recognition experiment on all the operators. The results of the experiment are shown in Figure 60.

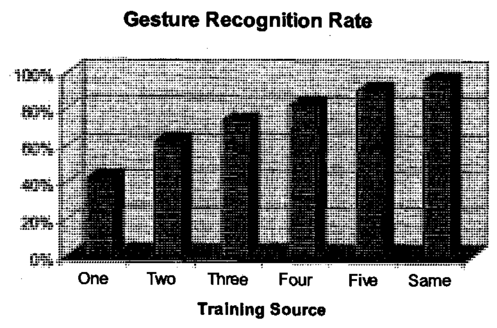


Figure 60: DBN Model Recognition Rates

The success rate progressively increased as we trained on more operators, as expected. We realized that the recognition rate is improved dramatically if we use the gestures trained on the same operator to recognize his gestures. A plausible explanation is due to the variation in the CyberGlove calibration between the operators.

We tested the integrated vision-based hand tracking and gesture recognition frameworks on a set of six dynamic gestures: fist, fan, quote, trigger, prm123, and imr123. The fist gesture is similar to grasp gesture described earlier. The fan posture is similar but the fingers flex in sequence – pinky, ring, middle, index and thumb.

The quote gesture emulates a double quote, using the index and middle fingers. The trigger gesture emulates triggering a gun; and the count 1,2,3 gestures are counting using the pinky, ring and middle fingers in the first case and using the index, middle and ring fingers in the second case. The DBN model was trained using a separate training set acquired using the CyberGlove. The posture attribute vector consists of 13 attributes: thumb curvature and fingers abduction and major and minor curvatures. The training data were acquired using the CyberGlove and were used to create the gesture models. The testing data was acquired simultaneously from the CyberGlove and two cameras at a rate of 10 Hz / fps respectively. The fuzzy posture set provided by the vision-based hand tracking sub-system was configured to include the most probable {1, 5, 10, 20, 40} postures, given the observed features. The probability measures provided by the DBN models are shown in Figure 61.

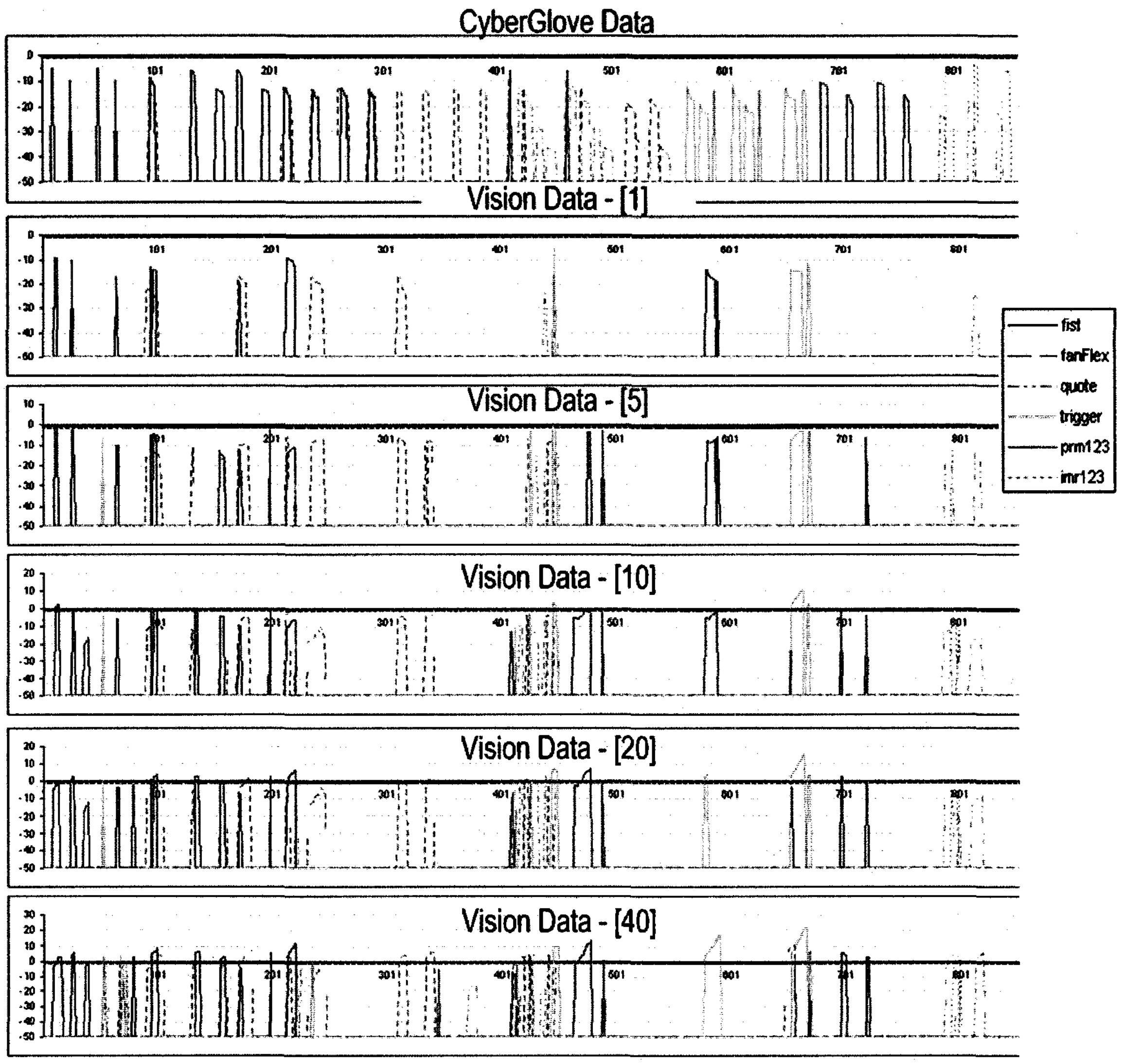


Figure 61: Dynamic Gesture Recognition

The model provided 100% recognition using the CyberGlove data. The vision system showed recognition improvement from 25% using a single posture per sample to around 70% recognition using 40 postures per sample, with 10 postures per sample providing the best balance between recognition and false trigger. The corresponding posture probability after increasing the posture attributes offset tolerance in shown in Figure 62.

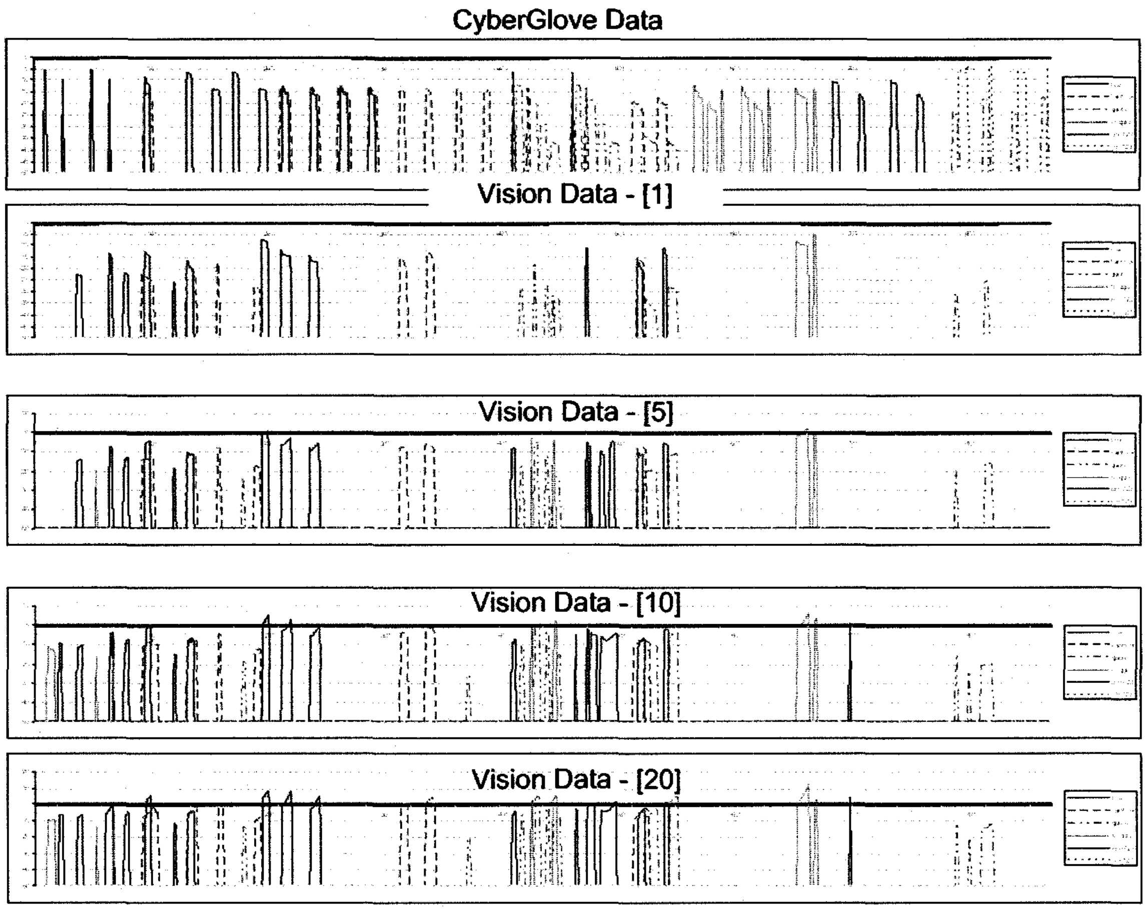


Figure 62: Gesture Probability After Increasing the Attributes Offset Tolerance

The integrated vision-based and gesture recognition system shows promising results but indicates that the vision-based tracking requires further enhancement in terms of posture estimation integrity and noise reduction. We noticed that in some cases the fist and fan gestures provided close probabilities, even in the CyberGlove case. Since the only difference between the two gestures is in the inter-finger timing, it is evident that in some cases, dividing the gesture model to independent five models can cause errors. We recommend dividing the fingers into 3 groups as follows: {thumb, (index, middle), (ring, pinky)}. The posture classification scheme is application dependent and may require further testing, but we will consider it outside the scope of this dissertation.

