ABSTRACT

Title of dissertation: Visual Tracking of Human Hand and Head Movements and Its Applications

Afshin Sepehri
Doctor of Philosophy, 2007

Dissertation directed by: Professor Rama Chellappa
Department of Electrical Engineering

Tracking of human body movements is an important problem in computer vision with applications in visual surveillance and human-computer interaction.

Tracking of a single hand moving in space is addressed and a set of applications in human-computer interaction are presented. In this approach, a disparity map and motion fields extracted from a stereo camera set are modelled using a robust estimation method. Then, the absolute position and orientation of the hand in space are estimated and the central region of the hand is tracked over time. Virtual drawing in space, a virtual marble game, and 3D object construction are shown as the applications of the single hand tracking.

Algorithms are presented for tracking the hands and head of a person or several interacting people viewed by a set of cameras in 3D. The problem is first defined as a general multiple object tracking problem in a multiple sensor environment and a two layered solution is proposed. The proposed solution includes a low-level particle filtering layer to track individual targets in parallel, and a finite state machine to analyze the interactions between the targets and apply application specific heuristics.

A set of activity recognition experiments in visual surveillance show the usefulness of the system. The recognized activities involve interactions between the hands and head of people and objects. A color analysis scheme and a technique for combining information from different cameras are presented. They are used to detect carried objects and exchanges between the hands.

Visual Tracking of Human Hand and Head Movements and Its Applications

by

Afshin Sepehri

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2007

Advisory Committee:
Professor Rama Chellappa, Chair/Advisor
Professor Larry S. Davis, Co-Advisor
Dr. Yaser Yacoob, Co-Advisor
Professor Min Wu
Professor Nuno C. Martins

Preface

Since about 40 years ago, when a project entitled "Solve the vision problem" was defined as a summer project in MIT, and since scientists and scholars such as Prof. Azriel Rosenfeld wrote several books and papers to introduce this field, there have been many significant advances and achievements. Many vision-based systems are now working smoothly and reliably in industry. Surveillance cameras being used on roads to control traffic, inspection and quality control systems in the assembly lines, and even fun applications in the area of computer games and animation production, are just a few examples. In all these systems, the idea of replacing the human eyes and brain with a set of cameras and a computer-based system is achieved successfully. The expectation is generally to obtain a system which can perform the job of a human being satisfactorily; however there have been several cases in which the computer-based system does a substantially better job than a human. This is due to higher resolution of cameras, and the increasing computational power of computer systems. Also, a human being is susceptible to fatigue and physical and emotional degradations which a computer system is not.

However, the question of how to build a comprehensive vision system in the general sense is still open. While the computational power of computers still grows rapidly, there has been little success in modelling the core vision capabilities of humans. Therefore, creating artificial systems which can work reliably under complicated and unpredictable real world conditions is still far from being realized. Computer vision is an exceptionally challenging field. Thousands of papers and dis-

sertations have been written to address its problems and challenges. I hope I have written one of these thousands!

Dedication

To the three ladies in my life:

My Mother

My Wife

My Daughter

Acknowledgments

First and foremost, I would like to thank my advisor, Prof. Rama Chellappa and my co-advisors Prof. Larry Davis and Dr. Yaser Yacoob for their support and guidance during my studies. They introduced me to the fascinating field of computer vision and gave me the opportunity to work on several interesting problems in this field. I was especially blessed to have the opportunity of learning from these three scholars simultaneously during my research and study. I really appreciate the time and attention they gave my work which was sometime beyond my expectation. I would like to thank my committee members, Prof. Wu and Prof. Martins for their time and valuable feedbacks. I would like to thank the faculty and staff of the department of electrical and computer engineering, computer science, and institute for advanced computer studies for their constant support and help.

During my studies, I had the opportunity of having a lot of great scientific discussions with some of my friends in the computer vision laboratory and beyond. I thank them all. I am especially thankful to my friend, Dr. Mehdi Kalantari who assisted me in better understanding of some of the concepts in the area of signal processing and random processes. I am also thankful to my friends, Shabnam Shafiee, Melody Djam, Amirali Sharifi and Kamiar Kordari who helped me in collecting the required video clips for this dissertation.

My sincere gratitude goes to my family for all they gave me during my life; my father who taught me a lot and encouraged me to pursue my studies to the higher levels, my mother who did the outmost sacrifices for me, my brothers who were very

reliable and consistently supportive to me, my wife Farangis, who gave me her love, patience and support and my little daughter Nusha, whose birth added so much joy to my life and extended the duration of my study!

My social life in Maryland was also quite a pleasant experience. I found numerous friends who helped me whenever I needed them. I am afraid I cannot name them here as they are too many. I would like to thank them all and wish them success wherever they are.

Table of Contents

List of Figures

Chapter 1

Introduction

One of the major subjects of research in the field of computer vision is understanding the human movement. Extensive work has been performed on tracking the entire body to estimate motion paths or body gestures. This is especially useful in the area of visual surveillance to recognize activities. Recent work in this area involve [1–9]. On the other hand, much research has focused on a single or a few body limbs. The most important limbs are the hands and head. Most work on visual analysis of the head focuses on modeling the face and its parts for *face recognition*, but head gaze tracking is also an important problem that has received considerable attention. For a comprehensive overview of the major works in this area, refer to [10].

The hand is the limb which performs most of people's physical interactions with the world. These interactions could be cooperative and be used in the field of *human computer interaction* (HCI) or non-cooperative, which is the subject of *visual surveillance*. In the HCI realm, a person or several people move their hands in space in a controlled way or create different gestures with their hands to communicate commands to a computer. They can even interact with virtual devices to control a computer system. In the visual surveillance area, the hand interacts with other people and physical objects. In this dissertation, we address some of the problems

in the area of hand tracking and demonstrate a few applications in both the areas of human computer interaction and visual surveillance.

This dissertation is mainly divided into two parts. In the first part which is covered in chapter 2, a single hand is tracked while being viewed by a *stereo camera*. The *disparity map* and *motion fields* of the acquired images are estimated and modelled to enable tracking of the hand as a region in space. Three applications in the area of HCI are introduced and several experiments show the effectiveness of the method. The main contribution in this chapter is in accurate estimation of the position and orientation of the hand in space and also the novel applications.

In the second part, which is contained in chapter 3, both the hands and the head of a person or several people are tracked in space. The people are viewed by multiple cameras distributed around the scene. Activity recognition is presented as an application of the proposed system in the area of visual surveillance. The contribution of this part is in the proposed two-level tracking scheme which involves *particle filters* and a *finite state machine*. Also, the likelihood estimation approach performed by re-projection of the observations from the image space to the 3D space and analyzing the 3D candidate points is novel. In addition, a new technique for combining the results of the color analysis of different images is proposed, which is based on the quality of the hand view in each image. The result of the color analysis is deployed in detecting carried objects as well as object exchanges.

Chapter 2

Visual Tracking of a Single Hand

2.1 Introduction

The human hand serves a dual purpose as a communication and manipulation device. This chapter is focused on employing a single hand as an interface device to a computer. It presents applications that require accurate estimation of the position and orientation of the hand in space with respect to a camera system. We describe a real-time stereo system to estimate the position and orientation of the hand in the camera and world coordinate systems and also track its spatial trajectory over time. We also demonstrate the utility of our system in virtual and real spaces using three applications:

1. A virtual drawing application, in which a user can write letters or draw on a virtual plane in space.

2. A 3D model construction application, in which the user runs his hand along the edges of a physical polyhedral object, and the system constructs a 3D model of that object, and

3. A 3D virtual marble game, in which the user controls the inclination of a virtual plane through hand motions to manipulate the movement of a ball through a maze.

The first two applications demonstrate the accuracy of the position and orientation estimation algorithms, while the third demonstrates the real time capabilities of our algorithms.

2.1.1 Previous Work

Modeling the human body as an articulated object has been extensively studied during the last decade (For a review, see [11]). Hand modeling is typically done with a device such as an instrumented glove to measure direct parameters of the hand [12] or the placement of colored-markers on a hand to simplify visual tracking [13]. These devices or markers may be reasonable for highly specialized application domains such as surgery in a virtual reality environment, but can be impractical in many consumer applications. The limitations of employing these auxiliary devices has motivated research on *bare-hand* approaches. This research has been focused on either static hand gestures (i.e. postures) or dynamic characteristics of gesture.

In the former case, a still image is analyzed with the goal of finding some predetermined parameters of hands (e.g. palm position and orientation, finger joint angles). A variety of models has been utilized for that purpose. Images of hands, geometric moments, contours, silhouettes, and 3D hand skeleton models are a few examples [13–17].

One of the earliest works on modeling a bare hand was performed by Rehg and Kanade [18] where they model the hand and fingers with a set of lines and points. They start tracking from a predefined state and performed a local search at each

frame for the new configuration. They used a modified Gauss-Newton algorithm to minimize the error. As an application, they presented a 3D graphical mouse. In another work [19], they took into consideration the problem of self-occlusion, which usually occurs with the fingers. They addressed the issue by defining visibility order and employing an occlusion graph. They registered overlapping templates using a window function to block the occluded templates.

Kuch and Huang in [16] defined a 26 degree of freedom (DOF) hand model and by considering a set of static and dynamic constraints reduced it to 15 DOF. A generic model was calibrated using three specific views and joint length and joint angles were selected manually. The system Started from a predefined orientation in a solid background, rendered a model at each frame, XORed 2D model projection with the real image and locally perturbed the model to minimize error.

Lee and Kunii in [13] employed torque minimization to estimate the hand parameters and used inverse kinematics to calculate joint angles. They assumed zero hollowness (planarity) of the palm and the position of the characteristic points were measured through a color-coded glove using two cameras. They defined two types of driving forces: models internal constraints and the external forces derived from images and concluded that the posture of the whole hand could be determined by the position of seven characteristic points, including 5 fingertips.

In [20], Heap and Dogg estimates 6 DOF hand position and orientation (limited) as well as deformation using a 3D version of the Point Distribution Model (PDM) and a surface mesh model. They used a set of training images with 3D landmark points captured semi-automatically and a reasonable initial guess to com-

pute the twelve model parameters including translation, rotation , scale, and five significant deformation parameters. An orthographic projection was used to project 3D mesh vertices to 2D image and model parameters were updated by finding the local movement for individual landmarks and statistical voting.

Delamarre and Faugeras [21] used a sequence of stereo images to estimate the pose of the hand. They proposed a 3D articulated model of the hand and tracked the forces that would attract the model.

Athitsos, Rosales, and Sclaroff took database-oriented approach to classify hand gestures [22–24]. Training data was obtained using a CyberGlove which monitors the angular motions of the palm and fingers [22] or was produced using synthetic hand views [23]. Different functions such as chamfer distance were used for likelihood measurement.

2.1.2 Hand Tracking System Overview

Figure 2.1 shows the block diagram of the system; Its main steps are as follows:

1. Images are grabbed from a stereo camera with a baseline comparable to the distance between the human eyes. Figure 2.2 shows a sample pair of input images. Since in some applications we need to give real-time feedback to the user, image acquisition should be performed at a reasonably high rate to provide an interactive system. Our implementation for the sample drawing application (explained in section 2.7) works up to 12 frames per second on an Intel 3.2GHz processor. Input images are rectified to make the disparity map

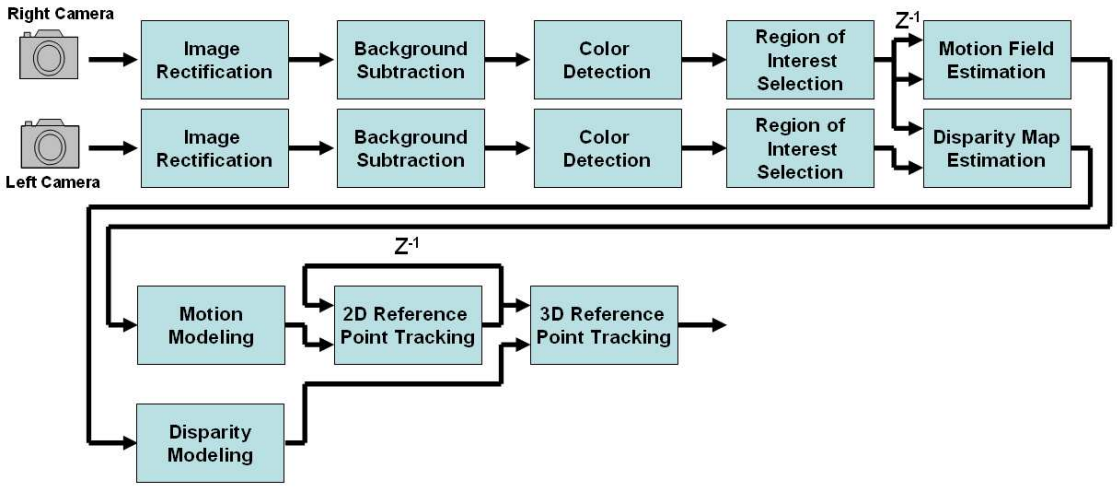
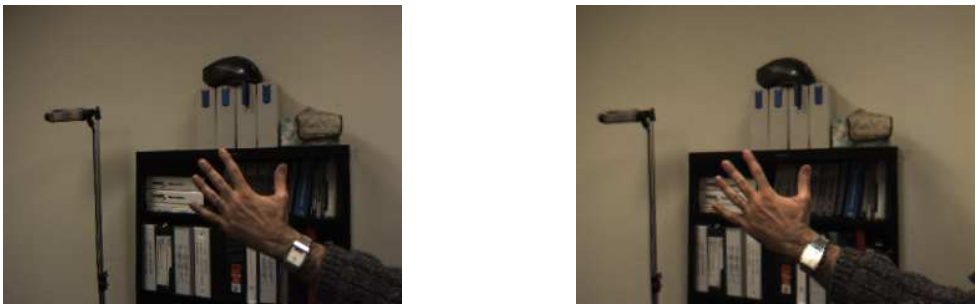


Figure 2.1: Block diagram of the system.

estimation faster.



(a) (b)

Figure 2.2: Sample stereo input images: (a) Left image, (b) Right image.

2. Background subtraction and skin color detection are employed to segment the hand. Also, for reliable tracking, the fingers and the arm are removed from the hand area so only the central region of the hand (i.e. palm, back of the hand) remains. This is discussed in section 2.2.

3. A disparity map is estimated from the two images taken at each time instant using a parametric planar model to cope with the nearly textureless surface of the hand. Section 2.3 discusses the details of this process.

4. A monocular motion field is estimated from two consecutive frames. It is modeled similarly to the disparity map. Parameters of the motion model are then adjusted to comply with the disparity model. The motion field is used for tracking selected points throughout a sequence. Section 2.4 addresses the steps.

5. At each time instant, the X, Y and Z coordinates of the position and the orientation angles *yaw*, *pitch*, and *roll* are calculated for a coordinate frame attached to the palm. The 3D plane parameters are calculated from the disparity plane as discussed in section 2.5.

6. For tracking the hand over time, a set of 2D image points are extracted from the images of one of the two cameras (e.g. left) and its motion model. Then using disparity models at different times, the points are mapped to the 3D world to provide the trajectory of the hand in space, as explained in section 2.6.

Based on the application, some extra steps may be employed. The following sections discuss the details of the process and the applications of the system are presented.