4.8. Conclusion

We presented a framework for dynamic gesture recognition and segmentation. The framework emphasizes the gesture dynamics by timely tracking the acquired gestures within predefined bounds derived form the training samples. The posture attribute bounds are learned from a few segmented gesture samples that are used as particle samples in the posture attributes space. The framework is similar to HMM but supersedes HMM in two main issues. Firstly, the model does not make an assumption of the existence of some hidden states, thus it is easier to build and does not require estimation of the number of hidden states or the state interconnection topology. More importantly, it does not make the assumption that the state transition probability is constant with respect to time. We argue that in our framework, the dependence on the training samples makes it less prone to detect false positives. The gesture generalization based on particle training examples and radial basis probability density functions is suitable for the intrinsic posture attributes, taking advantage of the static constraints of the hand posture. In other words, all attributes are bounded by predefined intervals and probability distribution over the intervals at any time can be approximated by particles associated with radial basis probability density functions.

Secondly, since the gesture model is derived directly from the probability functional used in the gesture recognition, the gesture recognition is directly determined from the model and does not require complex calculation, such as Viterbi algorithm, as in the case of HMM.

3D hand trajectories referenced to body are bounded by the hand length and can be added to the posture attributes. The trajectory space can be divided into circular ring sectors around the body. On the other hand, HMM is more suitable for 3D hand trajectory in absolute 3D space (cf. [65]), because the position vectors are not bounded to fixed ranges. Thus the HMM will generalize the training data better than the radial basis pdf's. To include hand trajectory and finger postures to our model, we recommend using a hybrid HMM/DBN model.

Chapter 5

5. The Role of Context in Gesture Applications

Gesture is a form of oral communication and has many similarities to conversations. To understand the context of a gesture, we investigate human conversations in general and relate them with our research in the context of hand gestures. Human communication has been studied by researchers in the fields of psychology, linguistics, anthropology, ethno methodology, discourse analysis, and conversation analysis. R. K. Sawyer [89] in his book "*Creating Conversation*" studies conversations as a form of a creative process. More generally, human to human communication includes oral communication e.g. conversations and gestures, written documents, e.g. memos, letters, emails, newspaper reports etc., technology mediated communications, e.g. emails, chatting and video conferencing, radio, television. Several influential modern theorists have even argued that all written language – not just letters and emails, but even novels and historical records – employ strategies which are similar to these employed in conversations. In other words, various forms of communication, oral and written, share the same goal but may differ in form.

The metaphor of all forms of creative communication is a mediated performance in which an artist is performing to an audience; but sometimes the audience can just see the end result and not the creating process. Interactive communication, including conversation and gestures, are an example of a more general characteristic of group behavior called emergence, where novel and coherent meanings arise in the process of interaction. A recent mathematical study of group decision making [90] has proved that group behavior is emergent and thus unpredictable; even if the views of all the participants are known a priori, because new ideas can emerge and some may change their opinion as a result of some other arguments.

5.1. Gesture Attributes

The following attributes of the process of human communication provide hint to the relevant context information associated with gesture recognition. Firstly, gestures are improvised, i.e. it is spontaneous. The element of improvisation is inversely proportional to the time it takes to prepare or anticipate a response. Figure 63 shows some means of communication and their preparation times, the improvisation level is noticeably inversely proportional to the preparation time.

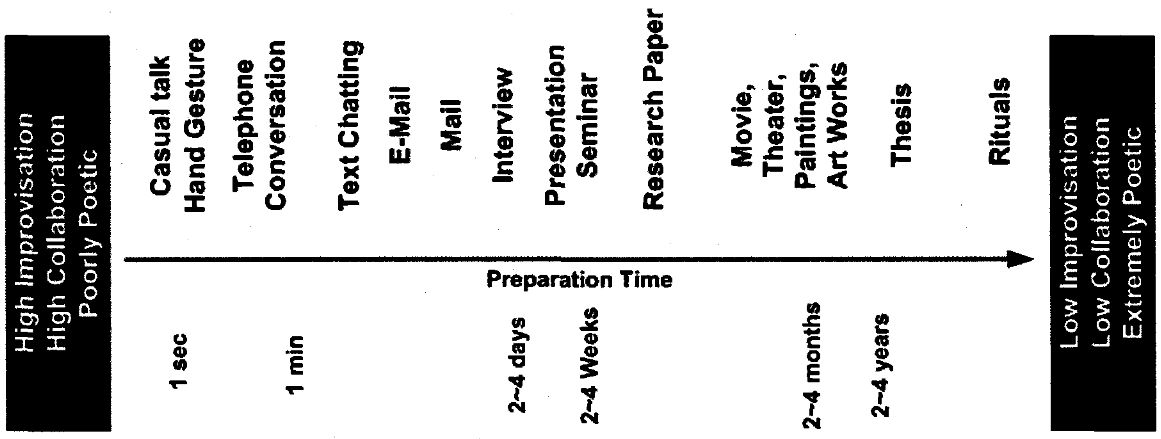


Figure 63: Improvisation Chart

Scripted forms of communication are characterized by formality, relatively long period of preparation, and slower response time; they also tend to have higher intellectual value and may, therefore, require storage and retrieval. Improvised communication, on the other hand, is characterized by fast response time and little preparation time; mythically, they tend to have less intellectual value and usually they are neither stored nor retrieved.

Secondly, gestures are indexical [91], which means they do not have a definitive meaning but derive their logical meaning from contextual and co-textual factors, usually called context of situation [93]. A gesture is indexical when it is indicative of something about the situation beyond the literal meaning of the gesture. Thirdly, sign gestures are not all purely symbolic, and some are in fact mimetic or deictic (these are defined by Quek [97] as act gestures where the movements performed relate directly to the intended interpretation). Mimetic gestures take the form of pantomimes and reflect some aspect of the object or activity that is being referred to. These are similar to classifier signs in American Sign Language (ASL) which can represent a particular object or person with the hand shape and then act out the movements or actions of that object. Gestures may also accompany speech and play a meta-talk role, which is an additional form of information about the speech. Kendon [94] described one of the roles of hand gesticulations that accompany speech as providing images of the shapes of objects, spatial relations between objects or their paths of movement through space. These are in fact some of the same functions of classifier signs in American Sign Language. Fourthly, a gesture has nuance aspects that may be used to add/differentiate meaning in the form of speed, acuteness, and beat variances. A technique, called 'timing', is commonly used by animators to give meaning to their animations beyond the text and setting [92]. Figure 64(a) shows examples of the sign ASK with different types of aspectual inflections. Generally, the meanings conveyed through these inflections are associated with aspects of the verbs that involve frequency, duration, recurrence, permanence, and intensity, and the sign's movement can be modified through its trajectory shape, rate, rhythm, and tension [95], [96]. Klima and Bellugi [96] list 8-11 different types of possible inflections for temporal aspect. Another type of inflection that can occur is person agreement (first person, second person, or third person). Here, the verb indicates its subject and object by a change in the movement direction, with corresponding changes in its start and end locations, and hand orientation. Figure 64(b) shows the sign ASK with different subject-object pairs.

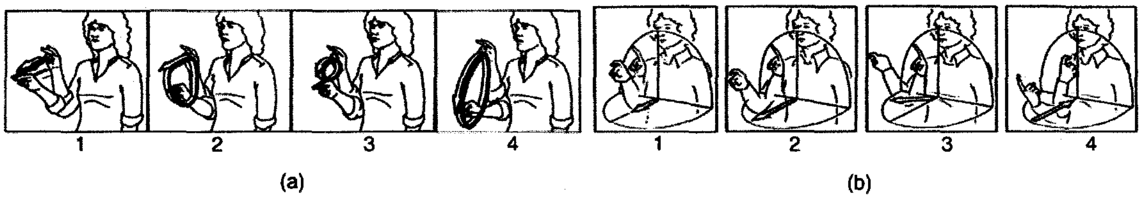


Figure 64: Grammatical inflections of the sign ASK – Adopted from [42]

Fifthly, a gesture is rule based; rules govern turn taking, and engaging and releasing the gesturing mode. The rules are usually drawn from external factors, such as the setting (objects and relevant services/applications available, location, culture) and people (performers, addressees and bystanders, and their status and relationships). Sixthly, a gesture is evolving; new gestures may be created as the need emerges, and new meanings may be induced from usage. The evolution process may take a long period of time and involves many people improvising and collaborating to reach a form of consensus. The creation of gestures is similar to the creation of new languages; there must be real need to motivate the process and a long evolution period to refine the gesture vocabulary till it reaches a complete, comprehensive, and useful gesture set. Moreover, there can be different sets of gestures available in different communities, affected by their environment, history of use and their communication goals. Those gesture vocabularies, evolve overtime, and can be shared among user communities. Finally, a gesture is collaborative. It takes a source (gesturer) and a recipient to perform and comprehend a gesture. As a form of creativity gesture is also collaborative and requires a level of intelligence to make it work efficiently. Thus, gestures must be studied as a collaborative process, which only works when shared effectively with others – including smart machines, in a gesture-enabled user interface. Thus, an accurate dynamic gesture recognition system, with high recognition rate and low false trigger rate, is an essential part of context-aware gesture enabled user interfaces.

Table 13: Gesture attributes and the associated requirement for gesture enabled UI

#	Attribute	Functional Requirement in the User Interface
1	Improvisation	Gestures emerge unexpectedly, and do not follow statistical norms
2	Indexical ability	The interpretation of gestures depends on external factors – context of the situation
3	Meta-talk	Some types of gestures are intended to complement speech
4	Nuance	Small variances in gestures may add new meanings, particularly timing
5	Rule based	The rules of engagement into gestures must be defined, the rules may vary based on external factors
6	Evolving	The gesture set is dynamic and may change over time an easy way to add gestures and combine old and new ones is required
7	Collaborative	Low latency and high accuracy gesture recognition is required to maintain the flow of collaboration

Table 13 summarizes the discussed gesture attributes and their associated functional requirements for the context aware, gesture enabled system.

To design a context-aware, gesture-enabled, user interface, we first must identify what is relevant to the process of recognizing hand gestures and the successful interpretation of the gestures in the form it was intended for.

When it comes to gesture interpretation, the relevant context can be categorized into three main categories: First, the *audience*, which include the gesture operator and the people interacting with her. The *audience* can be classified on three different levels:

- *Personal Level*: Every operator has her special way of performing the gesture. It has been shown that the gesture recognition rate is increased by training the gesture models using data acquired from the same operator. This context information includes the hand model, specific for the user, the gesture models associated with the user, and the user preference associated with the applications.

- *Social Level*: It includes frequent interactions with the operator's social network. It defines the frequent vocabulary and preferences and can be used to resolve ambiguity.
- *Community Level*: It includes the cultural factors, and defines the common facts shared by people in the same community but may vary from one place to another.