2.2 Region of Interest Segmentation

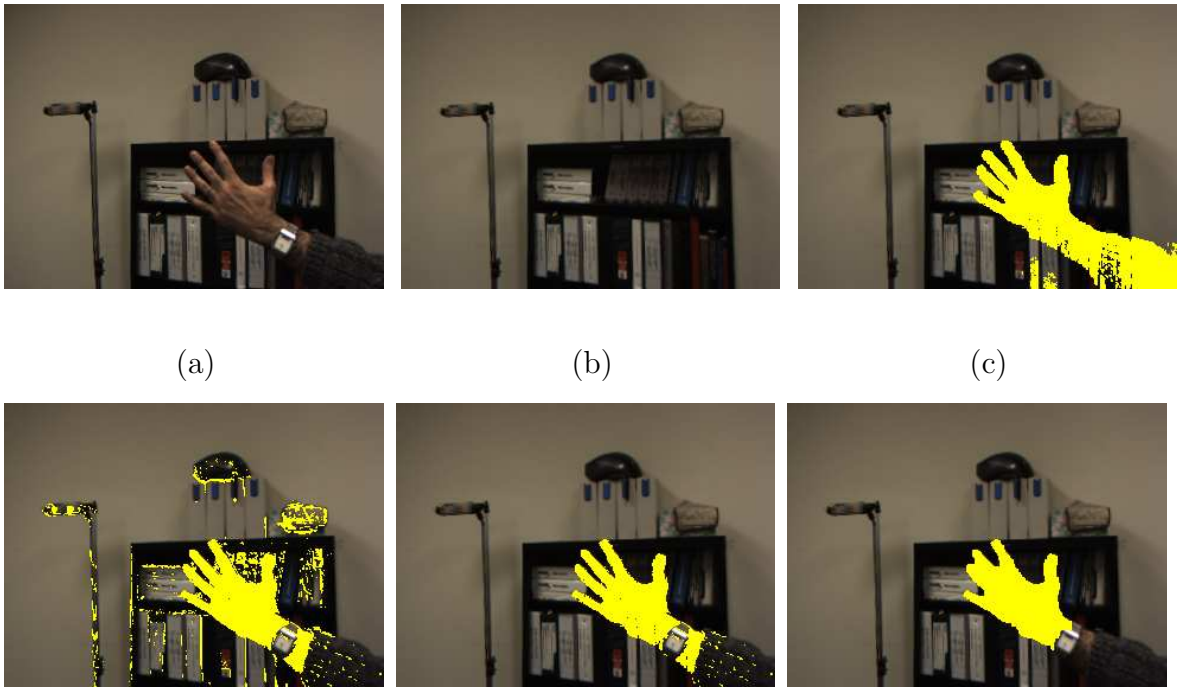
The central region of the hand (i.e. palm or back of the hand depending on the user's preference) is modeled and tracked in 3D. That region is segmented in two steps: Segmenting the entire hand from the image, and then selecting the central region from the segmented hand region. The following two subsections discuss these steps.

2.2.1 Hand Region Segmentation

Segmenting the hand from the image is performed by removing the background and moving objects other than the hand. Two cues are used:

- *Motion Cues* including background subtraction and motion-less region subtraction.
- *Color cues* which take advantage of the fact that human skin color is localized in the color space.

We use fusion of color and background subtraction to extract the hand with the color analysis applied to the results of background subtraction. Figures 2.3(b) and 2.3(c) show the background and foreground images of a sample input image 2.3(a) and figures 2.3(d) and 2.3(e) show the output of the color detector without and with the background subtraction module respectively. Background subtraction is simply implemented using a unimodal background model, followed by color skin detection and finally a *flood-fill* step. Figure 2.3(f) shows the final hand region after



(a) (b) (c)

(d) (e) (f)

Figure 2.3: Hand region segmentation: (a) Input image, (b) Background image, (c) Foreground image, (d) Color detector output without background subtraction , (e) Color detector output with background subtraction. (f) Final segmented hand region

flood-fill filtering.

2.2.1.1 Background Subtraction Module

There are many challenges in estimating and maintaining a background model including gradual or sudden illumination changes, camera oscillations and high frequency background objects. Different methods have been proposed to cope with these challenges: Median or Average Filtering [25, 26], Running Gaussian Average [27], Mixture of Gaussians [28, 29], Kernel Density Estimation [30], Mean-Shift based Estimation [31], and Eignbackgrounds [32].

However, in our case since we have an extra filtering stage which relies on the skin color, some inaccuracy and noise in the background subtraction process can be tolerated. Therefore, the background image is stored once in the beginning and a simple difference operator picks the foreground pixels.

In the latter case a temporal median filter can be employed. In case of dealing with a moving camera, a camera stabilization step precedes the above process. Motion-less region subtraction is based on the assumption that the object of interest has the dominant motion in the image. This cue can be only used in a filtering manner to eliminate the spurious regions as it is useless in case of a still hand.

2.2.1.2 Color Detection Module

It is well known that human skin color is localized in color space. In [33], it is shown that, the distribution of skin color tones is more localized in *HSL* color space than *RGB*. So, as a first step in our implementation, we convert pixel values from the RGB domain to the HSL domain using

$$\begin{aligned} H &= \begin{cases} \theta & G \geq B \\ 2\pi - \theta & G < B \end{cases} \\ S &= 1 - \tfrac{3\min(R,G,B)}{R+G+B} \\ L &= \tfrac{1}{\sqrt{3}}(R+G+B) \end{aligned} \tag{2.1}$$

where

$$\theta = \arccos\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right\}$$

The next step is to devise a skin model to discriminate between skin and non-skin pixels. There are different models used including parametric and nonparametric skin distribution models as well as explicit defining skin color region. We choose the explicit definition of the skin region in color space due to its speed. A survey on different modeling methods can be found in [34].

We divide the process into two steps. In the first step, a superset of the real skin area is selected by limiting the hue component of the color. The selected range should be large enough to tolerate the lighting conditions and variance of human skin colors. Hence we allow pixels in the range of $1 < hue < 25$ to be included. Even though the outcome covers skin area fairly good, it also contains some spurious pixels. If we narrow down the hue range, we might lose some legitimate skin pixels, so we delay additional filtering to the next step.

In the next step, candidate pixels are analyzed one by one using a neural network already trained with some sample skin colors to rule out spurious pixels. The training set is collected from two series of pixels in some sample sequences. The first series contain skin points and the second one contain non-skin points with similar color which the neural network failed to distinguish. The trained network covers a volume in HSL space consisting of a set of spheres each supported by a single neuron. For more details see [33].

2.2.2 Palm Region Segmentation

To extract the palm from the segmented hand region, we rely on the observation that the area of the palm is usually the widest part of the hand with the exception of some of the upper areas of the arm. Also, due to the presence of the fingers, the number of curvature maxima in the neighborhood of the palm is more than the arm areas. These facts allow us to model the area of the palm as a union of a set of intersecting circles.

The following summarizes the estimation process:

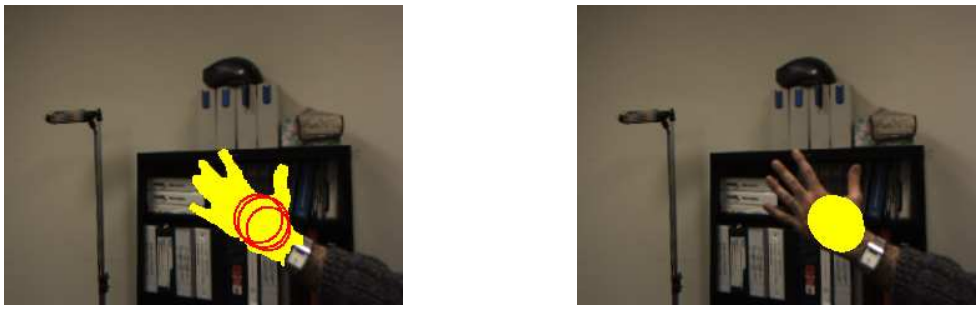
1. Segment the area of the hand as explained in section 2.2.1.

2. Find the largest interior circle (LIC) of the segmented area using the distance transform. This circle is likely to be located on the palm. However to avoid circles in the area of the arm, we find the center of gravity of the curvature maxima of the hand contour and consider only those circles that contain this point. Since the fingers create more curvature maxima than the smooth edges of the straight arm, this tends to place the center point on the palm.

3. Find other large interior circles with a radius larger than a given threshold (e.g. 0.8 of the radius of the LIC). The fingers inherently will not belong to a circle with such a radius even if a few of them are joined; To avoid including circles on the arm, we discard circles that do not intersect the LIC.

4. Compute the union of the area of all the obtained circles and consider it as the estimated area of the palm. We do not expect this area to cover the palm

perfectly. Also, the largest interior circles in the two images may not exactly correspond to the same actual hand region. Nevertheless, they will have a high percentage of overlap. Figure 2.4(a) shows the large interior circles for the image of the left camera shown in figure 2.2 and figure 2.4(b) shows the final region of interest as the union of the circles.



(a) (b)

Figure 2.4: Segmented palm regions and largest interior circles of input images of Figure 2.2: (a) Largest interior circles, (b) Final region of interest.

An alternative approach for segmenting the central region of the hand is modeling it with a square as discussed in [35].

2.3 Parametric Disparity Map Estimation

To reconstruct the position of the hand in 3D, we estimate the disparity map from a stereo. There are different sources of noise in the disparity estimation process. The two cameras usually have different levels of brightness, white balance, and contrast which makes the matching process challenging. Also, the low texture of the hand adds to this problem. The rectification process also causes some deviations

in the pixel values. To cope with noise issues, we perform this estimation in two distinct steps. First, we estimate the disparity map using a sample stereo method, and second, we model the estimated disparity map as a parametric plane to eliminate outliers. The following subsections discuss these two steps.

2.3.1 Disparity Map Estimation

There are many different methods to estimate the disparity map which can be categorized to some general approaches. Local methods, global optimization methods, dynamic programming, and cooperative algorithms are main categories [36]. We find the disparity map using a correlation-based method where the conjugate point pairs are detected. We identify and remove some incorrectly matched points by relying on the *uniqueness* of the matching and *left to right consistency* of the corresponding points. Uniqueness means each point in the left image should match one and only one point in the right image and vice versa. Consistency of the corresponding points means that if a point p_r in the right image is the best match to point p_l in the left image, point p_l should also be the best match for p_r. Applying these filtering criteria, we find a sparse disparity map. Note that the hand's low texture results in significant mismatching. Moreover, after all the filtering steps, we may still get a number of pixels with their disparity wrongly estimated. To overcome these problems, we parametrically model the disparity as explained next. Figure 2.5 shows the estimated disparity map for the sample image pair after the filtering steps.

Figure 2.5: Disparity Map of the Pair of Images in Figure 2.2.

2.3.2 Disparity Map Modeling

We model the palm as a 3D plane

$$Z = C_1X + C_2Y + C_3 = C_1(\frac{x}{f}Z) + C_2(\frac{y}{f}Z) + C_3 \tag{2.2}$$

where $P(X, Y, Z)$ is a point on the plane and $p(x, y, f)$ is the image of point P on the image plane with f denoting the focal length of the camera. Since Z is inversely proportional to the disparity value d (i.e. $Z = \frac{\alpha}{d}$ for some value α)

$$d = \frac{\alpha}{C_3} + (-\frac{C_1\alpha}{fC_3})x + (-\frac{C_2\alpha}{fC_3})y = c_1x + c_2y + c_3 \tag{2.3}$$

which means that points (x, y, d) obtained from the disparity map should also lie on a plane.

An important issue is that we need a set of disparity values distributed uniformly over the area of interest. This is due to the need of giving equal opportunity to both high-textured and low-textured areas to participate in planar fitting. So we pick up points from a disparity map through a uniform grid or a uniform random sampler.

To cope with outliers, we employ robust estimation to find the parameters of

the planar model. M-estimation [37] is a robust method of estimating the regression plane which works well in the presence of significant outliers. Considering the plane model

$$d_i = c_1 x_i + c_2 y_i + c_3 + e_i = \mathbf{x}_i^T \mathbf{c} + e_i \tag{2.4}$$

with $\mathbf{x}_i = (x_i, y_i, 1)^T$ and $\mathbf{c} = (c_1, c_2, c_3)^T$, the general M-estimator which corresponds to the *maximum-likelihood estimator* [37], minimizes the objective function

$$\sum_{i=1}^{n} \rho(e_i) = \sum_{i=1}^{n} \rho(d_i - \mathbf{x}_i^T \mathbf{c}) \tag{2.5}$$

where n is the number of points and ρ is the *influence function* [38].

Let $\psi = \rho'$ be the derivative of ρ. To minimize (2.5), we need to solve the system of three equations

$$\sum_{i=1}^{n} \psi(d_i - \mathbf{x}_i^T \mathbf{c}) \mathbf{x}_i^T = \mathbf{0} \tag{2.6}$$

Defining the weight coefficients $w_i = \psi(e_i)/e_i$, the estimating equations may be rewritten as

$$\sum_{i=1}^{n} w_i (d_i - \mathbf{x}_i^T \mathbf{c}) \mathbf{x}_i^T = \mathbf{0} \tag{2.7}$$

The solution $\mathbf{c}$ to (2.7) can be found using the iteratively reweighted least-squares, IRLS as follows [38]:

1. Select initial estimates $\mathbf{c}^{(0)}$ such as the least-square estimates.

2. At each iteration t, calculate residuals $e_i^{(t-1)}$ and associated weights $w_i^{(t-1)}$ from the previous iteration.

3. Solve for the new weighted-least-squares estimates

$$\mathbf{c}^{(t)} = [\mathbf{X}^T \mathbf{W}^{(t-1)} \mathbf{X}]^{-1} \mathbf{X}^T \mathbf{W}^{(t-1)} \mathbf{d} \tag{2.8}$$

where $\mathbf{X}$ is the matrix of points with $\mathbf{x}_i^T$ as the ith row and $\mathbf{W}^{(t-1)} = diag\{w_i^{(t-1)}\}$ is the weight matrix.

For fitting the 3D plane to our disparity data, we choose the *Geman-McClure* function for ρ [39]

$$\rho(x, \sigma) = \frac{x^2}{\sigma + x^2} \tag{2.9}$$

Since this function has a differentiable ψ-function, it provides a more gradual transition between inliers and outliers than some other influence functions [40].

To achieve fast convergence as well as to avoid local minima, we initialize weights $w_i^{(0)}$ with values proportional to the *confidence* of each point in the disparity calculation process. This confidence can be defined as the reciprocal of the sum of the differences of the pixel values in the correlation windows.

2.4 Parametric Motion Field Estimation

Calculating the disparity map and modeling it at each frame enables us to estimate the hand plane in space instantaneously; however, it does not provide a one to one mapping of the points on the planes in consecutive frames, which is required for tracking. Motion analysis is employed to recover this information.

Computing the motion field between two images taken in two consecutive time instants, corresponding points between the two images can be found. The same process can be performed for each one of the two cameras. However, due to the low-textured nature of the hand, outliers may appear in the same way as disparity

map. To remove these outliers, we model the motion elements as discussed in section 2.4.2.

2.4.1 Motion Field Estimation

To estimate the motion field, we exploit an optical flow estimation method. Some of the common methods are Horn-Schunck [41], Lucas-Kanade [42], and block matching method [43]. Through our experiments, we found block matching to be more robust and less noisy than the other two. However, this could change with lighting conditions, size of the motion, and parameters used in each method (e.g. block size, search space and etc.). Also to shrink the search space, thereby reducing spurious results, the hand region, segmented as explained in section 2.2.1, is clipped as a new image and used as an input image to block matching algorithm. In this way, we can shorten the size of the search yet keep it applicable to large motions.

2.4.2 Motion Field Modeling

The motion field is modeled using a similar approach to that used for disparity modeling. Let π be a moving plane in space with translational velocity $\mathbf{t}$ and angular velocity ω. It is well known [44] that components of the motion field $\mathbf{v} = (u, v)^T$ can be computed as

$$\begin{aligned} u &= \tfrac{1}{fd}(a_1x^2 + a_2xy + a_3fx + a_4fy + a_5f^2) \\ v &= \tfrac{1}{fd}(a_1xy + a_2y^2 + a_6fy + a_7fx + a_8f^2) \end{aligned} \tag{2.10}$$

where f is the focal length, d is the distance between π and the origin (the center of projection) and

$$a_i = g_i(\mathbf{t}, \omega, d, \mathbf{n}) \qquad 1 \leq i \leq 8$$

where $g_i(.)$ is a known function and the unit vector normal to π denoted as $\mathbf{n}$.

Defining the new coefficients b_i as

$$\begin{aligned} b_1 &= \tfrac{a_1}{fd} \qquad b_2 = \tfrac{a_2}{fd} \qquad b_3 = \tfrac{a_3}{d} \qquad b_4 = \tfrac{a_4}{d} \\ b_5 &= \tfrac{a_5 f}{d} \qquad b_6 = \tfrac{a_6}{d} \qquad b_7 = \tfrac{a_7}{d} \qquad b_8 = \tfrac{a_8 f}{d} \end{aligned} \tag{2.11}$$

equation (2.10) can be rewritten as

$$\begin{aligned} u &= b_1 x^2 + b_2 xy + b_3 x + b_4 y + b_5 \\ v &= b_1 xy + b_2 y^2 + b_6 y + b_7 x + b_8 \end{aligned} \tag{2.12}$$

If we define a new matrix $\mathbf{X}$ and a new vector $\mathbf{b}$ as

$$\begin{aligned} \mathbf{X} &= \begin{bmatrix} x^2 & xy & x & y & 1 & 0 & 0 & 0 \\ xy & y^2 & 0 & 0 & 0 & y & x & 1 \end{bmatrix} \\ \mathbf{b} &= (b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8)^T \end{aligned} \tag{2.13}$$

equation (2.12) can be rewritten as

$$\mathbf{v} = \mathbf{X}\mathbf{b} \tag{2.14}$$

Now, having a set of n points $p_i = (x_i, y_i)$ and their calculated motion vectors $\mathbf{v}_i = (u_i, v_i)^T$, we can compute $\mathbf{X}_i$s thereafter defining the new vector $\mathbf{v}_{all}$ and the new matrix $\mathbf{X}_{all}$ as a combination of $\mathbf{v}_i$s and $\mathbf{X}_i$s respectively:

$$\begin{aligned} \mathbf{X}_{all} &= \begin{bmatrix} \mathbf{X}_1^T & \mathbf{X}_2^T & ... & \mathbf{X}_n^T \end{bmatrix}^T \\ \mathbf{v}_{all} &= \begin{bmatrix} \mathbf{v}_1^T & \mathbf{v}_2^T & ... & \mathbf{v}_n^T \end{bmatrix}^T \end{aligned} \tag{2.15}$$

Equation (2.14) can be now generalized as:

$$\mathbf{v}_{all} = \mathbf{X}_{all}\mathbf{b} \tag{2.16}$$

Using M-Estimation in a similar way as in section 2.3.2, we can find the coefficient vector $\mathbf{b}$.

2.4.3 Motion Field Adjustment based on the Disparity

If we define image I as a function of spatial variables x and y and temporal integer variable t, the motion field in a stereo system can be written as:

$$\begin{aligned} I_l(x, y, t) &= I_l(x + u_l, y + v_l, t + 1) \\ I_r(x, y, t) &= I_r(x + u_r, y + v_r, t + 1) \end{aligned} \tag{2.17}$$

where indices l and r distinguish left and right cameras.

Meanwhile from stereo constraints in the rectified image pairs we know:

$$\begin{aligned} I_l(x, y, t) &= I_r(x - d, y, t) \\ I_l(x + u_l, y + v_l, t + 1) &= I_r(x + u_l - d', y + v_l, t + 1) \end{aligned} \tag{2.18}$$

with d and d' showing disparity values at times t and $t + 1$ respectively.

From (2.17) and (2.18) we can deduce:

$$u_l + d - d' = u_r \qquad v_l = v_r \tag{2.19}$$

The parameters of the motion model (2.10) were estimated for each camera individually as described in section 2.4.2. Due to mismatching and inherent deviation of the palm from a plane, the conditions in (2.19) are not exactly satisfied.

We modify the motion vectors to satisfy (2.19) to the best prior to calculating the motion coefficients as follows:

We select n sample points p_{il}^t from the region of interest on the left image at time t, find conjugate points p_{ir}^t on the right image using the disparity model and the corresponding points p_{il}^{t+1} and p_{ir}^{t+1} at time $t+1$ for the left and right images respectively using the modeled motion fields. Then, we compute a new set of points q_{ir}^{t+1} using p_{il}^{t+1} and the disparity model at time $t+1$. Now, points p_{ir}^{t+1} are replaced by a weighted average of p_{ir}^{t+1} and q_{ir}^{t+1} based on their fitness as measured by window intensity matching. We then repeat all the measurements exchanging the roles of the left and right cameras, and continue until points reach stable locations.

Using the motion vectors found by the new point locations, we can estimate more accurate motion coefficient vectors b_r and b_r. It is worth noting that even though the above algorithm can enhance the quality of the motion vectors, it requires additional processing time. Therefore for applications where the accuracy of the initial estimation is sufficient, we skip this optimization step.

2.5 Estimating 3D Palm Position and Orientation

The disparity plane can be mapped onto the palm plane. By locating a coordinate frame on this plane, the position and orientation of the palm can be calculated. Initially, we assume that there is no motion information provided. In section 2.6, we show how the motion information improves this coordinate frame assignment and palm pose estimation.

We find the palm plane in 3D using the calibration information and the disparity plane. Having found the coefficients (c_1, c_2, c_3) of the disparity plane, we can use (2.3) to find (C_1, C_2, C_3) the coefficients of the hand plane in 3D as defined in (2.2), when we have rectified images. A simple method for performing this mapping for unrectified images is to find three points lying on this plane in 3D and then fit a plane to these three points. To find points in 3D, we identify corresponding points from the disparity plane and use a simple triangulation process with the camera calibration information [44].

We define the palm plane as the transformed plane found after two rotations and one translation applied to the camera X-Y plane. Specifically, we rotate the X-Y plane with equation $Z = 0$ first about the X axis and then about the Y axis (i.e., *yaw* and *pitch*) to transform it to $Z = C_1X + C_2Y$. Then, we translate the plane along the Z axis by a constant value C_3 which makes the plane equation $Z = C_1X + C_2Y + C_3$. Coefficient values for C_1, C_2 which were already found through the plane fitting process, are used to determine the two rotation angles ψ and θ corresponding to *yaw* and *pitch* respectively as follows:

$$\psi = tan^{-1}(\frac{C_2}{\sqrt{1+C_1^2}}) \qquad\qquad \theta = tan^{-1}(-C_1) \tag{2.20}$$

Using the two rotation angles ψ and θ, and the translation vector $(0, 0, C_3)^T$, we compute the transformation matrix P, which transforms the X-Y plane to the

hand plane

$$P = \begin{bmatrix} cos(\theta) & sin(\theta)sin(\psi) & sin(\theta)cos(\psi) & 0 \\ 0 & cos(\psi) & -sin(\psi) & 0 \\ -sin(\theta) & cos(\theta)sin(\psi) & cos(\theta)cos(\psi) & C_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.21}$$

This matrix will be used in later stages of processing.

The next step is to assign a coordinate frame to the palm where the X-Y plane of this frame resides on the model plane. This coordinate frame provides the 6 parameters required to determine the position and orientation of the hand in 3D. To determine the position of the hand, we need to assign the origin of the frame to a fixed point on the palm. A good point is the center of the palm which can be approximated by the center of mass of the estimated area of the palm, built as the union of the set of circles as explained in section 2.2.2. The position of the origin $O = (O_X, O_Y, O_Z)^T$ in 3D is calculated through a simple triangulation process.

The rotation of the hand about the Z axis of the palm frame, the *roll*, can be computed using the orientation of the 2D silhouette points of the hand in the X-Y plane. Ignoring some infrequent cases where the arm is hidden and all fingers but the thumb are bent, *roll* can be computed as the angle of the axis of the least moment of inertia [45] and is calculated as

$$\phi = \frac{1}{2} tan^{-1}(\frac{2\mu_{1,1}}{\mu_{2,0} - \mu_{0,2}}) \tag{2.22}$$

where

$$\mu_{p,q} = \sum_{(x,y)} \sum_{\in R} (x - \overline{x})^p (y - \overline{y})^q$$

and

$$\overline{x} = \frac{1}{n} \sum_{(x,y)} \sum_{\in R} x$$

$$\overline{y} = \frac{1}{n} \sum_{(x,y)} \sum_{\in R} y$$

R includes the whole segmented region of the hand. This provides us with the direction of the arm or the rough direction of the fingers in case the arm is missing, and is a good approximation of true hand *roll*.

2.5.1 Experiments and Results

To measure the accuracy of the proposed technique, we compare it with a hand model computed using a set of markers on the palm, finding their positions on the images manually. We compute the coordinates of those points in 3D and fit a plane to them. Figure 2.6 shows a sample image with markers. As depicted in the figure, the positions of the markers are selected so that they cover the area of the palm uniformly. This provides us a better comparison as the region-based method picks points uniformly.

Figure 2.6: A sample image with markers.

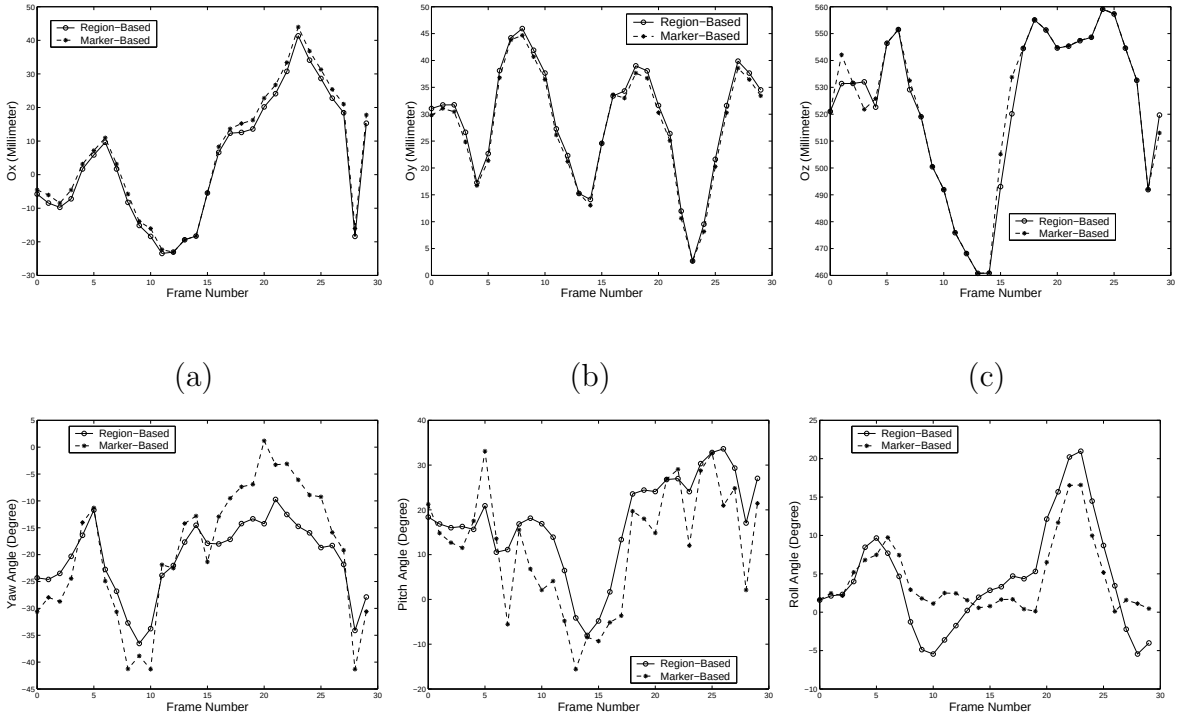


Figure 2.7: Experimental results. Top: Marker and region-based position values: (a) X, (b) Y, (c) Z, Bottom: Marker and region-based orientation values: (d) Yaw, (e) Pitch, (f) Roll.

Figure 2.7 shows the position coordinates O_X, O_Y and O_Z and orientation angles *yaw*, *pitch* and *roll* denoted as ψ, θ, and ϕ of this *marker-based plane* as well as the *region-based plane* estimated through disparity analysis. A sequence of 30 frames was used for this experiment. The results are shown in table 2.1.

Although, the marker-based plane passes through a set of reliable points, this plane may not be the optimal plane as the shape of the palm is not exactly a plane. For this reason we do not regard the marker-based plane as a *ground truth* plane; In fact, we believe that the plane estimated through disparity analysis is a better approximation, giving us more reliable position and orientation parameters.

	mean absolute difference	standard deviation of the absolute difference
O_Z	$1.8135mm$	$0.9215mm$
O_Y	$1.0514mm$	$0.4740mm$
O_Z	$2.0792mm$	$4.0983mm$
ψ	$5.1570°$	$3.2986°$
θ	$6.9515°$	$5.3280°$
ϕ	$3.3571°$	$1.9242°$

Table 2.1: Statistical Results

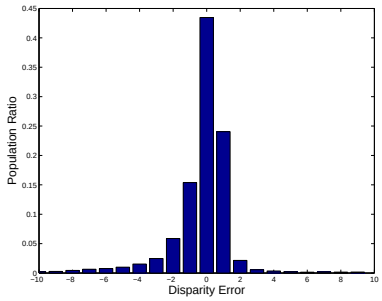


Figure 2.8: Distribution of the error of disparity values with respect to disparity plane. (Averaged over 30 frames)

Another useful parameter that assesses the accuracy of our algorithm captures the distribution of the disparity errors which measures how far the disparity points are from the fitted disparity plane. In other words how many of the points are outliers. This is an important issue because the M-estimation algorithm breaks down if the percentage of outliers is too high and then it diverges from the optimal plane drastically. Figure 2.8 shows the distribution of the errors measured by averaging the corresponding distributions over a 30 frame sequence. It is a normal distribution with mean 0.0050 and standard deviation 0.0237 which gives us a 35% rate of outliers if we define inlier-outlier threshold 1.5 and 9% if threshold is 2.5 levels of disparity. Therefore, the M-estimation algorithm is convergent.

Figures 2.9 shows sample frames selected from different image sequences show-

ing a hand in motion. The left image of the image-pairs along with the corresponding models built based on the estimated position and orientation of the hand are depicted in the figures. Different frames show a variety of cases to which the proposed method is applicable. There are frames where the fingers moving freely, and we still track the palm. Hands from different people in figure 2.10 also shows the applicability of the method in low-textured as well as high-textured cases. It also indicates that our algorithm works on front of the hand as well as the back of the hand.

Figure 2.9: Experimental results: Sample input frames along with corresponding estimated models.