Second is the *available services* or active applications supported by the user interface (the non-human factor), which define the active gesture set. The available services are usually location bound. Third are the gesture *intrinsic factors*. For example, timing is an essential part of gesture intent, variants in the speed and frequency (beat) of gestures can add to the meaning intended by the operator. Spatial and acuteness factors are also an intrinsic property of a gesture. Intrinsic gesture factors are best recognized at the gesture model level. The DBN model we propose can accommodate variants in timing and spatial and acuteness, it also provides a means to group related gestures into more generic models.

5.2. Agent Based Gesture Interpretation

We are using an agent-based system to abstract the loosely coupled aspects of the gesture context. Research on agent and multi-agent systems is a large and quickly growing subfield of distributed artificial intelligence (AI). Perhaps the most general way in which the term *agent* is used is to denote a hardware or (more usually) software-based computer system that possesses the following properties: *Autonomy*, being able to operate without direct intervention, and having some control on their decisions and internal state; *Social ability*, interacting with other agents and possibly humans; *Reactivity*, perceiving their environment and respond in a timely fashion; and *pro-activity*, able to exhibit goal-directed behavior.

Excellent introductions to the agent- and multi-agent- based systems can be found in [108]–[110]. More in-depth discussions of multi-agent systems issues are in [111] and [112]. Following [113], a tentative definition of an agent can be formulated as follows: "An agent is a (computer) system that is situated in some environment and that is capable of autonomous action in this environment in order to meet its design objectives." The environment in which an agent operates can be either real or simulated. The key advantage of agent-based systems is the fact that agents form a new and useful abstraction tool for analyzing and solving complex problems.

The phrase "agent-based system" can be used in two contexts that are not necessarily closely related [99]. One usage of that phrase is to describe systems of *autonomous agents*, usually autonomous robots. The second usage is to refer to systems consisting of *software agents*; loosely coupled software modules collaborating by sharing information to achieve a common goal. The former systems deal with all aspects of autonomous robot multi-tasking, such as task planning, mobile platform issues, and robot collaboration, whereas the latter systems employ an agent-based approach to *analyze/retrieve/process data* sequences and do not have to be owned by autonomous agents. The most common use of agent-based systems is to perform effective task decomposition by employing multiple agents with different processing competencies (visual capabilities) at varying levels of abstraction to solve specific subtasks and then to synthesize a solution. The agents are therefore usually used as means to effectively modularize and/or parallelize the design of a computer vision system.

Figure 65 shows our agent based approach to gesture interpretation.

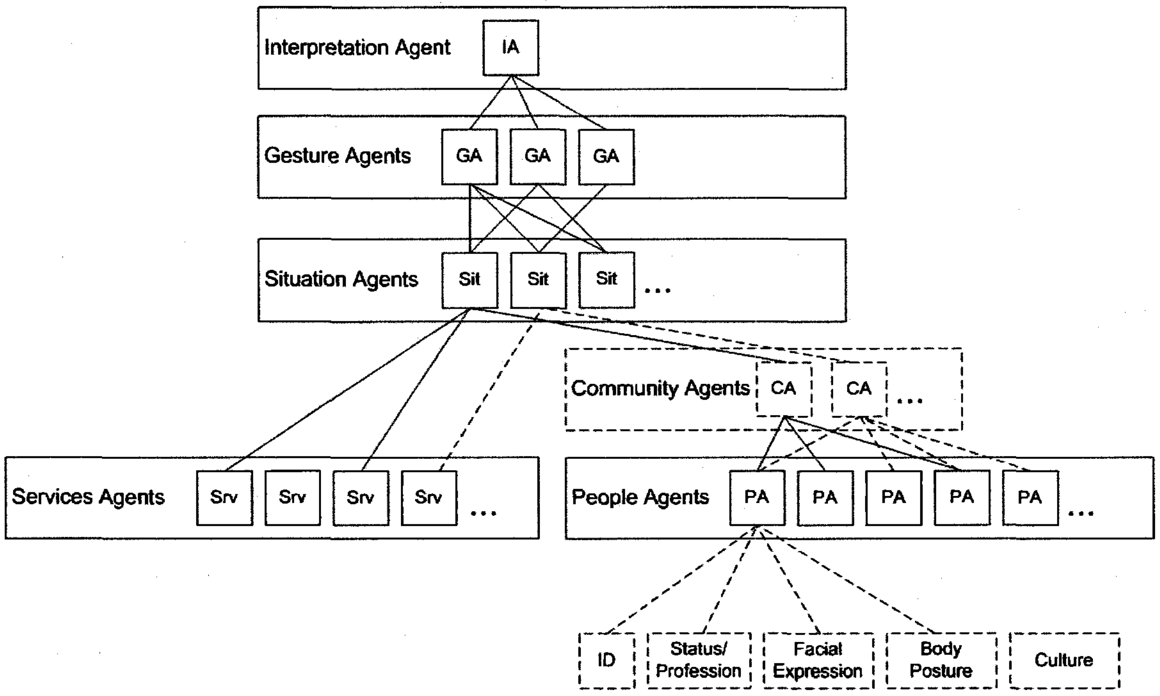


Figure 65: Relevant Context in Gesture Interpretation – Agent based approach

The relevant context to the process of gesture interpretation, namely the people and services, which are extrinsic to the gesture model, are shown at the bottom, indicating low level of abstraction. Context aggregation begins by introducing the community agents, relating group specific aspects. The situation aggregators integrate the people and services aspects into more meaningful entity relevant to gesture interpretation, it includes the roles played by the gesture participants (operator and audience) and it defines the operator's possible gesture communication goals. Intrinsic context, e.g. timing and special variants are addressed in the gesture recognition process and are presented as various gesture candidates in the form of gesture agents.

Gesture monitors are created to represent the existence of a gesture in the active gestures set. When a gesture monitor detects a gesture it creates a gesture agent, which includes information about:

- The canonical probability of the gesture, given the current situation,
- The probability of gesture, given the observed posture, and
- The possible interpretations of the gesture.

The gesture interpretation module oversees the gesture interpretation process and provides the candidate gesture interpretation to the user interface.

Other abstract entities relevant to process of gesture recognition and interpretation are:

- *Role*: we perform many different characters each day, presenting different facets of ourselves to family, friends, and coworkers. We also create different characters to new situations, and participate in different kinds of interactions. In gesture enabled perceptual user interfaces, one or more persons can play the role of the gesture operator (or speaker); the rest of the people may play the role of the audience or a bystander. Depending on the situation, the gesture can be used to control a computer system, or provide pictorial explanation to her speech (pantomime).
- *Performance*: is defined as a session in which the operator played a role to achieve his communication goal. A good performance achieves the conversation goals by allowing performers to improvise at their comfortable rate, which psychologists call 'the flow' or 'peak experiences' as called by the psychologist Mike Csikszentmihalyi [101]. We experience flow whenever we are doing something where our skills closely match the challenges of a situation. You will not experience flow if the situation isn't challenging enough, you just get bored, or if a conversation is over your head, for example everyone is speaking too fast that you can not understand what is going on, then you will not have flow either; instead you will just get frustrated. We experience

flow in between these extremes, in conversations where the creative challenges are just right for our skills. Performance analogy in user interface design is the usability study, where a session is stored and retrieved for evaluation. The performance must contain timely (frame-by-frame) retrieval of low level sensor data and high level system observations. For training purposes, the intended interpretation and reference observation are provided by a domain expert.

- *Rehearsal*: is the training process to fine-tune the different parameters of the system. Human communication, including conversations and gesture, are their own rehearsals. Unsupervised and supervised training of gesture-enable context-aware user interface is required to improve the system performance and increase its usability. The relevant aspects can be re-evaluated and the probabilistic priors embedded in the system must be frequently updated, with the goal of increasing the overall system usability.

The above principles are abstract concepts general enough to fit any gesture-enabled perceptual user interface application. In simple applications a single user and a single application or service is active at a time, but in complex applications multiple applications and multiple users may be operating simultaneously and collaboratively. For example we conducted an experiment to recognize alphabet signs from the *American Manual Alphabet for the Deaf*, shown in Figure 66.