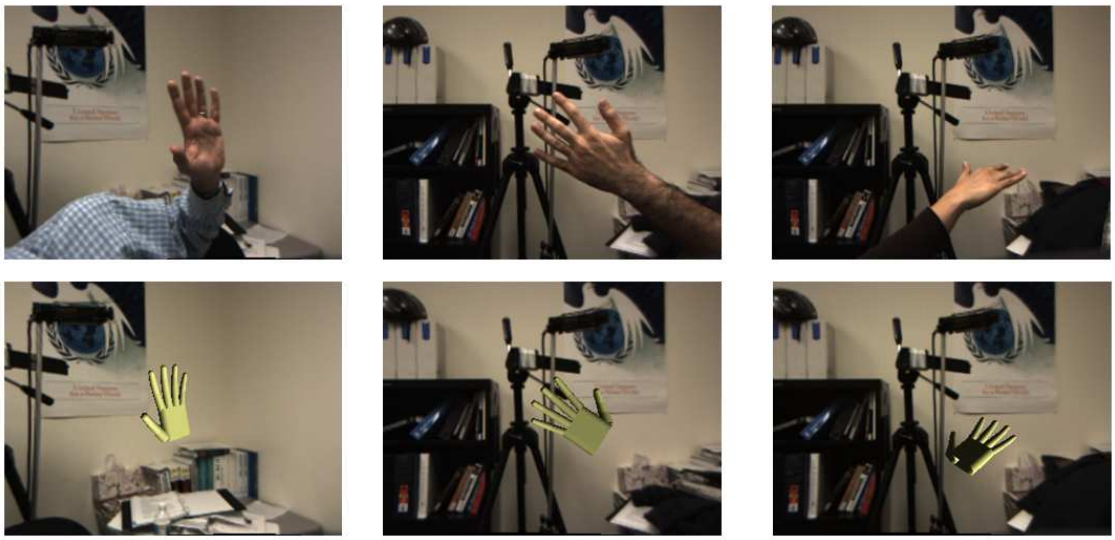


Figure 2.10: Experimental results: Sample frames showing front and back of the hands as well as high-textured and low-textured hands.

2.6 Tracking a Reference Point in 3D

Following the method discussed in the previous section over time, we can track the hand motion in 3D. However, since the location of the center of the hand is not determined accurately, the two center points in consecutive frames are not necessarily images of the same 3D point on the hand. This causes jumps in the hand trajectory. Instead, we track a fixed 3D reference point with the help of the modeled motion field to measure the trajectory of the hand in space. The choice of such a reference point is not critical; however, tracking a point in the center of the segmented region is more reliable than points on the boundary, since we might lose the boundary points as a consequence of hand rotation. Also, given the physical axes of the hand, the effect of rotation on the location of a point is smallest near the hand center. This point is tracked indirectly by employing the parameterized planar

hand region tracking. We estimate the motion model of the whole central region of the hand and reduce the impact of outliers using the method discussed in section 2.4. The new position of the reference point is computed from the motion model and is mapped to a 3D point in space using the parametric model of the disparity map. The algorithm to find the 3D position of the reference point throughout a sequence follows:

1. In the first frame, the reference point in the left image, $p_l^1 = (x_l^1, y_l^1)$ is chosen as the center of mass of the central region of the hand.

2. At time t, the position of the reference point $p_l^t = (x_l^t, y_l^t)$ is computed using the motion model estimated in section 2.4.2:

$$\begin{aligned} x_l^t &= x_l^{t-1} + u_l^t \\ y_l^t &= y_l^{t-1} + v_l^t \end{aligned} \tag{2.23}$$

   where u_l^t and v_l^t are calculated using motion model coefficients $\mathbf{b}$ and equation (2.12). If the size of the motion or the variance of the motion field components is larger than a threshold (e.g. a couple of pixels), we reject the new estimated position of the reference point and recalculate it as for the first frame. This might cause an instantaneous error, but does avoid error accumulation.

3. The disparity map is estimated at each time t.

4. The spatial location of the reference point $P^t = (X^t, Y^t, Z^t)$ corresponding to the image point p_l^t is estimated using disparity model and calibration information:

$$
\begin{aligned}
Z^t &= bf_l/d^t \\
X^t &= (x_l^t - O_{lx})Z^t/f_l \\
Y^t &= (y_l^t - O_{ly})Z^t/f_l
\end{aligned}
\tag{2.24}
$$

where f_l, O_{lx} and O_{ly} are focal length and principle point coordinates of the left camera and b is the baseline of the stereo set, all found through camera calibration process. The disparity value at point p_l^t calculated using the model is denoted as d^t.

In the following sections, we show three applications of hand tracking using 3D tracking of the palm.

2.7 Application: Virtual Drawing

2.7.1 Introduction

Employing the hand as a means for human-computer interaction has been explored extensively in the past few years. Using the hand as a 3D mouse [18, 46], a virtual gun [16], and a remote controller [47] are just a few examples. Communicating *alphabets* to a computer through hand movements is a powerful way for entering information. Much research has been performed to interpret hand gestures as sign language alphabets [48], a method useful mostly for people with disability. However, people typically input information through writing natural language and typing at a keyboard if it is available. Using a keyboard requires a virtual visible keyboard so that user can move the hand to press a desired letter. However, writing

letters does not need such visual feedback. Moreover, shapes other than alphabets can be specified in the same way.

In [49] it is shown how to use paper and the fingertip as a panel and pointer to draw sketches and writings. Tracking the 3D position and orientation of the panel makes it a flexible tool for writing. Nam and Wohn [50] showed how to use Hidden Markov models can be used to recognize drawings made by moving the hand in space. They used a one-hand VPL Dataglove and an attached Polhemus tracker to record the angles of the fingers as well as the 3D absolute position of the hand in space. They assume that there is no hand posture or orientation change while the hand is drawing. In [35], two cameras looking at the hand from the top and the side model the back of the hand as a square to estimate the direction of the hand and then the index finger tip is tracked in 3D.

In our approach, we employ parametric models for fitting both disparity in stereo pairs and motion in monocular video for tracking the hand region in 3D. We do not require the user to maintain her hand in any particular pose (e.g., stretched and separated finger), but track the hand in natural poses that people typically use while writing, for example. We take advantage of the observation that when a person writes (especially using large fonts such as writing on a board), she usually keeps her hand almost rigid and maintain a constant hand pose throughout the writing. As a result, the transformation between a particular point on the hand and the pen point is almost constant. Hence, we can track a fixed point (in 3D) on the hand to determine what the person is writing or drawing. We describe a vision-based system for virtual drawing in space without pen and paper (or board).

Due to the low-textured nature of the hand, fixing and tracking a point on the hand is challenging. A silhouette or contour-based method is not adequate since the hand motion is in 3D space (consider that the hand motion may not be parallel to the image plane). As a result, we need a 3D model for estimation of hand motion. Our approach includes tracking the central region of the hand over time using a combination of parametric models of disparity and motion.

Writing is mostly a 2D activity, except when the hand is lifted off the writing plane. Letters or drawings are sketched on a planar piece of paper as a well-connected series of points. However, writing in space and tracking the hand in a frame-based manner using stereo provides a set of unconnected 3D points. The distance between two consecutive points is determined by the speed of the hand, which is normally not constant. Converting the set of 3D points to a 2D continuous contour is an important component of the application. We develop uniform sampling and planar modeling that allows us to derive an accurate 2D continuous contour.

Writing involves two types of pose and motion of the hand: *on-plane* when the hand is writing, and transient *off-plane* motions performed as gaps between letters or figures. Differentiating between these two activities is essential to virtual writing. We use incremental planar modeling to detect on-plane termination. For initialization, cooperation from the user is expected.

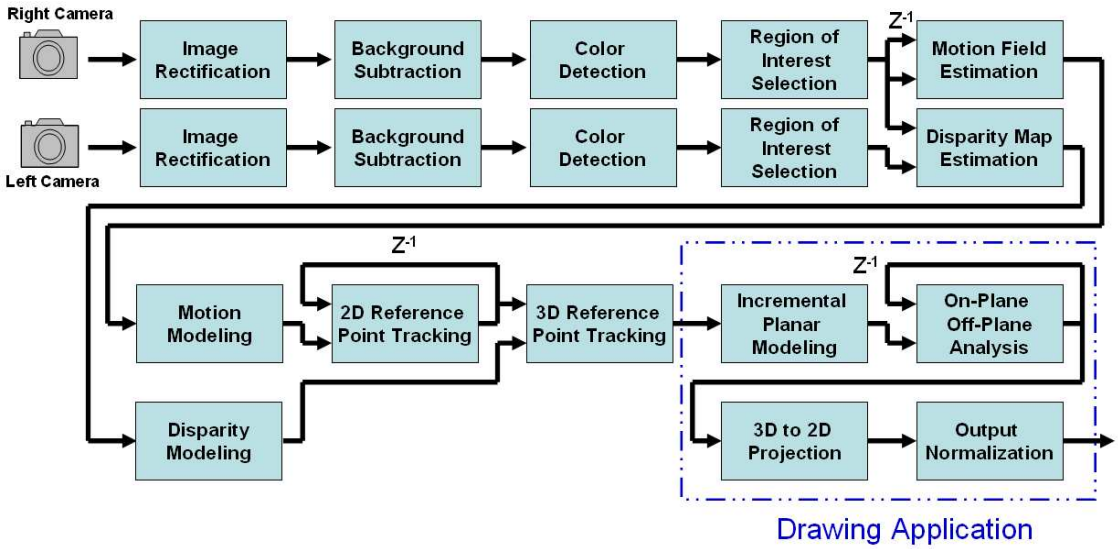


Figure 2.11: Block diagram of the virtual drawing system.

2.7.2 Approach

Figure 2.11 which is an extended version of figure 2.1 shows the block diagram of the drawing system. Different steps of the process follow:

1. Images are obtained from the stereo camera set, the central region of the hand is segmented, the disparity map and motion field are estimated and modeled and the center of the hand region is tracked as a fixed reference point to provide a set of 3D points which determines the hand trajectory in space. All these steps were discussed in detail in previous sections.

2. At each time instant, the set of calculated 3D points are fitted with a plane and the states on-plane (when the hand writes) and off-plane (when the hand is in transition between letters or shapes) are detected. The set of points in the last on-plane state is projected to a plane parallel to the image plane and

a 2D point set is constructed. Section 2.7.3 explains the algorithm in detail.

3. If the user intends to draw a multi-segment figure, an extra step including some orthographic and perspective projection is required to retain the relative size and displacement of the disjoint segments. Section 2.7.4 has more details.

2.7.3 Extracting the Drawn Segment

2.7.3.1 3D to 2D Conversion

Tracking the reference point in 3D over time gives us a set of points:

$$L_3 = \{P^t = (X^t, Y^t, Z^t) | 1 \leq t \leq T\} \tag{2.25}$$

with no guaranteed connectivity. In fact the distance between two consecutive points is determined by the speed of the hand, which is not uniform. Also, we cannot expect the user to move exactly on a plane while she is writing virtually in space. In addition, OCR (Optical Character Recognition) programs expect a set of two dimensional inputs:

$$L_2 = \{p_i = (x_i, y_i) | 1 \leq i \leq N\} \tag{2.26}$$

which should be well connected. On the other hand, we need:

$$\forall i \in [2, N] : \|p_i - p_{i-1}\| < \delta \tag{2.27}$$

to recognize the written letter. Therefore we need to convert the set of 3D points L_3 to the best approximated set L_2 in 2D. We use the robust estimation method defined in section 2.3.2 for this. This allows the user's hand to shake or move in an unexpected way.

As mentioned earlier, the distribution of the reference points on the plane is non-uniform due to variable speed of the hand motion. This might bias the plane toward locations where the hand is moving more slowly. To cope with this variation, a re-sampling of the points in L_3 is performed. Neighboring points are connected using a straight line and then the resulting edge image is sampled uniformly to make a set of points L_3^u with uniform distances in 3D.

2.7.3.2 On-Plane vs. Off-Plane

An essential part of our virtual writing system is to distinguish between on-plane and off-plane states. The on-plane finishing frame is recognized automatically whereas the on-plane starting frame requires the cooperation of the user. The user needs to hold her hand still for a few frames so that the system can detect it as a sign of the start of writing. Thereafter, the system starts fitting planes to the point set L_3^u and incrementally fits the plane in subsequent frames. When it detects a significant deviation from the fitted plane for the last few frames, it recognizes it as a sign of drawing termination. The user usually lifts the hand from the board after writing a letter or drawing a shape; however this action needs to be more conspicuous in virtual writing than when writing on a real plane. To achieve better performance, we fit a planar model to all the points except the last few (to prevent off-plane points from deviating the plane from its true position and causing off-plane detection failure - See figure 2.12). It is worth noting that a similar test could be used to recognize on-plane starting point, where the tracked point resides on a

plane for a few frames; however this needs more cooperation from the user with more controlled movements. Informal testing indicated that users found it more natural to remain still for a few frames.

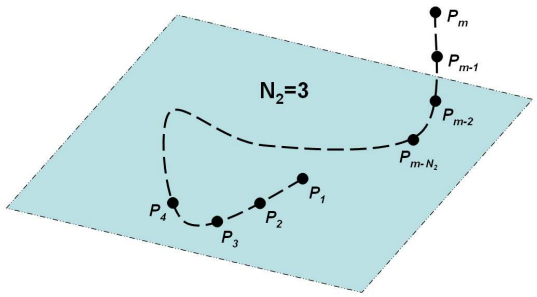


Figure 2.12: Off-plane mode detection: Last N_2 points are supposed to be out of the plane.

2.7.3.3 Algorithm

To extract the user's drawing from the set of points L_3, the following steps are taken at each time instant:

1. The points in the set L_3 are connected sequentially using straight lines and the resulting edge shape is re-sampled uniformly to produce the point set

$$L_3^u = \{P_i = (X_i, Y_i, Z_i) | 1 \leq i \leq M\}$$

where M depends on the sampling rate and is preferably larger than T.

2. If the system is in the off-plane state (which is the initial state), it checks whether there has been any significant displacement in the last N_1 points. For this purpose, the parameter D_1 is calculated:

$$D_1 = \max(\|P_i - P_{i-1}\|) \qquad M - N_1 \leq i \leq M$$

and we switch to on-plane state and reset the index t to one if D_1 is less than a certain threshold.

3. Otherwise, if the system is in the on-plane state, we fit the best plane to a subset L_3^{on} of points in L_3^u

$$L_3^{on} = \{P_i = (X_i, Y_i, Z_i) | P_i \in L_3^u, 1 \leq i \leq M - N_2\}$$

using M-Estimation as discussed in section 2.3.2. As mentioned earlier, the reason for excluding the last N_2 points is that we do not want the potential off-plane points to bias the plane (See figure 2.12). The fitted plane is

$$Z = \alpha_1 X + \alpha_2 Y + \alpha_3 \tag{2.28}$$

4. Parameter D_2 is calculated based on the distance of the last N_2 points to the plane:

$$D_2 = \min(\|P_i - P_i^{proj}\|) \qquad M - N_2 < i \leq M$$

where P_i^{proj} is the projection of point P_i on the estimated plane computed as

$$P_i^{proj} = P_i + \lambda(\alpha_1, \alpha_2, -1)$$

with λ calculated as

$$\lambda = -\frac{\alpha_1 X_i + \alpha_2 Y_i + \alpha_3 - Z_i}{\alpha_1^2 + \alpha_2^2 + 1}$$

If parameter D_2 is larger than a threshold, then we switch to the off-plane state showing that the drawing of one segment is over. Reset the index t to one.