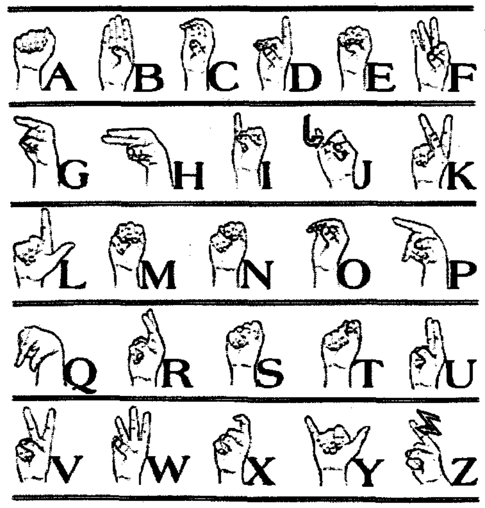


Figure 66: American Manual Alphabet for the Deaf

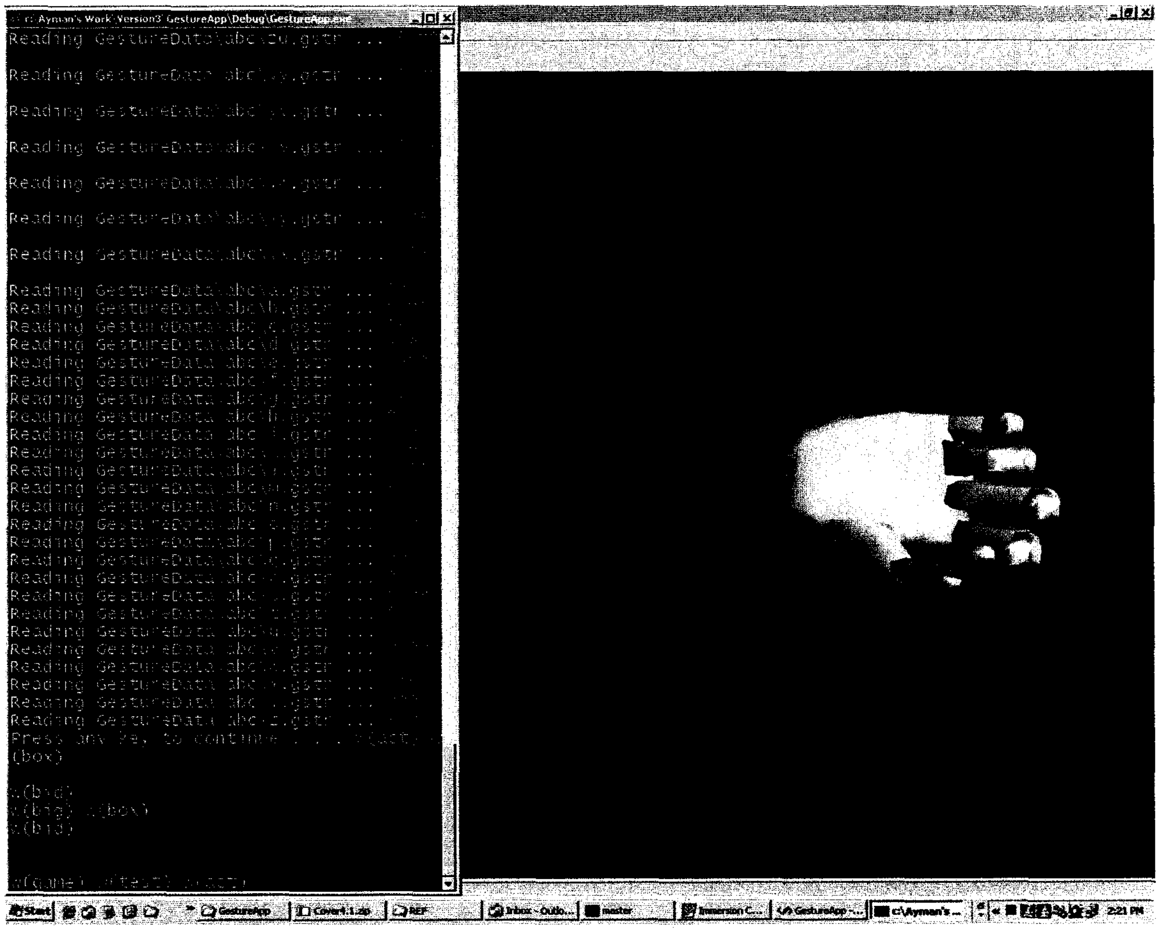
We utilized the dynamic aspect of the DBN model by modeling the transition between the different letters in the alphabet as they are spelled using manual gestures. The gesture set we modeled included 347 hand gestures, including 322 dynamic gestures and 25 postures, as shown in Table 14. For rehearsal we used CyberGlove data acquired at 50Hz. We collected the training data of every couple in pares of alternate orders, e.g. 'ac' and 'ca' were trained from the same training sample. The objective was to recognize letter couples as they are preformed and to be able to recognize words from a small dictionary provided to the application.

To relate to the context of gesture as previously introduced, the human factor is represented by the system operator, who is identified to the system and the corresponding gesture models are uploaded, as well as his CyberGlove configuration file. The service provided by the system is Alphabet / Word recognition, which mandates the corresponding gesture set.

Table 14: Alphabet Gesture Set

	a	b	c	d	e	f	g	h	i	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	a	ab	ac	ad	ae	af	ag	ah	ai	ak	al	am	an	ao	ap	aq	ar	as	at	au	av	aw	ax	ay	az
b	ba	b			be				bi		bl	bm		bo	bp		br	bs		bu				by	bz
c	ca		c		ce			ch	ci	ck	cl			co			cr	cs	ct	cu	cv			cy	
d	da			d	de				di					do			dr	ds		du				dy	
e	ea	eb	ec	ed	e	ef	eg	eh	ei	ek	el	em	en	eo	ep	eq	er	es	et	eu		ew	ex	ey	ez
f	fa				fe	f			fi		fl			fo			fr	fs		fu				fy	
g	ga				ge		g	gh	gi		gl			go			gr	gs		gu				gy	
h	ha		hc		he		hg	h	hi					ho				hs	ht	hu				hy	
i	ia	ib	ic	id	ie	if	ig	ih	i	ik	il	im	in	io	ip	iq	ir	is	it	iu	iv	iw	ix	iy	iz
k	ka		kc		ke				ki	k				ko				ks							
l	la	lb	lc		le	lf	lg		li		l			lo				ls		lu					
m	ma	mb			me				mi			m	mn	mo				ms		mu				my	
n	na				ne				ni			nm	n	no				ns	nt	nu				ny	
o	oa	ob	oc	od	oe	of	og	oh	oi	ok	ol	om	on	o	op		or	os	ot	ou	ov	ow	ox	oy	oz
p	pa	pb			pe				pi					po	p		pr	ps		pu				py	
q	qa				qe				qi							q				qu					
r	ra	rb	rc	rd	re	rf	rg		ri					ro	rp		r	rs	rt	ru				ry	
s	sa	sb	sc	sd	se	sf	sg	sh	si	sk	sl	sm	sn	so	sp		sr	s	st	su	sv	sw		sy	
t	ta		tc		te			th	ti				tn	to			tr	ts	t	tu		tw		ty	
u	ua	ub	uc	ud	ue	uf	ug	uh	ui		ul	um	un	uo	up	uq	ur	us	ut	u	uv	uw	ux	uy	uz
v	va		vc						vi					vo						vu	v			vy	
w	wa				we				wi					wo				ws	wt	wu		w		wy	
x	xa				xe				xi					xo						xu			x	xy	
y	ya	yb	yc	yd	ye	uf	yg	yh	yi			ym	yn	yo	yp		yr	ys	yt	yu	yv	yw	yx	y	
z	za	zb			ze				zi					zo						zu					z

Table 14 specifies the gesture set utilized by the system. 347 gestures of character posture and character-to-character transition were used. The People-, Community-, and Situation-agents are not represented in this simple application. The gesture agents are data structure created (emerge) when a DBN model for a gesture triggers a recognition event. The data structure includes: the gesture name (e.g. 'ac'), the start- and end- times of the gesture trigger, and the integrated probability of the gesture over the recognition time, i.e. *[name, start, end, probability]*. The interpretation agents are concatenated characters (gestures) that represent whole or parts of a word from a dictionary, i.e. *[sub-word, start, end, probability]*. The dictionary is an ontology of how the gestures must be combined together to provide a meaningful interpretation. Figure 67 shows the word recognition application we developed.



```
c:\Ayman's Work\Version3\GestureApp\Debug\GestureApp.exe
Reading GestureData\abc\[illegible].gstr ...

Reading GestureData\abc\[illegible].gstr ...

Reading GestureData\abc\[illegible].gstr ...

Reading GestureData\abc\[illegible].gstr ...

Reading GestureData\abc\[illegible].gstr ...

Reading GestureData\abc\[illegible].gstr ...

Reading GestureData\abc\[illegible].gstr ...

Reading GestureData\abc\a.gstr ...
Reading GestureData\abc\b.gstr ...
Reading GestureData\abc\c.gstr ...
Reading GestureData\abc\d.gstr ...
Reading GestureData\abc\e.gstr ...
Reading GestureData\abc\f.gstr ...
Reading GestureData\abc\g.gstr ...
Reading GestureData\abc\h.gstr ...
Reading GestureData\abc\i.gstr ...
Reading GestureData\abc\[illegible].gstr ...
Reading GestureData\abc\[illegible].gstr ...
Reading GestureData\abc\[illegible].gstr ...
Reading GestureData\abc\n.gstr ...
Reading GestureData\abc\o.gstr ...
Reading GestureData\abc\p.gstr ...
Reading GestureData\abc\q.gstr ...
Reading GestureData\abc\[illegible].gstr ...
Reading GestureData\abc\[illegible].gstr ...
Reading GestureData\abc\[illegible].gstr ...
Reading GestureData\abc\[illegible].gstr ...
Reading GestureData\abc\[illegible].gstr ...
Reading GestureData\abc\[illegible].gstr ...
Reading GestureData\abc\[illegible].gstr ...
Reading GestureData\abc\[illegible].gstr ...
Reading GestureData\abc\[illegible].gstr ...
Press any key to continue . . . [illegible]
(box)

[illegible](bad)
[illegible](big) [illegible](box)
[illegible](bid)


[illegible]
```

Figure 67: Alphabet/ Word Recognition Application

The status widow on the right hand side shows the words detected and the gestures models which are loaded. On the right hand side, the hand mod shows a real-time visualization of the hand posture.

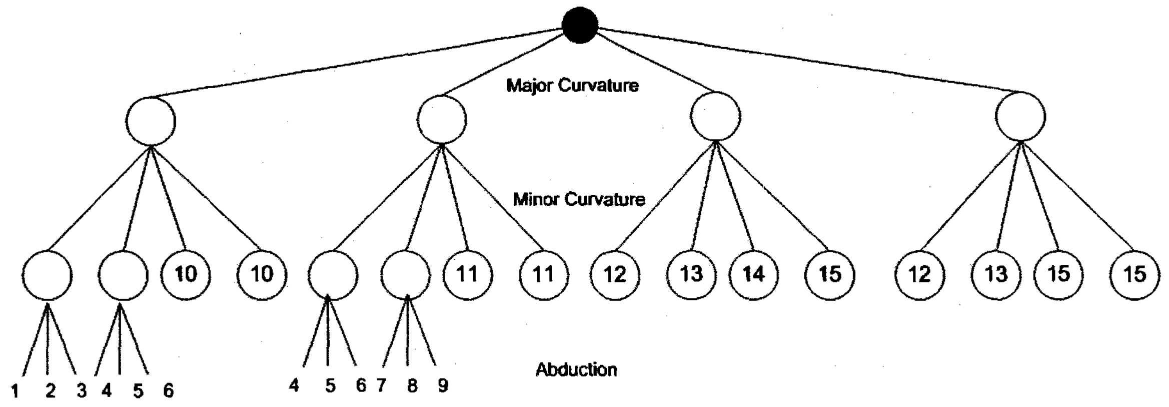
The large number of gestures created two types of challenges to the system design process. The first challenge is a *scaling challenge.* The gesture training data is conspired by performing 186 training sessions, where 25 training session are performed for the 25 character postures and 161 training sessions are performed for the 322 gesture transition gestures – as they were performed in couples. The segmentation of the gesture training data had to be automated to cope with the huge number of training gestures. A Python script was developed to recognize the posture attribute transitions, which are recognized as peaks in the posture attribute differentials. Segmentation is performed in relatively constant intervals around the peaks. A validating algorithm was developed to check the sanity of the proposed segments, and an oracle was used to validate the suspicious segmentation.