5. If a segment of drawing is just recognized (i.e. L_3^{on}), rotate it such that it resides on a plane parallel to the image plane and denote the new point set as L_3^{rot}

$$L_3^{rot} = \{P_i = (X_i^{rot}, Y_i^{rot}, Z_i^{rot}) | 1 \leq i \leq M - N_2\}$$

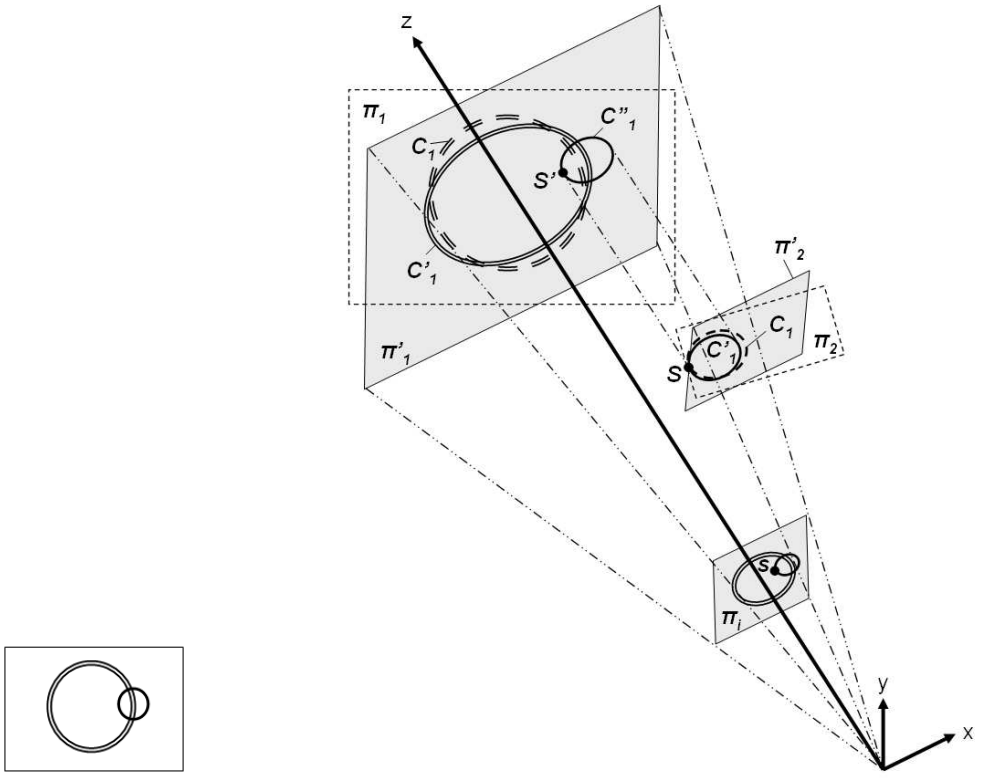
The points in this set should satisfy the condition

$$Z_1^{rot} = Z_2^{rot} = ... = Z_{M-N_2}^{rot}$$

6. Define the new set of 2D points L_2 as

$$L_2^{un} = \{p_i = (X_i^{rot}, Y_i^{rot}) | 1 \leq i \leq M - N_2\}$$

7. Normalize the size and location of the point set L_2^{un} to obtain the final output point set L_2 as defined in (2.26). We can now determine that $N = M - N_2$



(a) (b)

Figure 2.13: Drawing multi-segmented shapes in 3D: (a) desired shapes and output of the system, (b) The 3D scene.

2.7.4 Multi-Segment Drawing with Feedback

Even though the system discussed so far works well for writing alphanumeric characters, where most letters can be drawn using one segment (no off-plane state in the middle), for drawing shapes, the hand moves between off-plane and on-plane states. Therefore, disjoint segments need to be positioned and sized correctly relative to each other. In virtual writing in space, the user does not need to have any visual reference and the system does not consider the location and size of the letters, as these are all normalized in the recognition module. However, when we are dealing

with more than one segment of drawing, the user needs a display which we call an *output board* where the system provides visual feedback about the current position of the user's hand with respect to the already drawn shapes. Also, the user needs to see all the drawn segments to adjust the size and position. The system works as follows: After drawing each segment and fitting the imaginary plane in 3D, the drawn segment is rotated to reside on a plane parallel to the image plane. Also, to keep the size of all the segments proportional to the first one, each rotated segment is projected orthographically to the first projected plane so that the same perspective ratios are applied to all segments while the picture shown on the output board is created. As a specific example, assume we would like to draw two overlapping circles with different sizes as shown in figure 2.13(a). The steps involved in creating this drawing are as follows:

1. The user starts drawing the bigger circle (shown as dashed and outlined in figure 2.13(b)) on an imaginary plane. According to the algorithm in section 2.7.3.1, the best plane in 3D is fitted to the points. The drawn circle and the fitted plane are called C_1 and π_1 respectively.

2. C_1 is rotated into C_1' which resides on plane π_1', a plane parallel to an output board we place on image plane π_i.

3. The circle C_1' is now projected on π_i through a perspective transformation.

4. Finishing the drawing of the first circle and moving away from the estimated plane π_1, the user switches to the off-plane state where his hand moves freely

in space to prepare for the next segment. During this period, the reference point on the central region of the hand is projected orthographically to the plane π_1' and thereafter perspectively to π_i. This gives the user instantaneous feedback of the starting point of the next circle. The reason for using an orthographic projection is that the user does not require perspective projection while drawing in space. On the other hand, the user does not adjust the size of the desired shape with respect to its distance to the camera. Getting real-time feedback from the system, the user moves the hand so that its projection on the board goes to the desired location.

5. The hand stops at point S; its orthographic projection on plane π_1' is denoted as S'. The perspective projection of S' goes to s on the board where the user observes the result. Thereafter, the user remains still for a few frames to let the system know that a new segment is starting.

6. The user draws the second circle C_2 and the second fitting plane π_2 is estimated in the same way as the previous circle. It is worth noting that there is no relationship between planes π_1 and π_2 as they are both imaginary planes in space and the user need not keep their positions in mind.

7. Circle C_2 rotates about point S such that rotated circle C_2' resides on plane π_2' parallel to π_1' and π_i. The reason for using S as the center of rotation is that its position on the rotated shape remains the same, and consequently, its position with respect to circle C_1' is unaltered.

8. Applying orthographic projection to circle C_2' we obtain circle C_2'' on plane π_1'.

9. Perspective projection is performed for circle $''_2$ to add it to the output board where the projection of C_1' already resides. The result is two overlapped circles with the desired size ratio, as shown in figure 2.13(a).

An example of drawing a multi-segment shape is shown in the next section.

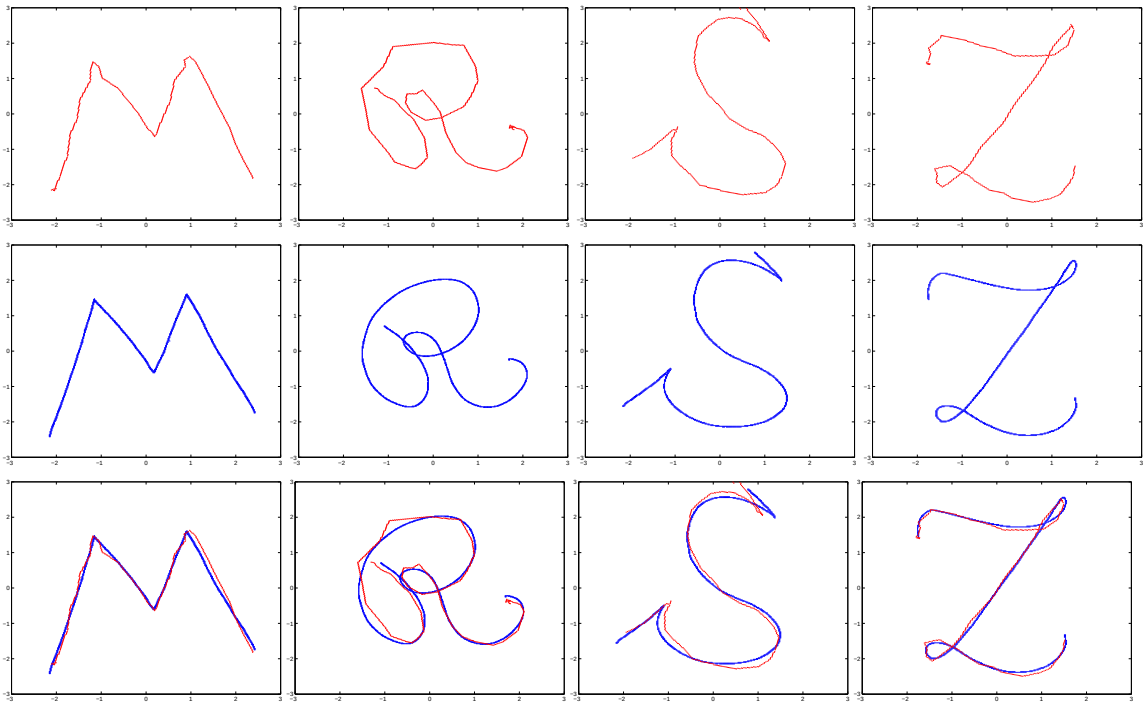


Figure 2.14: (Top Row) Vision-based estimation, (Middle Row) Paper-based output (Bottom Row) overlay of vision-based estimation on paper output.

2.7.5 Experimental Results

To measure the accuracy of the method, sequences were taken where a person was actually writing on paper using a pen. Comparison of the letters extracted from our vision-based system and the real letters written on the paper shows how well our

tracking method corresponds reference points on the hand to the pen point tracks. In figure 2.14, the outputs of our system, as well as the letters on the paper scanned as digital images, are shown. Figure 2.15 shows frames picked from the beginning, middle and end of the writing for the sample letter Z.

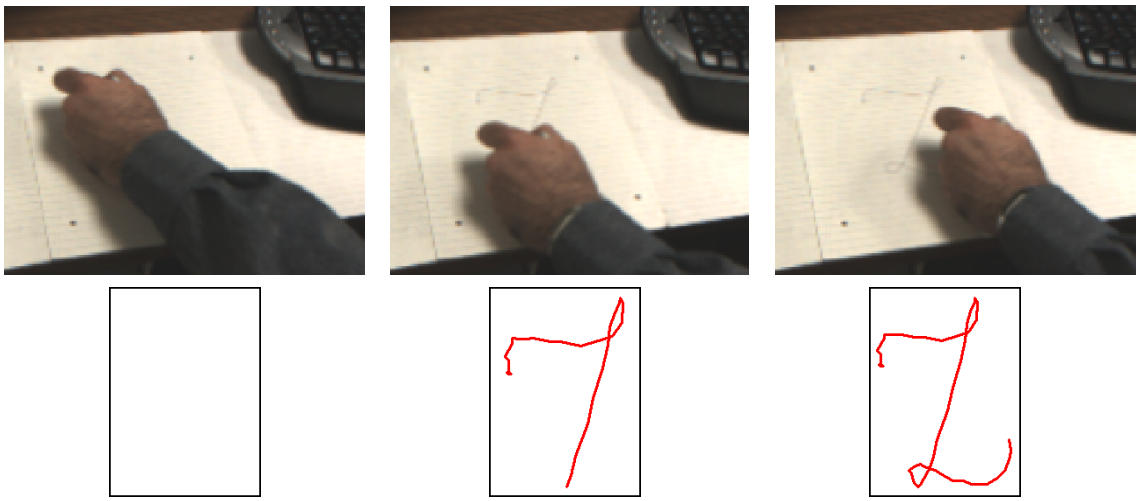


Figure 2.15: Input and output of some sample frames: (Left) Beginning, (Middle) Middle, (Right) End.

The *Chamfer distance* is used to measure the similarity of the two shapes. It is a well-known method to measure the distance between two edge images $\mathbf{X}$ and $\mathbf{Y}$:

$$c(\mathbf{X}, \mathbf{Y}) = \frac{1}{\mid \mathbf{X} \mid} \sum_{x \in \mathbf{X}} \min_{y \in \mathbf{Y}} \parallel x - y \parallel \tag{2.29}$$

To make the measure independent of the size of the image, the distance is normalized by dividing it by the largest dimension of the shape (i.e., the distance of the two furthest points):

$$c_n(\mathbf{X}, \mathbf{Y}) = \frac{c(\mathbf{X}, \mathbf{Y})}{\max_{x_1, x_2 \in \mathbf{X}} \parallel x_1 - x_2 \parallel} \tag{2.30}$$

A more accurate measure, bidirectional Chamfer distance [51], is defined as:

$$C_n(\mathbf{X}, \mathbf{Y}) = \frac{c_n(\mathbf{X}, \mathbf{Y}) + c_n(\mathbf{Y}, \mathbf{X})}{2} \tag{2.31}$$

As the two shapes extracted from the two different methods might be in different position, orientation and scale, we need to find the best translation vector $\mathbf{t} = (t_x, t_y)$, rotation angle θ and scaling factors $\mathbf{s} = (s_x, s_y)$ which minimize the distance thereby maximize the similarity of the two shapes:

$$C_{ns} = \min_{\mathbf{t},\theta,\mathbf{s}} C_n(\mathbf{TX}, \mathbf{Y}) \tag{2.32}$$

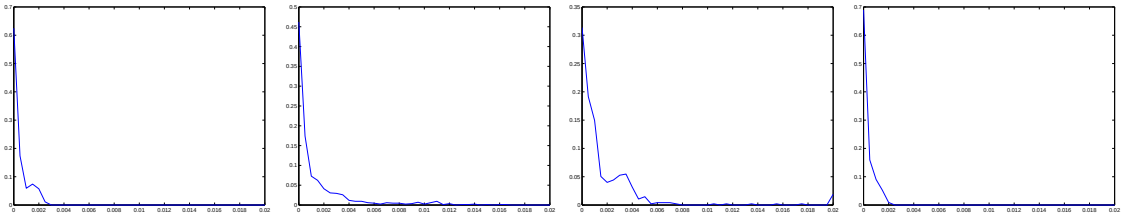
The distance measures C_{ns} for the letters shown in figure 2.14 are listed in Table 2.2.

Letter	$C_{ns}(.)$	$\Delta\psi$ (Degree)	$\Delta\theta$ (Degree)
M	0.0081	-3.90	0.72
R	0.0140	3.19	0.43
S	0.0147	-1.78	4.39
Z	0.0062	-1.98	0.95

Table 2.2: Chamfer distance results

Figure 2.16 also shows the histogram of the distance measure for all the pixels of the shapes.

We employ a second measure for the accuracy of the estimation: The orientation of the paper in 3D is estimated through 4 marker points (see Figure 2.15) drawn on the paper and is then compared with the orientation of the estimated plane (i.e. the final fitted plane to the set of reference points). The orientation angles *yaw* and *pitch* for the two planes denoted as ψ and θ respectively are computed from the 3D



(a) (b) (c) (d)

Figure 2.16: Histogram of the distance measure of all pixels: (a) M, (b) R, (c) S, (d) Z.

plane equation (2.28) as:

$$\psi = tan^{-1}(\frac{\alpha_3}{\sqrt{1+\alpha_2^2}})$$

$$\theta = tan^{-1}(-\alpha_2)$$

Table 2.2 shows the difference of the orientation angles for the two planes in degree.