The total number of training gestures was 2951, corresponding to an average of 8.5 training samples per gesture. Twelve posture attributes were selected as follows:

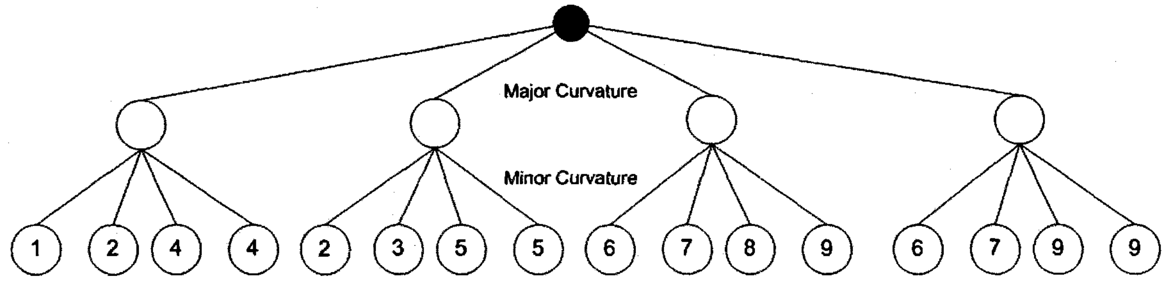
- Thumb : Minor Curvature, and CMC
- Index : Major Curvature, Minor Curvature, and Abduction
- Middle : Major Curvature, Minor Curvature, and Abduction
- Ring : Major Curvature, and Minor Curvature
- Pinky : Major Curvature, and Minor Curvature

The index and middle fingers abduction were essential to distinguish the letters U and V (cf. Figure 66), but was not required in the case of the pinky and ring fingers.

The total number of posture attributes is 12 attributes, which were divided into 5 groups as follows: {thumb: 2, index: 3, middle: 3, ring: 2, and pinky: 2}. The number of dividing intervals per attribute was as follows: {(3, 3), (4, 4, 3), (4, 4, 3), (4, 4), (4, 4)}, respectively. The nine postures associated with the thumb were labeled by a *one-to-one* mapping onto the integer range [1, 9]. The labeling for the postures associated with the index and a middle finger is shown in Figure 68(a) and the labeling for the postures associated with the ring and pinky are shown in Figure 68(b).



(a)



(b)

Figure 68: Posture Labeling for:
(a) Index and Middle fingers, (b) Ring and Pinky fingers

The labeling functions are designed to be symmetrical w.r.t. the order of the curvature attributes. In other words, the opposite order of curvature attributes will result in an equivalent labeling scheme. Moreover, the finger abduction at high curvature is ignored.

The second challenge is the *high false trigger rate.* As the gesture set grows larger, it is possible to keep the recognition rate close to 100%. In our experiment the positive trigger rate was at 89%, but the false trigger rate was at 84%, which means on average there is 4.58 letters are falsely recognized for every correctly recognized letter.

Table 15 shows the recognition rates in more details. The first column in Table 15 shows the words used to evaluate the system performance. The following columns shows the number of characters in each word, the number of gestures in each word, the number of recognized gestures, the number of missed gestures, and the number of false triggers in each word respectively.

Table 15: Gesture Agents Recognition Rates

Word	Num Characters	Num Gestures	Recognized	Missed	False Triggers
xylophone	9	17	12	5	86
work	4	7	4	3	30
word	4	7	5	2	25
way	3	5	5	0	2
wait	4	7	6	1	21
telephone	9	17	17	0	102
source	6	11	9	2	57
optimise	8	15	14	1	78
heat	4	7	7	0	49
head	4	7	7	0	48
game	4	7	7	0	37
fault	4	7	7	2	24
box	3	5	5	0	33
bid	3	5	5	0	13
bid	3	5	5	0	12
back	4	7	7	0	13
act	3	5	5	0	16
Total	79	141	127	16	646
			89%	11%	84%

The number of gestures expected includes the word characters and character transitions (2n-1), where n is the number of characters.

Part of the missing positive recognition errors was due to lack of posture attributes such as hand position and orientation. For example the letters H and U are distinguished mainly by the orientation of the hand, which was not included in the posture attributes. We also omitted the letter J because it is only distinguished from the letter I by trace of the hand position, also not included in the posture attributes. The false trigger rate could be reduced by filtering characters recognized for a period less than 200 ms (corresponding to 10 samples). The total number of filtered false triggers was 120, reducing the false trigger rate to 504%. Using the list of test words as a dictionary we utilizing a character sequence continuation algorithm.

At the word recognition level, we ensure that all letters are detected through a single character or a character transition, i.e. "act" = {a, ac, c, ct, t}, {a, c, t}, {ac, t}, {a, ct}, etc., where overlaps increase the total probability of the word. We correctly recognized 19 words out of 21 words – recognition rate of 90%.

5.3. The Big Picture

Intelligent environment researches have been conducted by technology companies, e.g. Microsoft Smart Living, Xerox PARC Parc Tab, AT&T Laboratories Active Badge, and universities MIT put-this-here, Georgia Tech Cyber-Guide and Classroom 2000, Stick-e-notes Kent. It mainly focused on device interconnectivity and building operating systems and middleware software that support heterogeneous device communication. The main components of the context-aware system design according to Meyer and Rakotonirainy [78] are shown in Figure 69.

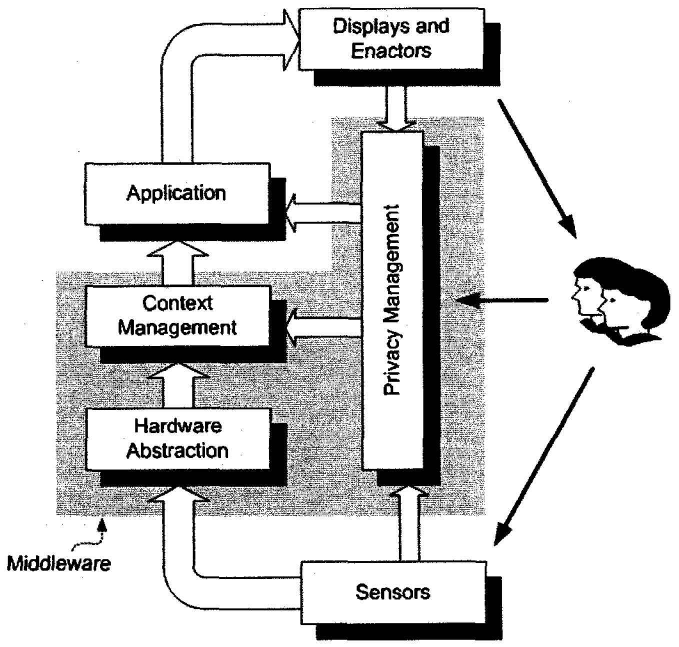


Figure 69: Basic Components of a context-aware system interacting with users

The system basic components include:

- Input hardware and software sub systems such as hand trackers, mouse, keyboard, video cameras, etc.
- Output hardware and software sub systems, such as displays and projectors
- A context-aware application running on the available networked processing platforms
- A context-aware middleware providing programming abstractions that hide the details of and mask the heterogeneity of the underlying networks, hardware, operating systems and programming languages.

- Privacy Management protecting the people's privacy from being violated by the system.

User identification and localization in context-aware systems is done in several ways, e.g. user log in, fingerprint identification, active badges, mobile devices such Parc Tab, and face recognition. The recognition devices have varying degree of recognition success that prohibits the widespread of its use. Voice and Gesture recognition as an integral part of such systems can provide a versatile way of user interaction with the system, but it remains in the far future for the reasons of low success rate and slow response time.

The role of context, in our point of view, should address both the Positivist and the Phenomenological views of context. On one hand, from a positivist point of view, the relevant aspects of context must be defined for the context-aware system at the time it is designed. Low level sensor data such as Data Glove raw data or image features has to be identified because the recognition algorithms use it as input. Hand models and kinematics must be identified as well as gesture models. On the other hand, from the phenomenological point of view, interdependency between these pieces of information can vary from one situation to another based on external factors we call the context. The current situation of the system must be re-evaluated continuously to allow new situations to emerge and old ones to be eliminated. The system must be able to react to new situations, by modifying its topology in terms of interconnectivity and interdependency between its components.

Stochastic processes, such as the observation model and the gesture recognition model, and agent-based system architecture, such as the gesture interpretation model, are suitable for topological variations in response to the context. Context monitors share the low level raw sensor data with tracking and gesture recognition modules and provide aggregated context information that can affect the performance of the system indirectly by modifying configuration parameters of the system, such as the canonical probability of the gestures and postures, injecting posture hypothesis, modifying the interpretation agents' behavior by modifying its ontology, such as the word dictionary in the word recognition application.

Context should not be localized to a particular stage of the system (cf. Figure 1). On the other hand, the role of context should be distributed in all the modules and utilized to enhance the performance in all aspects of the system. The following figure shows a modified perspective of the role of context in gesture recognition. A revised role of context in a vision-based hand tracking and gesture recognition system utilized by a context-aware user interface is shown in Figure 70.

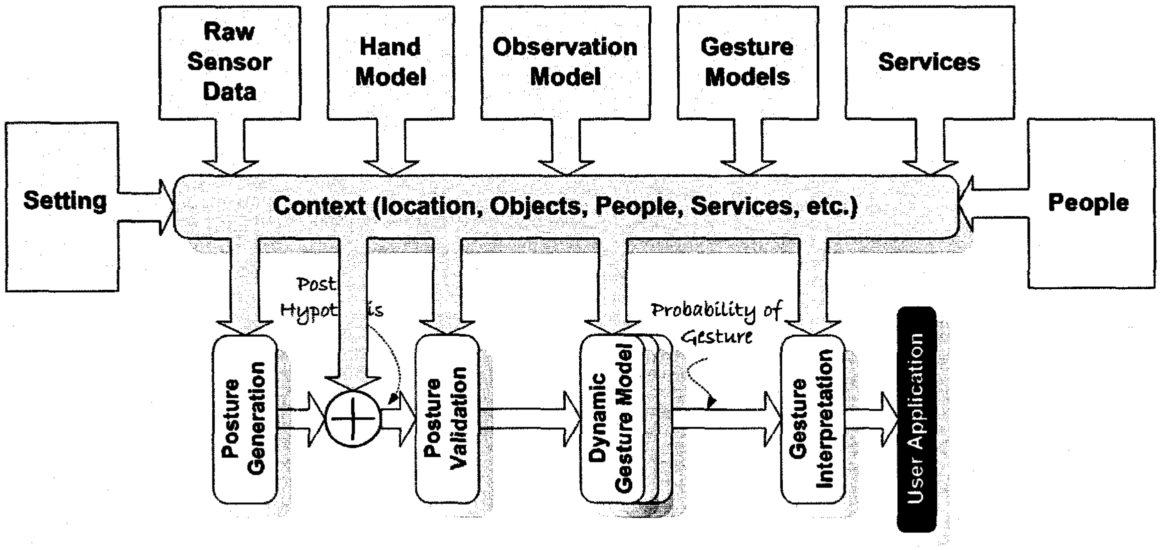


Figure 70: Role of Context in Perceptual User Interfaces

The figure shows context as a middleware layer utilized by all of the data flow pipeline stages. The aspects of context include all the valuable pieces of information to the hand tracking, gesture recognition and gesture interpretation. It can be divided into two main categories:

- Persistent context: are the fundamental inputs to the system; namely the low level sensor data (e.g. CyberGlove data, or video frames), hand model, gesture models, and observation model. The persistent context is an essential part of the system design that belongs to the positivist viewpoint of context, and is covered earlier as we explain our framework for dynamic gesture recognition.
- Volatile context: is a collection of relevant information that is emergent in nature and can not be solely designed for, thus belongs to the phenomenological viewpoint of context, but are used as an enhancement to the system in the form of a better response time, better accuracy, or smarter response to the user's gestures. The sources of this type of context are a collection of smart sensors that can recognize people, objects, and services in the vicinity of the user.

5.4. Conclusion

We presented the role of context in gesture enabled applications. We highlighted the relevant aspects of the gestures that should be part of the gesture interpretation process. We also presented an agent-based system design for gesture interpretation, where ultimately many gesture enabled services can share the interpretation agent. We demonstrated the different layers of the agent-based system design on a manual word recognition system, utilizing 347 gestures and recognizing words from a dictionary. Finally we presented our integral view of the role of context in any application and particularly to gesture enabled user interfaces. This view entails a dedicated middleware layer for context, facilitating the flow of context information throughout the different stages of the system design. We concluded that for a context-aware application to manage the uncertainty of the context information, it must be designed with dual origins of context in mind. The positivist approach is a bottom-up approach that defines the essential data structures and processing modules required for the basic functionality of the system. It provides configurable parameters for each module that can indirectly enhance the system performance or alter the system's functionality, e.g. canonical posture and gesture probability, given the context of the situation, and the posture hypothesis derived from the context. An agent paradigm is used to abstract the outputs of these modules. The phenomenological approach is used to glue these agents together in meaningful topology that satisfies the system's objectives. It utilizes the configurable parameters defined by the system modules to enhance/alter its performance.

Chapter 6

6. Conclusion and Future Work

In this dissertation we tried to conquer new frontiers in the field of gesture recognition. First, we compared the de facto standard hand tracking system, namely the data glove, to a representative of the vision-based hand tracking approach. This experiment helps initiate test bench mark comparisons between the two approaches, bearing in mind the maturity level of each approach. We recognized that the vision-based approach can provide accurate hand posture estimation, but the accuracy is affected by external factors, such as the camera viewpoint and the hand self occlusion. In our opinion, more research has to be conducted to increase the stability of the hand posture estimation provided by the vision-based approach. Collaboration of multiple cameras and the utilization of more markers are essential to make the approach survive the inevitable self-occlusion problem. More easily detected markers can improve the overall feature extraction performance, light emitting diodes, fiber optic cables, or infrared light sources comes to mind. Light sources are easily extracted by their intensity and color markers surrounding the light source can be used to identify the feature.

We started with a single camera approach and then extend the approach by fusing the output of multiple cameras. We utilize a color marker approach to make the system insensitive to noise from the surrounding environment and to reduce the restriction on the user's motion. Many state of the art research in hand tracking use marker-less hands but insist on wearing long sleeve and dark colors and the background is usually monotone white. Moreover, the hand motion and orientation is usually restricted to planar motion facing the camera.

We tried to keep the requirements for the tracking system at the minimum level, because this would demonstrate the worst case scenario. Using redundant feature will only enhance the performance of the system. We use perspective geometric invariants and inverse kinematics to produce posture hypotheses and then validate those hypotheses using probabilistic observation models. Hypothesis validation utilizes color and edge artifacts, the vicinity of the hypotheses to the targets and to the estimated postures. Posture estimation is performed using damped polynomial extrapolation, to compromise estimating the hand dynamics and estimation stability. In our opinion this is a more appropriate approach than Kalman filter estimation, because it requires minimum training, fewer parameters to tweak, and responds faster to the hand dynamics.

We developed a probabilistic observation model to validate posture hypotheses, which are projected on the image plane. The observation model utilizes hand/glove color and edges artifacts, the detected fingertip marker candidates and previous estimated postures.

In the future, we would like to enhance the glove design by including more redundant markers. We would like to use a lighter color glove because the darker (black) color glove suppresses inter-finger edge detection. We would like to place the markers on the wrist, all round the hand, allowing multiple markers to be visible in all hand orientations, which will enhance the accuracy of the hand position and orientation calculations. We would like to investigate CRC forward error correction codes to encode the palm markers, which will provide easier recognition even under partial occlusion (cf. ARtag [114], [115]). Simple color markers can be added on palm and fingers to estimate the hand articulation using inverse kinematics as explained in this dissertation. We also would like to utilize a Kalman filter and other techniques to reduce the noise of the vision-based hand posture estimation.

We discovered an efficient method for finger inverse kinematics that utilizes the error model of a simpler 2-DOF inverse kinematics problem to solve the 4-DOF of the fingers. The new method provides superior accuracy than the trivial table lookup and also does not require storage of tables of kinematics data. The method is also very efficient in terms of processing time.

We also presented Dynamic Bayesian Model for Dynamic Gesture recognition. The model relaxes some of the inherent assumptions of the HMM models that do not apply to dynamic gestures, particularly the assumption of a constant state transition matrix that controls the causality of the posture transition. The techniques for model training or parameter estimation and the process of gesture recognition were explained in detail. Briefly, training is performed from few gesture samples by generating a gesture ensemble from each gesture sample by adding random time-scale and acuteness factors. The model's parameters are then derived by integrating the probability density of the random variable over ranges where the hand posture remains unchanged. The gesture models construct a ring under the merge operation, thus allowing regrouping the gesture set in more meaningful ways. Gesture recognition is performed over a sliding sequence of postures by first classifying the postures and labeling the postures as elements from a discrete set of posture covering the domain of all possible postures, using the posture labels and models transition matrices the probability of the observed posture sequence, given the gesture hypothesis

Postures are defined in a multi-dimensional attribute space, usually comprised by position and orientation vectors and finger join angles. The posture set is mapped to a discrete set of labels in the range [1, N], where N is the total number of posture classes. Posture classification is performed by dividing the static constraints of finger joints into disjoint intervals, and using the Cartesian product of all the different joint intervals to label the posture. We derive finger curvature attributes the finger joints and showed that the curvature is more suitable for vision based tracking because it provides better estimation and it can be easily visualized. In the future works, we would like to try an <u>unsupervised clustering</u> technique to classify the posture set with the dual objective of minimizing the posture set and maximizing the discriminating effect of the gesture models.

The dynamic gesture DBN model triggers potential gestures as it unfolds with minimum latency. It also has high recognition rate even between large numbers of gestures. It was tested with a high number of gestures and demonstrated high rate of positive triggers. To reduce the number of false triggers, more attributes (e.g. hand orientation and position) and a larger number attribute classes must be used to distinguish similar gestures. This can increase the size of the model's matrices significantly. We would like to try <u>compression techniques</u> on the models data structure, since it contains a large number of sparse matrices. A high level gesture interpretation layer that takes advantage of syntactical, grammatical and other rule-based interpretation techniques can also be used to reject false triggers.

The DBN model is suitable for the recognition of intrinsic attributes of the hand posture, and the hand orientation, we would like to test it in the recognition of the hand trajectory, which may require be referenced to operator's body, e.g. torso and may be defined by the vector from the body's centroide to the hand and may be classified in circular ring sectors around the body.

The DBN model is also very sensitive to the noise in the posture data; we tried fuzzy gesture with a genetic crossover twist but the posture data was too noisy to provide an impressive results. We would like to revisit this issue to get better results.

Gestures are emergent phenomena that is usually improvised, indexical, nuance and collaborative. A limited set of gestures can be use to achieve usable human computer interaction. To achieve a flow in the gesture interaction the gestures must be recognized at rates close 100% and less than 100 millisecond latency. Pantomime type gestures can be used to pictorial images using gestures, provided that the trajectory tracking is very accurate.

Context awareness is an emerging technology in ubiquitous and mobile computing. We investigated the role of context in gesture-enabled user interface design. We demonstrated an agent-based system for gesture interpretation, taking into account all the context aspects relevant to gestures, as we learned from our survey of research in psychology. Although the system is abstract, it is general enough to accommodate any gesture application, up to our knowledge. We presented a context middle layer approach where context could play a dialectic role in vision based hand tracking. The technology is at its infancy and will develop as cheaper sensors are available and more applications utilize hand gestures. We hope our glove prototype could play such a role. We would like to develop a perceptual user interface in a tangible application.

The advantage of the vision-based tracking lies in its ability to recognize more aspects of the context and to integrate those aspects playing the dialectic role of detecting and utilizing the context in the application. Unless this advantage is utilized, the potential of the vision-based hand tracking approach will remain compromised.