We applied our virtual writing method to all of the English alphabet letters as well as digits in a continuous writing process (i.e., writing consecutively from A to Z) in space. The quality of the results was good and we anticipate that an OCR algorithm which recognizes handwritten letters can convert the images into coded characters. Figure 2.17 shows output of the program for English letters. Figure 2.18 also shows a few frames of a video sequence for writing the letter B in space and the output of our system.

Next, we illustrate drawing a multi-segment face enhanced by the real-time feedback to the user as explained in section 2.7.4. The user is also provided with three virtual buttons so he can choose the pen color by moving his hand to the area

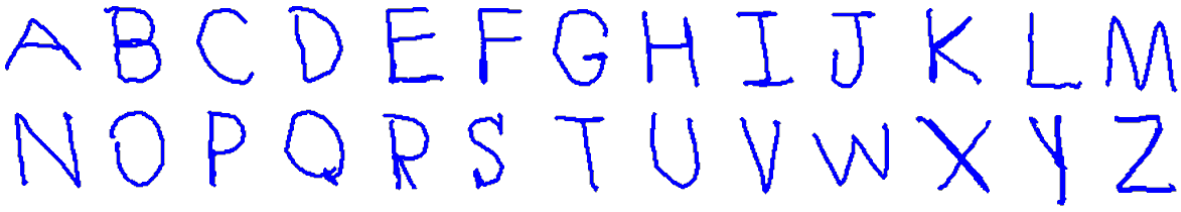


Figure 2.17: Output of the program for English letters.

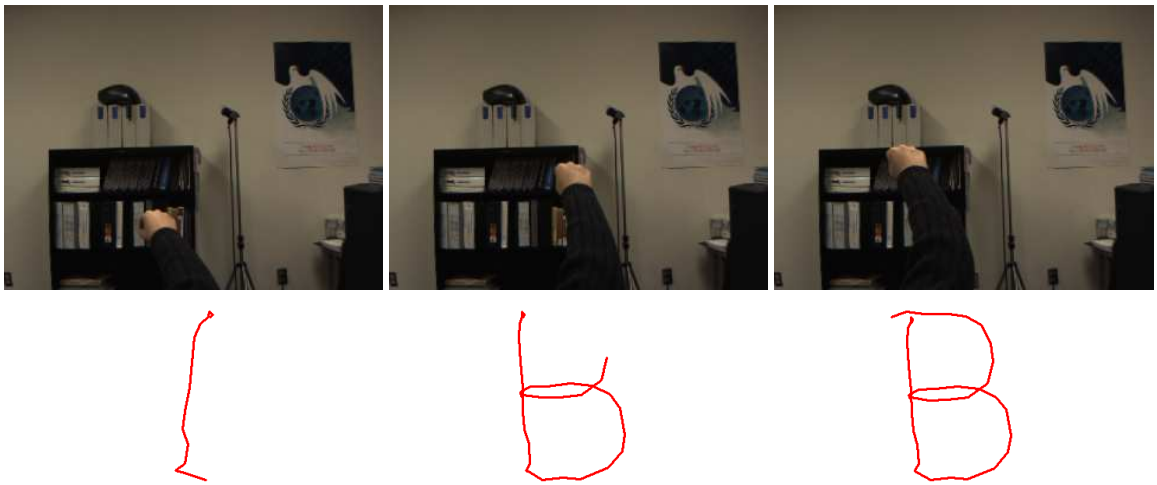


Figure 2.18: Sample frames of writing letter B in space.

of the buttons added to the output board. Figure 2.19 shows a sample frame and the output drawn face. It also shows how the user obtains real-time feedback from the system through the monitor. In fact, he can see the live images taken by the cameras as well as the current state of the output board. He also observes the color buttons in the left side of the output board so he can select the pen color.

Our virtual drawing system requires minimal cooperation from the user. However, we have not conducted user studies to assess the performance and fatigue that may occur in long term use. Nevertheless, we established the feasibility and determined the performance accuracy of our proposed system.

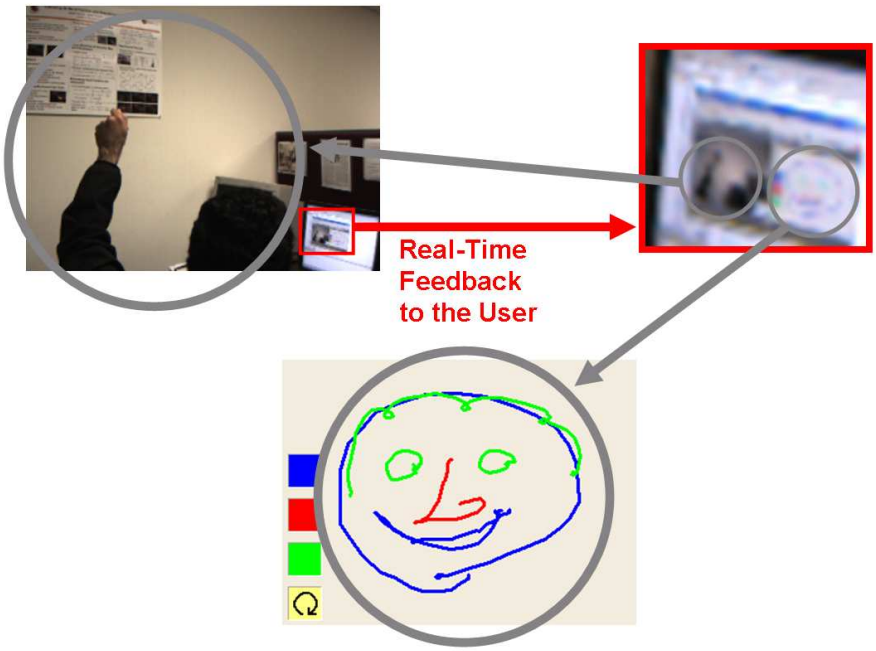


Figure 2.19: A sample frame of the sequence drawing a face showing how user gets real-time visual feedback

2.8 Application: Virtual Marble Game

Visual tracking of human body parts is being used in the game industry [52,53]. Also, manipulating virtual objects not only eliminates the need for constructing expensive physical simulators, but can also support more flexibility. Virtual marble, which resembles a physical toy marble game, is an example of such a virtual object. In this game, the user moves a ball through the hallways of a maze to reach a predefined goal location. The user performs this by moving the hand thereby making a suitable *ramp* for the ball which moves using virtual gravity. In a virtual marble game, the user rotates her hand while the system tracks the hand orientation and simulates the marble board tilts. The system also provides visual feedback of the virtual marble board and the current position of the ball so that the user can adjust

Figure 2.20: Sample frames of a virtual marble game

her hand orientation to navigate the virtual ball toward the goal. Figure 2.20 shows different frames of a sample virtual marble game where both the hand images taken by the camera and the visual scene the user sees are shown.

Our implementation of the virtual marble game estimates the absolute orientation of the hand at each frame and applies it to the model, as shown in figure 2.22. To make the game more intuitive to the user, the initial frame is considered as a reference so that at each frame the model is rotated as much as the difference between orientations in the current frame and the reference frame. Tracking and visual feedback at a rate of about 10 frames per second enables the user to see the current state, decide and tilt the hand to navigate the ball comfortably. To make the game more attractive, physical parameters such as bouncing as a result of collision and inertia could also be modeled.

The flexibility of this virtual game comes from the ability to change the map of the maze easily. Our system modifies it using a random maze generator. Figure

2.21 shows a few sample maze maps. We can also manipulate the coefficient of friction to adjust the level of difficulty of the game and make the navigation more challenging. This friction parameter cannot be easily changed in the physical world. Figure 2.22 shows another example of the virtual marble game.

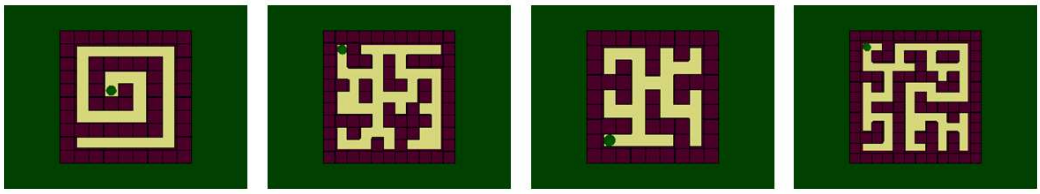


Figure 2.21: Sample maze maps for virtual marble game.

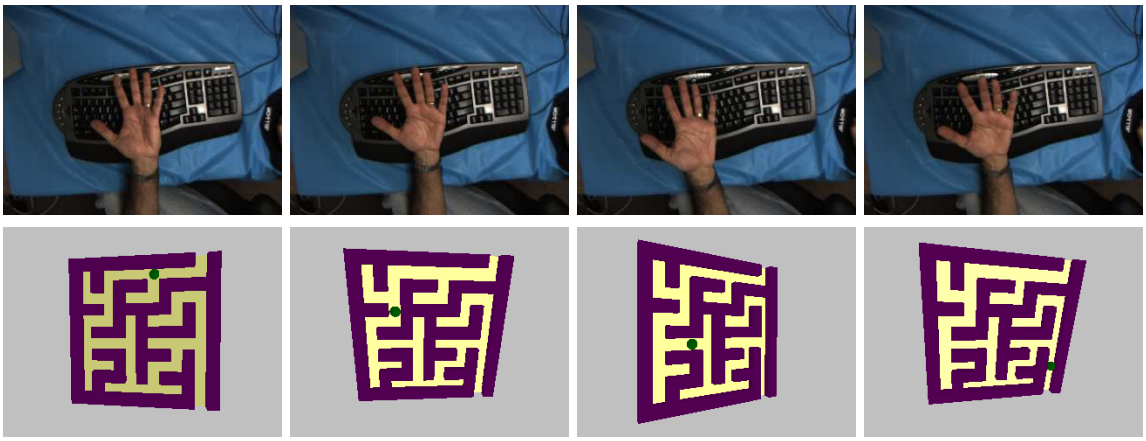


Figure 2.22: Sample frames of a virtual marble game.

2.9 Application: 3D Construction

Tracking the hand trajectory in space as presented in this chapter can open up the opportunity to draw 3D objects in space and communicate them to a computer system where they can be rendered as 3D object models.

In the 3D construction application, a user moves her hand over the edges of a

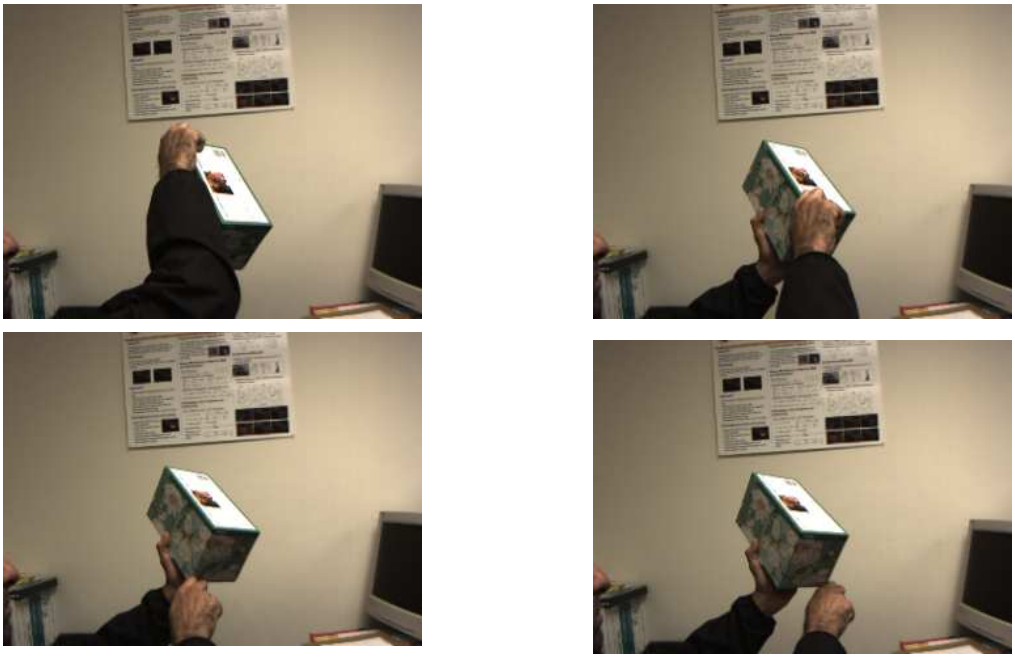


Figure 2.23: Sample Frames of the hand traversing three orthogonal sides of a box.

physical 3D object and the system tracks the hand to measure the dimensions of the object and to render the object virtually. We assume that the user's hand is held rigid with respect to the edges of the object and the back of the hand remains visible throughout. Figure 2.23 shows sample frames in which a user moves his hand along three orthogonal sides of a box. Measurements performed demonstrate the accuracy of the hand tracking method. A few parameters calculated from hand tracking in the sequence were compared with the ground truth measured from the actual box and the results are summarized in Table 2.9. The measured parameters include the angle between the two planes p_1 and p_2, the angles between the lines l_1 and l_2 and the lines l_2 and l_3, and the length of the lines l_1, l_2 and l_3 as defined in figure 2.24. As indicated, the relative errors are small and are mostly due to camera calibration inaccuracy as well as shaking of the hand holding the box.

Parameter	Nominal Val.	Measured Val.	Rel. Error
$Angle(p_1, p_2)$	90°	88.25°	1.94%
$Angle(l_1, l_2)$	90°	93.19°	3.33%
$Angle(l_2, l_3)$	90°	92.95°	2.22%
$Length(l_1)$	$238mm$	$208mm$	12.18%
$Length(l_2)$	$132mm$	$127mm$	3.79%
$Length(l_3)$	$120mm$	$106mm$	10.83%

Table 2.3: Parameter Comparison between hand tracking approach and actual box measurements

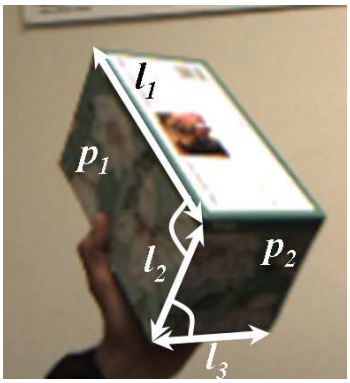


Figure 2.24: The tracked box and defined measurement parameters.

2.10 Summary

In this chapter, a method for tracking the hand in 3D space was presented. Based on a stereo camera set, a sequence of image pairs is acquired and analyzed to estimate the position and orientation of the hand in 3D. Since due to low-textured nature of the hand, point matching algorithms provide noisy data, we estimate the disparity map and motion field and model them to reduce the impact of the low-textured hand and noise. Planar modeling of the hand requires disparity values to reside on a plane too. A plane in motion defines a quadratic model for the motion field, where model parameters are estimated using robust estimation and adjusted to comply with the disparity model. Virtual drawing in space, virtual marble game,

and 3D object construction were presented as three sample applications.

Chapter 3

Multiple Hand/Head Tracking using Multiple Cameras

3.1 Introduction

In this chapter, we address the problem of tracking the hands and head of a person or multiple people interacting with each other in a scene viewed by a set of cameras. We pose a multiple target tracking problem and propose a two-layer solution consisting of a particle filtering layer and a finite state machine. Also, we discuss the activity recognition problem for the set of activities involving heads and hands of human subjects. Color analysis of the area surrounding the hands is presented to determine whether a person holds an object or not. A new approach is suggested to determine the reliability of each image and to combine the color information extracted from different cameras.