References

[1] N. Liu, B. Lovell, and P. Kootsookos. "Evaluation of HMM Training Algorithms for Letter Hand Gesture Recognition," IEEE Int'l Symposium on Signal Processing and Information Technology, pp 648-651, 2003

[2] Intel® Software Products Open Source, "Intel Open Source Computer Vision Library Reference Manual," Aug. 2003 – www.intel.com/technology/computing/opencv

[3] N. D. Binh, E. Shuichi, T. Ejima, "Real-Time Hand Tracking and Gesture Recognition System," Int'l Journal on Graphics, Vision and Image Processing (GVIP 05), 2005

[4] A. Licsár, T. Szirányi, "Dynamic Training of Hand Gesture Recognition System," Int'l Conf. on Pattern Recognition (RICPR04), Vol. 4, pp 971-974, 2004

[5] A. Licsár, T. Szirányi, "Hand Gesture Recognition in Camera-Projector System," Int'l Workshop on HCI, Lecture Notes in Computer Science, Springer, Vol. LNCS 3058, pp.83-93, 2004

[6] A. Utsumi, N. Tetsutani, S. Igi, "View-based detection of 3-D interaction between hands and real objects," Proc. Int'l Conf. on Pattern Recognition, ICPR 2004, Vol.4, pp 961-964

[7] J. Lin, Y. Wu, T. S. Huang, "Articulate hand motion capturing based on a Monte Carlo nelder-mead simplex tracker," Proc. Int'l Conf. on Pattern Recognition, ICPR 2004, Vol.4, pp 975-978

[8] CyberGlove from Immersion Inc.: www.immersion.com

[9] H. Fei, I. D. Reid, "Joint Bayes Filter: A Hybrid Tracker for Non-rigid Hand Motion Recognition," Proc. European Conf. on Computer Vision, ECCV04, pp 497-508, 2004

[10] H. Fei, I. D. Reid, "Probabilistic Tracking and Recognition of Non-Rigid Hand Motion," Int'l Workshop on Analysis and Modeling of Faces and Gestures (AMFG03), pp 60-67, 2003

[11] S. Lu; D. Metaxas, D. Samaras, J. Oliensis, "Using multiple cues for hand tracking and model refinement", Proc. IEEE Conf. Computer Vision and Pattern Recognition, vol. 2, pp 443-450, 2003

[12] S. Lu; G. Huang; D. Samaras, D. Metaxas "Model-based integration of visual cues for hand tracking", Proc. IEEE Workshop on Motion and Video Computing, 2002, pp 118 – 124

[13] B. Stenger, P. R. S. Mendonc, A and R. Cipolla, "Model-Based Hand Tracking Using an Unscented Kalman Filter", British Machine Vision Conf. BMVC 2001, 63 - 72

[14] B. Stenger, A. Thayananthan, P. Torr, R. Cipolla, "Model-Based Hand Tracking Using a Hierarchical Bayesian Filter," IEEE Trans. PAMI, V. 28, 9, pp 1372- 1384, 2006

[15] K. Nirei, H. Saito, M. Mochimaru, S. Ozawa, "Human Hand Tracking from Binocular Image Sequences," Proc. Int'l Conf. IECON 1996, pp 297-302, 1996

[16] J. Lee, T. L. Kunii, "Model-Based Analysis of Hand Posture," IEEE Computer Graphics and Applications, 15 (5), pp 77-86, 1995

[17] H. Guan, J. S. Chang, L. Chen, R.S. Feris, M. Turk, "Multi-view Appearance-based 3D Hand Pose Estimation," Computer Vision and Pattern Recognition Workshop (CVPRW06), 2006, pp 154- 154

[18] M. Kolsch, M. Turk, T. Hollerer: "Vision-Based Interfaces for Mobility," In Proc. IEEE Intl. Conf. on Mobile and Ubiquitous Systems (Mobiquitous), August 2004.

[19] M. Kolsch and M. Turk: "Fast 2D Hand Tracking with Flocks of Features and Multi-Cue Integration," In Proc. IEEE Workshop on Real-Time Vision for Human-Computer Interaction (CVPR04), 2004.

[20] M. Kolsch and M. Turk: "Robust Hand Detection," In Proc. IEEE Intl. Conf. on Automatic Face and Gesture Recognition, 2004.

[21] M. Yeasin, S. Chaudhuri, "Dynamic hand gesture understanding-a new approach," Proc. Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP), V. 6, pp 3073 – 3076, 1999

[22] G. Burdea, and P. Coiffet, , "Virtual Reality Technology", Wiley Interscience, 2003

[23] S. Ahmad, "A Usable Real-Time 3D Hand Tracker", IEEE Asilomar Conf., Vol. 2, pp 1257 – 1261, 1994

[24] Y. Wu and T. S. Huang, "Capturing articulated human hand motion: A divide-and-conquer approach," Proc. of IEEE Int'l Conf. Computer Vision, pp 606–611, 1999

[25] I. Albrecht, J. Haber, H-P Seidel, " Construction and Animation of Anatomically Based Human Hand Models," Eurographics/SIGGRAPH Symposium on Computer Animation, pp 98 – 109, 2003

[26] Kenichi Kanatani, "Geometric Computation for Machine Vision", Oxford Engineering Science Series #37, Oxford Science Publications, 1993

[27] R. Mas, D. Thalmann, "A Hand Control and Automatic Grasping System for Synthetic Actors," Proc. of Eurographics 1994

[28] H. Rijpkema, M. Girard, "Computer Animation of Knowledge-Based Human Hand," SIGGRAPH 1991, pp 339-348

[29] G. Hillebrand, M. Bauer, K. Achatz, G. Klinker, "Inverse Kinematic Infrared Optical Finger Tracking," 9th Int'l Conf. on Humans and Computers (HC 2006), , 2006

[30] J Yin, A. Dhanik, D. Hsu, Ye Wang, "The Creation of a Music-Driven Digital Violinist," ACM Multimedia, ACM-MM04, 2004, pp 476-479

[31] G. ElKoura, K. Singh, "Handrix: Animating the Human Hand," Eurographics / SIGGRAPH Symposium on Computer Animation, pp 110-119, 2003

[32] C. V. Koten, A. Gray, "Bayesian Statistical Effort Prediction Models for Data-centered 4GL Software Development," Discussion Paper 2005/09. Department of Information Science, University of Otago, Dunedin, New Zealand, 2005

[33] A. Blake, M. Isard, "Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion," Springer 2000

[34] J. LaViola Jr., "A Survey of Hand Posture and Gesture Recognition Techniques and Technology," Brown University Technical Report CS-99-11, 1999

[35] V.I. Pavlovic, R. Sharma, and T.S. Huang. "Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review," IEEE Trans. PAMI, V. 19, No.7, pp 677 - 695, 1997

[36] L. Gui, J-P Thiran, N. Paragios, "Joint Object Segmentation and Behavior Classification in Image Sequences," IEEE Conf. on Computer Vision and Pattern Recognition, CVPR`07, pp 1-8, 2007

[37] D. Cremers, "Dynamical statistical shape priors for level set-based tracking,", IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), V. 26, Issue 8, pp 1262-1273, 2006

[38] A. J. Howell, K. Sage, H. Buxton, "Developing Task-Specific RBF Hand Gesture Recognition," 5th Int'l Gesture Workshop, GW 2003, pp 269-276, 2003

[39] Z. Ghahramani, "An Introduction to Hidden Markov Models and Bayesian Networks", Int'l Journal of Pattern Recognition and Artificial Intelligence 15(1), pp 9-42, 2001

[40] E. T. Jaynes (Author), G. Larry Bretthorst (Editor), "Probability Theory: The Logic of Science," Cambridge University Press

[41] D. Xu, "A Neural Network Approach for Hand Gesture Recognition in Virtual Reality Driving Training System of SPG," Int'l Conf. on Pattern Recognition, ICPR 2006, pp 519 – 522, 2006

[42] S. Ong, S. Ranganath, "Automatic Sign Language Analysis: A Survey and the Future beyond Lexical Meaning," IEEE Trans. PAMI, Vol. 27, No. 6, pp 873-891, 2005

[43] S. Mitra, T. Acharya, "Gesture Recognition: A survey, " IEEE Trans. Systems, Man and Cybernetics, Vol. 37, issue 3, pp 311 - 324, 2007

[44] F. Mahmoudi, M. Parviz, "Visual Hand Tracking Algorithms, " Geometric Modeling and Imaging--New Trends, pp 228 – 232, 2006

[45] A. Pentland, "A Computational Model of Social Signaling," Int'l Conf. on Pattern Recognition, ICPR 2006, pp 1080 – 1083

[46] F. K. H. Quek, "Toward a Vision Based Hand Gesture Interface," in Virtual Reality Software and Tech. Conf., pp 17-31, 1994

[47] D.J. Sturman and D. Zeltzer, "A Survey of Glove-Based Input," IEEE Computer Graphics and Applications, vol. 14, pp. 30-39, 1994

[48] X. Liu,K. Fujimura, "Hand Gesture Recognition using Depth Data", Proceedings of the 6th IEEE Int'l conf. on Automatic Face and Gesture Recognition (FGR'04), pp 529 – 534, 2004

[49] Wachs, J., Kartoun, U., Stern, H., Edan, Y., "Real-time hand gesture telerobotic system using fuzzy c-means clustering", Proceedings of the 5th Biannual World Automation Congress, Vol. 13, 403 – 409, 2002

[50] Wachs, J., Stern, H., Edan, Y., "Parameter search for an image processing fuzzy C-means hand gesture recognition system", Proceedings of Int'l Conf. on Image Processing, 2003, Vol. 3, 14 - 17

[51] B. Min, Y. Ho-Sub, S. Jung, Y. Yun-Mo, E. Toshiaki, "Hand Gesture Recognition Using Hidden Markov Models," IEEE Int'l conf. on Systems, Man, And Cybernetics, Vol. 5, pp 4232 – 4235, 1997

[52] Bretzner, L., Laptev, I., Lindeberg, T., "Hand Gesture Recognition using Multi-Scale Colour Features, Hierarchical Models and Particle Filtering", Proceedings of the 5th IEEE Int'l Conf. on Automatic Face and Gesture Recognition (FGR.02), 2002, PP 423 – 428

[53] K. Nickel., R. Stiefelhagen, "Pointing Gesture Recognition based on 3D-Tracking of Face, Hands and Head Orientation," Proc. of the 5th Int'l conf. on Multimodal Interfaces, pp 140-146, 2003

[54] S. Marcel, O. Bernier, J.-E. Viallet, D. Collobert, "Hand gesture recognition using input-output hidden Markov models," Proc. of the 4th IEEE Int'l conf. on Automatic Face and Gesture Recognition, pp 456 – 461, 2000

[55] Bengio Y., Frasconi P., "An Input/Output HMM Architecture", Advances in Neural Information Processing Systems, pp 427 – 434, 1995

[56] Y. Wu, J. Lin, and T. S. Huang, "Capturing natural hand articulation," Proc. of IEEE Int'l Conf. Computer Vision, volume II, pages 426–432, 2001

[57] D.N. Metaxas, Physics-Based Deformable Models – Applications to Computer Vision, Graphics and Medical Imaging, Kluwere Academic Publishers, 1998

[58] J. J. Kuch, "Vision based Hand Modeling and Gesture Recognition for Human Computer Interaction," Master's Thesis, University of Illinois at Urbana-Champaign, 1994

[59] G. Burdea, J. Zhuang, E. Roskos, D. Silver, N. Langrana, " A Portable Dextrous Master with Force Feedback," Presence, Vol. 1, No. 1, pp 18-27, 1992

[60] J. Wachs, H. Stern, Y. Edan, "Cluster Labeling and Parameter Estimation for the Automated Setup of a Hand-Gesture Recognition System," IEEE Trans. on Systems, Man, and Cybernetics, Vol. 35, No. 6, pp 932-944, 2005

[61] A. Blake and M. Isard, "Active Contours: The Application of Techniques from Graphics, Vision, Control Theory and Statistics to Visual Tracking of Shapes in Motion," Springer, 2000

[62] T. Starner, J. Weaver, A. Pentland, "Real-Time American Sign Language Recognition Using Desk and Wearable Computer Based Video," IEEE Trans. PAMI, Vol. 20, No. 12, pp 1371-1375

[63] L. Br`ethes, F. Lerasle, P. Dan`es, "Data Fusion for Visual Tracking dedicated to Human-Robot Interaction," Proc. Intl Conf. on Robotics and Automation, pp 2075-2080, 2005

[64] W. B. Langdon, R. Poli, "Foundation of Genetic Programming", Springer, 2002

[65] K. Sage, A. J. Howell, and H. Buxton, 5th Intl Gesture Workshop, GW 2003, pp 277-287

[66] A. Malima, E. Ozgur, M. Cetin, "A Fast Algorithm for Vision-based Hand Gesture Recognition for Robot Control," IEEE Conf. Signal Processing and Communications Applications, pp 1-4, 2006

[67] P. Dourish, "What We Talk about When We Talk about Context," Personal Ubiquitous Computing, Vol. 8, pp 19-30, 2004

[68] L. Suchman, "Plans and Situated Actions: The Problem of Human-Machine Communication". Cambridge: Cambridge University Press, 1987.

[69] B. Schilit, N. Adams, R. Want "Context-Aware Computing Applications", 1st International Workshop on Mobile Computing Systems and Applications. 1994. pp 85-90

[70] T. Strang, C. Linnhoff-Popien, "A context Modeling Survey," Workshop on Advanced Context Modeling, Reasoning and Management as part of UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England, September 2004

[71] T. Strang, C. Linnhoff-Popien, "Service Interoperability on Context Level in Ubiquitous Computing Environments", International Conference on Advances in Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet (SSGRR2003w), January, 2003

[72] T. Strang, C. Linnhoff-Popien , K.Frank, "CoOL: A Context Ontology Language to enable Contextual Interoperability", 4th IFIP International Conference on Distributed Applications and Interoperable Systems, , 2003, pp 236-247

[73] A. Dey, G. Abowd, D. Salber, "A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, HCI Journal, Vol. 16(2-3), 2001, 97 – 166

[74] A. Dey, "Understanding and Using Context", Personal and Ubiquitous Computing Journal, V 5(1), 2001, pp 4-7

[75] M. Weiser, "The Computer for the 21st Century", Scientific American, September 1991, pp 94-104

[76] M. Weiser, R. Gold, and J. S. Brown, "The origins of ubiquitous computing research at PARC in the late 1980s", IBM *Systems Journal* 38(4), pp 693-696

[77] M. H.Coen, "Design Principles for Intelligent Environments", Proc.15th National Conf. on Artificial Intelligence (AAAI'98), pp 547-554, 1998

[78] S. Meyer, A. Rakotonirainy, "A survey of research on context-aware homes", Proc. of the Australasian Information Security Workshop Conf. on ACSW frontiers 2003, V. 21, pp 159 - 168

[79] M. Beigl, H. W. Gellersen , and A. Schmidt, "Mediacups: experience with design and use of computer-augmented everyday artifacts", Computer Networks 35(4), pp 401-409

[80] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System architecture directions for networked sensors", In Operating Systems Review, 34(5), pp 93-104. ACM, 2000

[81] B. Warneke, M. Last, B. Liebowitz, and K. S. J. Pister, "Smart Dust: communicating with a cubic millimeter computer," Computer 34(1), pp 44-51, 2001

[82] N. B. Priyantha, A. Chakraborty, and H. Balarishnan, "The Cricket location support system", In MobiCom 2000. Proc. 6th Intl Conf. on Mobile Computing and Networking, pp. 32-43, 2000

[83] M. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggles, A. Ward, and A. Hopper, "Implementing a Sentient Computing System", Computer 34(8), pp 50-56

[84] H. Ishii, "Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms", Proc. CHI '97, 234-241

[85] S. Greenberg, and C. Fitchett, "Phidgets: easy development of physical interfaces through physical widgets", 01UIST, Proc. of the 14th ACM Symp. on User Interface Software and Technology. ACM, pp. 209-18, 2001

[86] T. Darrell, D. Demirdjian, N. Checka, and P. Felzenszwalb, "Plan-view trajectory estimation with dense stereo background models". Proc. 8th IEEE Intl. Conf. on Computer Vision. ICCV 2001, pp 628- 35

[87] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, and S. Shafer, "Multi-camera multi-person tracking for EasyLiving", In *Proceedings Third IEEE International Workshop on Visual Surveillance. IEEE Comp. Soc, Los Alamitos, CA, USA; 2000; vi+85 pp.* 3-10

[88] B. Brumitt and J.J. Cadiz, "Let there be light: examining interfaces for homes of the future", In Human Computer Interaction. INTERACT'01. IFIP TC.13 Intl Conf. on Human Computer Interaction, pp 375-82, 2001

[89] R. K. Sawyer, "Creating Conversations – Improvisation in everyday discourse", Hampton Press, 2001

[90] A. Meyer & T. A. Brown, "Statistical mechanics of voting", Physical Review Letters, 81(8), 1998, pp 1718-1721

[91] H. Garfinkel, "Studies in Ethnomethodology," Polity Press/Blackwell Publishing, , 1984.

[92] J. Lasseter, "Principles of Traditional Animation Applied to 3D Computer Animation", ACM SIGGRAPH, 1987, pp 35-44

[93] M.A.K. Halliday, "An Introduction to Functional Grammar" ; revised by Christian M.I.M. Matthiessen, London: Edward Arnold, 2004

[94] A. Kendon, "Human Gesture," Tools, Language, and Cognition in Human Evolution, K.R. Gibson and T. Ingold, eds., pp. 43-62, Cambridge Univ. Press, 1993

[95] T. Kobayashi and S. Haruyama, "Partly-Hidden Markov Model and Its Application to Gesture Recognition," Proc Int'l Conf. Acoustics, Speech and Signal Processing, vol. 4, pp. 3081-3084, 1997

[96] H. Poizner, E.S. Klima, U. Bellugi, and R.B. Livingston, "Motion Analysis of Grammatical Processes in a Visual-Gestural Language," Proc. ACM SIGGRAPH/SIGART Interdisciplinary Workshop, pp. 271-292, 1983

[97] F. Quek, "Toward a Vision-Based Hand Gesture Interface," Proc. Virtual Reality Software and Technical Conf., pp. 17-29, 1994

[98] Y. Wu, J. Lin, T. S. Huang, "Analyzing and capturing articulated hand motion in image sequences," IEEE Trans. PAMI, V. 27, 12, pp 1910 – 1922, 2005

[99] R. Bryll, R. T. Rose, F. Quek, "Agent-Based Gesture Tracking," IEEE Trans. SMC, V. 35, No. 6, pp 795 – 810, 2005

[100] F. Quek, Xin-Feng Ma, R. Bryll, " A parallel algorithm for dynamic gesture tracking," Int'l Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-time Systems, pp 64 – 69, 1999

[101] M. Csikszentmihalyi "Flow: The psychology of optimal experience", New York: Harper and Collins, 1990

[102] S. Kirkpatrick and C. D. Gelatt and M. P. Vecchi, Optimization by Simulated Annealing, Science, Vol 220, Number 4598, pages 671-680, 1983. http://citeseer.ist.psu.edu/kirkpatrick83optimization.html.

[103] V. Cerny, A thermo dynamical approach to the traveling salesman problem: an efficient simulation algorithm. Journal of Optimization Theory and Applications, 45:41-51, 1985

[104] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. "Equations of State Calculations by Fast Computing Machines". Journal of Chemical Physics, 21(6):1087-1092, 1953.

[105] R. Feris, M. Turk, R. Raskar, K. Tan, and G. Ohashi, "Exploiting depth discontinuities for vision-based finger spelling recognition," in IEEE Workshop on Real-time Vision for HCI (in conjunction with CVPR'04), 2004

[106] Qi-Gang Gao, A.K.C. Wong, "Curve Detection Based on Perceptual Organization," Pattern Recognition, Vol. 26, No. 7, pp 1039 – 1046, 1993

[107] Dourish P, Button G , "On techno-methodology: foundational relationships between ethnomethodology and system design," HCI 13(4):3 95 – 432, 1998

[108] P. Stone and M. Veloso, "Multi-agent systems: A survey from a machine learning perspective," *Auton. Robots*, vol. 8, no. 3, pp. 345–383, 2000

[109] M. Woolridge and N. R. Jennings, "Intelligent agents: Theory and practice," *Knowl. Eng. Rev.*, vol. 10, no. 2, pp. 115–152, 1995

[110] N. R. Jennings, K. Sycara, and M. Woolridge, "A roadmap of agent research and development," *Auton. Agents Multi-Agent Syst.*, vol. 1, no. 1, pp. 7–38, 1998

[111] G. Weiss, "Multi-agent Systems: A Modern Approach to Distributed Artificial Intelligence", MIT Press, 1999

[112] W. Brenner, R. Zarnekow, H. Wittig, "Intelligent Software Agents: Foundations and Applications," Springer-Verlag, 1998

[113] M. Woolridge, "Intelligent agents," in Multi-agent Systems: A Modern Approach to Distributed Artificial Intelligence, G. Weiss, Ed., MIT Press, 1999, pp. 27–77

[114] Artag Fiducial Technology, www.artag.net

[115] M. Fiala, "ARTag Revision 1. A Fiducial Marker System Using Digital Techniques", NRC/ERB-1117, November 2004, www.iit-iti.nrc-cnrc.gc.ca/iit-publications-iti/docs/NRC-47419.pdf

[116] K. Kahol, P. Tripathi, S. Panchanathan, "Computational analysis of mannerism gestures," Proc. Intl. Conf. Pattern Recognition, ICPR`04, V. 3, pp 946 – 949, 2004

[117] S. Zhong and J. Ghosh. "HMMs and coupled HMMs for multi-channel EEG classification." Proc. IEEE Intl. Joint Conf. on Neural Networks, pp 1154 - 1159, 2002