3.1.1 Previous Work

The problem of tracking hands and heads of people in space can be posed as an instance of the Multiple Target Tracking (MTT) problem which has been extensively studied in the fields of radar and signal processing [54,55]. The main challenge is the assignment of observations to the multiple targets [56] to simultaneously solve the two distinct problems of data association and target location estimation. Multiple hypothesis tracking [57], probabilistic multiple hypothesis tracking [58], and joint

probability data association [59] are the classical approaches for data association used in the literature.

The problem of target tracking becomes more challenging when interactions between the targets need to be addressed. Khan et al. [60] showed how a Markov random field motion prior can substantially improve tracking when targets interact. They also incorporated Markov chain Monte Carlo sampling to improve efficiency. In some applications, the number of targets in the scene is unknown and even variable over time. Särkkä et al. proposed Rao-Blackwellized Monte Carlo data association [61] and extended it to track an unknown and time varying number of targets [62]. They modeled both the target states and the data associations and also the births and deaths of the targets as hidden stochastic processes observed through measurements.

Employing particle filtering for multiple target tracking has received much attention in the last decade [54, 56, 62]. Avitzour [63] and Gordon [64] investigated the feasibility of this technique for MTT from a theoretical perspective. Morelande and Musicki showed that the use of particle filtering is particularly advantageous in scenarios where targets are in close proximity [65]. In such scenarios, the posterior distribution is multi-modal and tracking multiple modes becomes more effective. Inherent characteristics of particle filters makes this approach feasible.

In the field of computer vision, particle filtering is used for multiple target tracking. Yang et al. [66] modeled multiple objects using color and edge orientation histogram features and applied a particle filtering algorithm for tracking. They tracked the objects in a 2D image space using a single camera. In other work,

Isard and MacCormick [67] proposed a likelihood function for tracking multiple blobs. They used a particle filter to infer the number of objects as well as their configurations. Qian el al. [68] employed singular value decomposition to cluster the samples related to different moving targets and tracked the posterior distribution of the motion parameters using particle filters. Shan et al. [69] employed a combination of a particle filter and mean shift for 2D hand tracking in image space.

In most work considering target coincidence, the consistency of the behavior or appearance of the targets are the main cues to distinguish the targets after they separate [60,70]; however, in the application addressed in this chapter, none of these assumptions hold. In fact, the targets being tracked (hands and head) occupy a very small portion of the image and also change their appearances constantly (especially the hands). Meanwhile, after coincidence, the hands usually do not continue their motions in the same direction, so the motion continuity assumption is also violated. In fact, the behavior of the limbs after an interaction is determined by the activity they are involved in.

3.2 Overview of the Tracking Method

The proposed tracking algorithm consists of two layers:

1. *Low level layer*: Here, a set of parallel *particle filters* are deployed. Each particle filter tracks an individual target in the tracking space. In this layer, no interaction is considered between these filters. The only relationship between the filters is through the shared observation space where a set of common

observations are made.

2. *High level layer*: Here, the particle filters are assigned a state based on their likelihood levels as well as their interactions. Each particle filter can be in one of the following states: *Uninitialized*, *unlabelled*, *normal*, *combined*, or *lost*. Assigning a state to each target enables handling situations where a few targets join and separate or disappear and reappear in the scene. The identity of the targets being tracked are also determined in this layer. Labelling the targets in the scene is essential in activity recognition as addressed in this chapter.

3.3 Single Target Tracking Using Particle Filters

Particle filtering [71, 72] is a powerful method for target tracking. In this section, we discuss how a particle filter is deployed for tracking a single target.

3.3.1 Bayesian Target Tracking

Consider a system with dynamic equation

$$x_k = f_k(x_{k-1}, v_{k-1}) \tag{3.1}$$

where x_k is a n-tuple state vector in the state space, v_{k-1} is an i.i.d. noise process and $f_k : \Re^n \times \Re^n \rightarrow \Re^n$ is a non-linear function. Also, consider an observation space, where each observation vector z_k is related to the state vector x_k through a non-linear function

$$z_k = h_k(x_k, n_k) \tag{3.2}$$

where n_k is also an i.i.d. process independent from v_k. The tracking problem is to estimate the state variable x_k from the set of all the observations $z_{1:k}$ measured from the beginning to the current time instant k.

According to the Bayesian approach, the best estimation for x_k is performed through a recursive process including the following two steps:

1. At time k, the prior pdf of the state vector is predicted using the *Chapman-Kolmogorov equation* [73] and the *Markov property* [74] of the process:

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1} \tag{3.3}$$

   where $p(x_k|x_{k-1})$ can be computed using equation (3.1).

2. When the observation z_k becomes available at time k, the prior can be updated using *Bayes' rule*:

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})} \tag{3.4}$$

Different methods and algorithms have been proposed to solve the above equations in an optimal or sub-optimal way; however, each one of these solutions is based on a set of limiting conditions and assumptions.

Kalman filtering [75] is an approach to finding the optimal posterior density assuming that the posterior density and the noise processes all have Gaussian models and the dynamical model of equation 3.1 defines a linear function of x_{k-1} and v_{k-1}.

Another set of methods, called *Grid-based methods* [76], assume that the state space is discrete and consists of a finite number of states. Relying on these assumptions, finding the optimal solutions becomes possible.

Sometimes, the above assumptions do not hold and approximations must be made to use some sub-optimal methods:

If the functions $f_k(.)$ in equation 3.1 are non-linear, a local linearization may be sufficient. The *Extended Kalman Filter* [75] is based on such an approximation. Note that it still makes the Gaussian model assumption.

If the state space is inherently continuous but can be dicretized by grouping the states into cells, an approximate Grid-based method [76] can be applied. *Hidden Markov Model (HMM) filters* [77] are an application of such a method.

Particle filtering methods are also another way of approximating the optimal solutions while relaxing the linearity of the state dynamics and the Gaussian models. We explain particle filtering methods in the next section.

3.3.2 Particle Filtering Methods

3.3.2.1 Sequantial Importance Sampling

Estimating probability density functions using *Monte Carlo simulations* is one of the approximations to analytical solutions. This approach is usually used when the underlying density is complex and not necessarily parametric. The *Sequential Importance Sampling (SIS) Algorithm* [71, 72] referred to as bootstrap filtering [64], the condensation algorithm [78, 79], and particle filtering [80] is a Monte Carlo method which represents the posterior density function by a set of random samples each having a weight proportional to its likelihood.

Let $\{x^i_{0:k}, w^i_k\}^{N_s}_{i=1}$ denote the vectors of sample points from the posterior density

function $p(x_{0:k}|z_{1:k})$ at time instants 0 to k ($p(x_0)$ is actually the prior) and their corresponding weights. Then, at time k, the posterior density can be approximated as

$$p(x_{0:k}|z_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta(x_{0:k} - x_{0:k}^i) \tag{3.5}$$

with $\delta(.)$ defining the *unit impulse function*. Note that the weights need to be normalized to represent the probability density function.

The goal of all particle filtering approaches is to determine good values for the weights w_k^i. In the SIS algorithm, these weights are estimated based on the *Importance Sampling Principle* [81]. According to this principle, if the desired density function $p(.)$ is difficult to draw sample from, yet is calculable at any single point, the samples x^i are instead sampled from another distribution $q(.)$, the *Importance density* and the weights of the samples are determined as

$$w^i = \frac{p(x^i)}{q(x^i)} \tag{3.6}$$

It can be shown [76] that using the importance sampling principle and the Markov property of the process, the weights in equation (3.5) are updated at each time frame using

$$w_k^i \propto w_{k-1}^i \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{q(x_k^i|x_{k-1}^i, z_k)} \tag{3.7}$$

and the posterior density $p(x_k|z_{1:k})$ is approximated as

$$p(x_k|z_{1:k}) \approx \sum_{i=1}^{N_s} w_k^i \delta(x_k - x_k^i) \tag{3.8}$$

This approximation converges to the true posterior density as the number of samples, N_s increases.

Based on the above arguments, the SIS algorithm is summarized as follows:

At each time instant k, a set of N_s samples are drawn from the importance density $q(x_k|x_{k-1}^i, z_k)$ and weights are assigned to them according to 3.7.

Even though the above algorithm works effectively in approximating the true posterior density without having prior knowledge about its shape, it faces some common problems:

1. In [82], it has been shown that the variance of the weights w_k^i increase over time. This makes the degeneracy problem inevitable where all except a few of the particles have negligible weights. To avoid this, a resampling mechanism is deployed to replace the low-weight samples with ones with higher weights. In fact, the probability of selecting x_k^i for the new set equals its normalized weight w_k^i. The weights of all the particles are now reset to $w_k^i = 1/N_s$. Also, a qualitative measure N_{eff} is defined to determine when the resampling process is required:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N_s}(w_k^i)^2} \tag{3.9}$$

   A small value of N_{eff} indicates severe degeneracy and the need for resampling.

2. The weight formula of equation (3.7) shows that the variance of the weights w_k^i increases as the two density functions $p(.)$ and $q(.)$ deviate from each other. This means that if the importance density $q(x_k|x_{k-1}^i, z_k)$ is not selected properly, the chance of degeneracy problem increases. This requires us to choose the importance density as close to the true density $p(x_k|x_{k-1}^i, z_k)$ as possible.

One convenient choice for the importance density is the prior density function:

$$q(x_k|x_{k-1}^i, z_k) = p(x_k|x_{k-1}^i) \tag{3.10}$$

Making this choice, the weight update equation (3.7) can be rewritten as

$$w_k^i \propto w_{k-1}^i p(z_k|x_k^i) \tag{3.11}$$

The Sequential Importance Sampling algorithm explained above is the basis for most of the particle filtering methods; however there are some other algorithms used in the literature, which can be considered as special cases for SIS. *Sampling Importance Resampling (SIR)* filter, *Auxiliary Sampling Importance Resampling (ASIR)* filter, and *Regularized Particle filter (RPF)* are some examples of these methods. Sampling Importance Resampling is discussed in the following section. For learning about different particle filtering methods and also a comparison between all the methods discussed in this section, refer to [76].

3.3.2.2 Sampling Importance Resampling

The Sampling Importance Resampling (SIR) filter is another Monte Carlo method for solving recursive Bayesian filtering problems. The algorithm is obtained by making the following modifications to the SIS algorithm:

1. The prior density $p(x_k|x_{k-1}^i)$ is chosen as the importance density $q(x_k|x_{k-1}^i, z_k)$. This choice eliminates the need for involving the observations in the sampling step. This eases the sampling process and importance weight calculation and is useful especially in cases where observations have a complex nature.

2. The resampling step is applied at every time instant. This means that $w_{k-1}^i = 1/N_s \ \ \forall i$ and therefore

$$w_k^i = \lambda p(z_k|x_k^i) \tag{3.12}$$

where λ is the normalizing factor.

The independence of the sampling step from the observations z_k results in exploring the state space without observation, which may cause inefficiency or sensitivity to outliers. Also, the resampling step may reduce the diversity of the particles. Therefore, some further steps may be taken and some application-based heuristics be applied to improve the performance of the SIR algorithm. The extra steps and heuristics for our particular problem will be discussed later.

3.4 Multiple Target Tracking

The goal of a multiple target tracking system is to track a set of N moving targets in an n-dimensional space called the *tracking space*, while being observed by M sensors. The motions of the targets are considered independent from each other and follow dynamical models with variable accelerations. Along their trajectory in space, some of these targets may pass near each other or even join each other and move together for a while; our assumption is that no more than two targets join or become very close to each other at the same time. Even though removing this assumption is not theoretically hard, it is beyond the scope of this dissertation.

One particular property of the problem addressed here is that the multiple deployed sensors make their observations in a space which is different (and usually

of lower dimension) than the tracking space. This space is called the *observation space*. One important point to notice is that the difference between the two spaces we are working with implies that no target can be tracked merely using the information acquired by a single sensor. It is assumed that any target must be detected by at least two sensors to be trackable. A situation where a single target is detected by fewer than two sensors is referred to as a *lost track*.

As mentioned in section 3.2, the tracking process consists of two steps. In the first step, each target is tracked using a particle filter as explained in section 3.3.2. We focus on the Sampling Importance Resampling (SIR) method and its corresponding equations; however, other particle filtering methods can also be used.

According to equation (3.12), the likelihood function $p(z_k|x_k^i)$ needs to be computed to determine the particle weights. Applying Bayes' rule, the likelihood function can be rewritten as

$$p(z_k|x_k^i) = \lambda' p(x_k^i|z_k)p(z_k) \tag{3.13}$$

with λ' being the normalizing factor. In a multiple object environment, there are multiple observations $z_k = \{z_k^j | j = 1, ..., N_c\}$. Therefore, equation (3.13) can be rewritten as

$$p(z_k|x_k^i) = \lambda' \sum_{j=1}^{N_c} \psi(z_k^j) p(x_k^i|z_k^j) p(z_k^j) \tag{3.14}$$

where $\psi(z_k^j) \in [0, 1]$ is an *indicator function* with the property that

$$\sum_{j=1}^{N_c} \psi(z_k^j) = 1 \tag{3.15}$$

In our multiple target tracking problem, the likelihood and indicator functions should be calculated for all the parallel particle filters. The problem of determining

the set of indicator functions $\psi_n(z_k^j)$ for $n = 1, ..., N_c$ turns into the *data association problem* for multiple target tracking. There are several methods for data association which can be divided into two main classes: *Unique-neighbor* data association methods which associate each observation with one target track and *all-neighbors* data association methods which exploit all the observations for updating all the target track estimates [62]. In the former class, for any target n and at any time instant k, all the $\psi_n(z_k^j)$ values are zeros except one. *Multiple hypothesis tracking (MHT)* [57] which is one of the methods of this class, calculates the likelihood of the observations and the posterior probability of the hypotheses and stores only the most probable hypothesis. *Probabilistic multiple hypothesis tracking (PMHT)* [58] is a modification of the MHT, where by assuming independence between the data associations and the target tracks, it reduces the computational complexity.

In *joint probability data association (JPDA)* [59] which is one of the methods of the second class, each of the target estimates gets updated using every observation with weights that depend on the predicated probabilities of the associations [62]. Since, in our problem, the number of observations is considerably more than the number of targets and each target might correspond to several observations, a similar approach is taken for the association. In fact, by assigning $\psi_n(z_k^j) = 1 \quad \forall j, k, n$, the likelihood $p(x_k^i|z_k^j)$ determines how much each of the observation prior probability $p(z_k^j)$ should be involved in the target posterior density estimation.

3.4.1 Observation Prior Probability Estimation

In this section, we discuss how to determine $p(z_k^j)$, the prior probability of the observations. As mentioned earlier, it is assumed that the observation space is different from the tracking space; however, to evaluate $p(x_k^i|z_k^j)$ as required in equation (3.14), x_k^i and z_k^j need to be compared in a common space. There are two options:

1. All the particles from the tracking space can be projected to the observation space of each sensor, which is usually of a lower dimension, and then be compared to the available observations of that sensor. All the likelihoods need to be aggregated to make the final likelihood measure.

2. As an alternative, the observations may be re-projected to the tracking space; however, since we assumed lower dimensionality for the observation space, no single observation can determine a single point in the tracking space. Therefore, observations from different sensors should be combined to construct points in the tracking space, which we call *candidate points*. Consequently, based on equation (3.14), for each particle, the posterior probability of each particle is a combination of its likelihood for all the candidate points multiplied by the prior probability of the candidate points.

Each of these approaches has disadvantages. In the former method, all the particles of all the filters need to be projected to all the sensor spaces, which is a computationally-intensive process. Also, since some of the targets might not have been detected by some sensors, trying to find likelihoods in those observation spaces

is futile. On the other hand, in the latter approach, combining two observations corresponding to two different targets will result in an invalid candidate point in the tracking space. However, since there are ways to evaluate the validity of each candidate points, this approach is selected.

Let z_{km}^{j} denote the j^{th} observation made by the m^{th} sensor at time k where $j = 1, ..., N_{om}$ and $m = 1, ..., M$. As mentioned earlier, it is assumed that having the information of a target point acquired by two sensors, we are able to re-project this information to the tracking space. This re-projection is defined using a function R as

$$Z_k^c = R(z_{km}^{j}, z_{km'}^{j'}) \tag{3.16}$$

where Z_k^c denotes the c^{th} re-projected observation point, called the *candidate point* in the tracking space. Using candidate points in the tracking space, equation (3.14) can be rewritten as

$$p(z_k|x_k^i) = \lambda' \sum_{c=1}^{N_c} p(x_k^i|Z_k^c)p(Z_k^c) \tag{3.17}$$

Note that, as mentioned earlier, we assign $\psi(Z_k^c) = 1$ for $c = 1, ..., N_c$ and remove them from the formula.

Combining all the pairs of observation points acquired by different sensors, N_c candidate points are obtained where N_c can be computed as

$$N_c = \sum_{m=1}^{M} \sum_{m'=m+1}^{M} N_{om} N_{om'} \tag{3.18}$$

with N_{om} and $N_{om'}$ being the number of observations made by sensors m and m' respectively; however, it is clear that not all of these candidate points correspond to valid targets. For example, if observations associated with two different targets

made by two different sensors make a pair and are re-projected to the tracking space using equation (3.16), the result will be an invalid point and should be treated as a *false alarm*; therefore we need to devise a scheme to determine these points. This scheme uses the prior probability measurement in a way that small $p(Z_k^c)$ values are assigned to the invalid points. Also, due to noise, some of the observations may not be as accurate and they should also be penalized in the same way.

To evaluate how good a candidate point is, the prior measure $p(Z_k^c)$ is constructed with three terms:

$$p(Z_k^c) = p^o(Z_k^c)p^t(Z_k^c)p^b(Z_k^c) \tag{3.19}$$

where $p^o(Z_k^c)$ is a means for evaluating the quality of the measurements in the observations space, $p^t(Z_k^c)$ is a function depending on the error of the re-projection from the observation space to the tracking space and $p^b(Z_k^c)$ is a function of the error in projecting the candidate points back to the observation space. It is worth noting that in some applications, some of these terms may be redundant and would therefore be removed, but as it will be shown in section 3.5.4, we estimate and use all these terms in the hand/head tracking application.

3.4.2 High Level Tracking Layer

Finite State Machine (FSM) can be used to model the functionality of a system based on inputs. In fact, it can be considered as a high-level module to interpret and control the behavior of the low-level modules of a system. In our multiple target tracking problem, a *Finite State Machine (FSM)* is deployed to detect events such

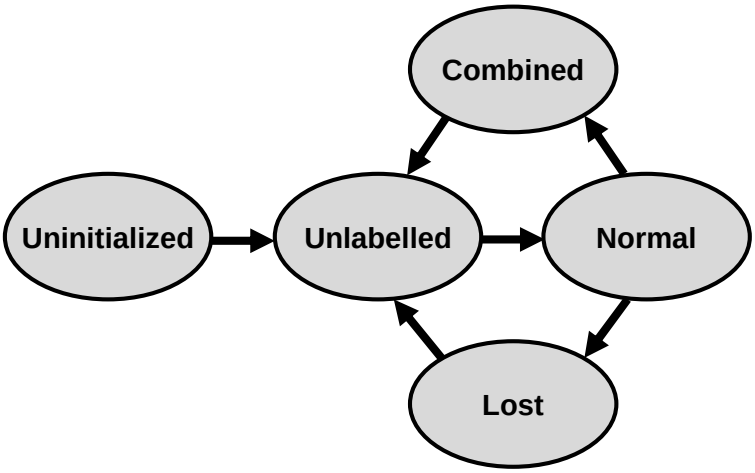


Figure 3.1: Finite state machine for multiple target tracking.

as when track of a target is lost or targets join and follow the same track. Figure 3.1 shows the states and transitions of the FSM. Note that the states are assigned to each of the trackers; therefore at a certain time, two different trackers may be in different states.

The states of the FSM are defined as follow:

Uninitialized State: In the beginning, all trackers are in the uninitialized state. As time passes, observations are made by the sensors and candidate points are defined. As the prior probability $p(Z_k^c)$ becomes larger than a threshold, a new tracker is initialized and the likelihood evaluation is performed using equation (3.17). The initialized tracker then moves to the unlabelled state.

Unlabelled State: In this state, the location of the tracker in the tracking space is initialized and tracking via a particle filter is in progress; however, the type of the target is not determined. This is only an issue in applications where targets are of variant types. For example, in the hand/head tracking application,

the targets are head, left hand, and right hand of a person or a group of people. Activity recognition applications heavily depend on the types of the targets. In such applications, it is important to find cues for target type recognition so that a target is labelled properly. When a target under track is labelled, the tracker makes transition to the normal state.

Normal State: When a target tracker is initialized and labelled, it moves to the normal state. In this state, the target can be tracked efficiently by a particle filter through prior and likelihood measurements and posterior probability estimation as explained in section 3.3.2.

Combined State: As two sample targets T^i and $T^{i'}$ get close to each other, independently tracking them becomes difficult. This is due to the fact that the observations corresponding to the two targets may join in some sensors. Also, the error of re-projecting z_m^j and $z_{m'}^{j'}$ corresponding respectively to T^i and $T^{i'}$ in sensors m and m' becomes smaller and even comparable to the error for the true candidate points representing T^i and $T^{i'}$. In this situation, some of the particles may assign higher likelihood to the invalid candidate points or even the other valid candidate points. This makes the posterior probability $p(z_k|x_k^i)$ a multi-modal distribution with modes becoming similar in a probabilistic sense. As a result, individual tracking of the targets loses accuracy. The proposed action in such situation is to combine the two trackers and track them using a single particle filter. This is done when the maximum likelihood points of the two distributions have a distance less than a threshold. Recognizing this moment also helps in activity recognition as shown in section 3.6.

As long as the two trackers are in the combined state, the modes of the posterior probability $p(z_k|x_k^i)$ should be extracted and monitored. As the two targets move, the positions of these modes change accordingly. They move apart when the targets move away from each other. This trend finally results in separation of the two targets, which can be detected by thresholding the distances between the modes. At this time, the two target trackers each receive a mode and both change their states to the unlabelled state, where they need to be assigned correct labels corresponding to their types. The cues for labelling the targets at this stage are application dependent; the hand/head tracking based cues will be discussed later.

Lost State: As mentioned earlier, it is assumed that to re-project a target to the tracking space, it should be observed by at least two sensors; however, there might be cases where a particular target is detected by fewer than two sensors at the same time. In this scenario, no candidate point will represent the desired target and the corresponding tracker should move to the lost state; however detection of this situation is challenging. When a desired candidate point is not available, the particles tend to approach other candidate points which represent other targets. The proposed solution is to test the posterior probabilities against a threshold. The lost state is similar to the uninitialized state with the difference that it tries to use prior information of the target position to retrieve the track.

3.5 Multiple Hand/Head Tracking

Based on the algorithms presented in the previous sections, we now address the problem of tracking multiple human hands and heads viewed by multiple cameras. As shown later, this tracking can have various applications.

Since, the target detection in this problem is preformed via color segmentation of the images, we assume that heads and hands are the only major body parts with skin color and the only other skin-color regions in the foreground are randomly appearing noise. Even though we rely on the assumption that no other body part is unclothed and the dresses of the subjects do not contain any region of skin-like color, there are some methods in the literature to generalize our color-segmentation approach for better detection of the desired targets. For example, shape analysis and modeling of the body parts can help us to locate hands or heads in the images; however as several people interact in the scene, this approach also becomes challenging. Also, modeling the body would make the algorithm more complex and time consuming.

Figures 3.2 and 3.3 show two indoor scenes with one and two human subjects respectively. For our experiments, four color cameras are used which look at the scene from different directions. Increasing the number of cameras will naturally increase the quality of tracking; however it also makes the tracking more computationally intense. Using fewer cameras increases the chance of losing track of targets as some of the limbs often get occluded in some cameras and so cannot be detected.

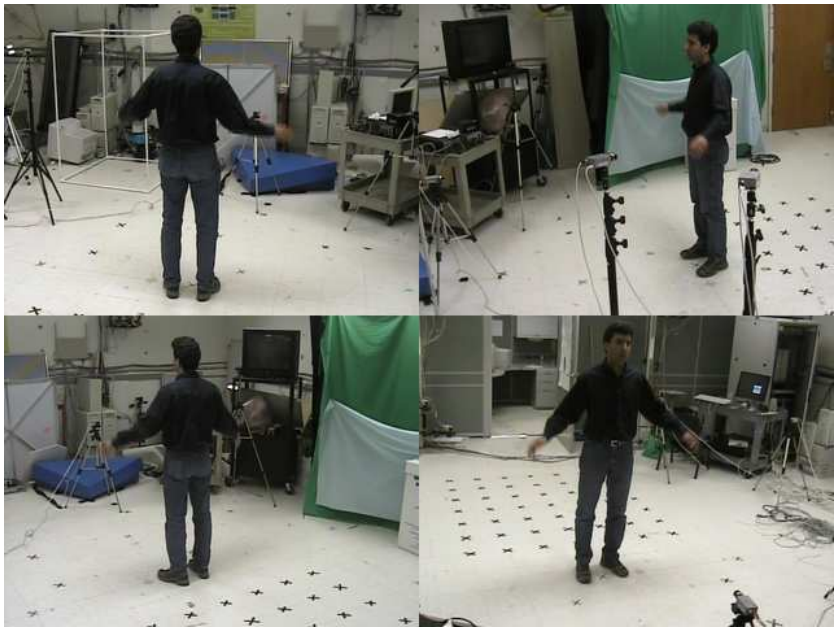


Figure 3.2: Sample input images taken by four cameras from a single human subject.

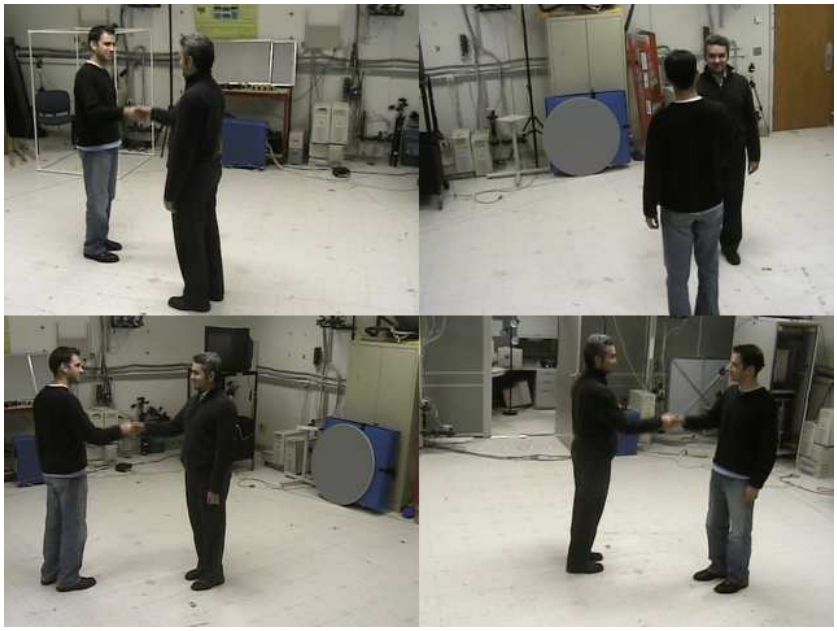


Figure 3.3: Sample input images taken by four cameras from two interacting human subjects.

3.5.1 Pre-processing Steps

Before starting the tracking process, we perform several pre-processing steps. These steps include camera calibration, background modeling and training of the color segmentation module. These steps need not be repeated as long as the cameras stay still and the scene illumination does not change significantly. For outdoor scenes, some further modules for image stabilization and dynamic background modeling would be required.

3.5.1.1 Camera Calibration

All the cameras must be calibrated with respect to a common world coordinate frame and a set of projection matrices need to be computed [83]:

$$\mathbf{P}_m = \left[\begin{array}{cccc} \mathbf{p}_m^1 & \mathbf{p}_m^2 & \mathbf{p}_m^3 & \mathbf{p}_m^4 \end{array} \right] \tag{3.20}$$

$\mathbf{p}_m^l = (p_m^{1l}, p_m^{2l}, p_m^{3l})^T$ for $l = 1, 2, 3, 4$ and $m = 1, ..., M$ define the columns of the M projection matrices with M showing the number of cameras. Knowing the matrices $\mathbf{P}_m$, the relationship between a 3D point $\mathbf{Z} = (X, Y, Z, 1)^T$ in homogenous space and its 2D projection $\mathbf{z}_m = (u_m, v_m, 1)^T$ on the m^{th} camera with images coordinates (u_m, v_m) in pixel satisfies:

$$\begin{pmatrix} u_m \\ v_m \\ 1 \end{pmatrix} = \mathbf{P}_m \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{3.21}$$

Therefore, any 2D point $\mathbf{z}_m$ can be projected into a 3D line which is defined as the intersection of two planes satisfying

$$Q_m \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = q_m \tag{3.22}$$

where

$$Q_m = \begin{pmatrix} u_m p_m^{31} - p_m^{11} & u_m p_m^{32} - p_m^{12} & u_m p_m^{33} - p_m^{13} \\ v_m p_m^{31} - p_m^{21} & v_m p_m^{32} - p_m^{22} & v_m p_m^{33} - p_m^{23} \end{pmatrix} \tag{3.23}$$

and

$$q_m = \begin{pmatrix} p_m^{14} - u_m p_m^{34} \\ p_m^{24} - v_m p_m^{44} \end{pmatrix} \tag{3.24}$$

A 3D line can be defined by a 3D vector $\mathbf{v}(v_x, v_y, v_z)$ and a 3D point $\mathbf{p}(p_x, p_y, p_z)$, therefore, finding the two planes $B_{10} + B_{11}X + B_{12}Y + B_{13}Z = 0$ and $B_{20} + B_{21}X + B_{22}Y + B_{23}Z = 0$, the 3D line $(\mathbf{v}, \mathbf{p})$ intersecting the two planes can be defined by

$$\mathbf{v} = \begin{pmatrix} B_{11} \\ B_{12} \\ B_{13} \end{pmatrix} \times \begin{pmatrix} B_{21} \\ B_{22} \\ B_{23} \end{pmatrix} \tag{3.25}$$

and

$$\mathbf{p} = \begin{pmatrix} \chi \\ \frac{B_{10}B_{23} - B_{20}B_{13} + (B_{11}B_{23} - B_{21}B_{13})\chi}{B_{22}B_{13} - B_{12}B_{23}} \\ \frac{B_{10}B_{22} - B_{20}B_{12} + (B_{11}B_{22} - B_{21}B_{12})\chi}{B_{23}B_{12} - B_{13}B_{22}} \end{pmatrix} \tag{3.26}$$

with χ being an arbitrary real value and $\times$ defining the cross product of two vectors. This 3D line will be used later in the re-projection process to find the 3D candidate points.

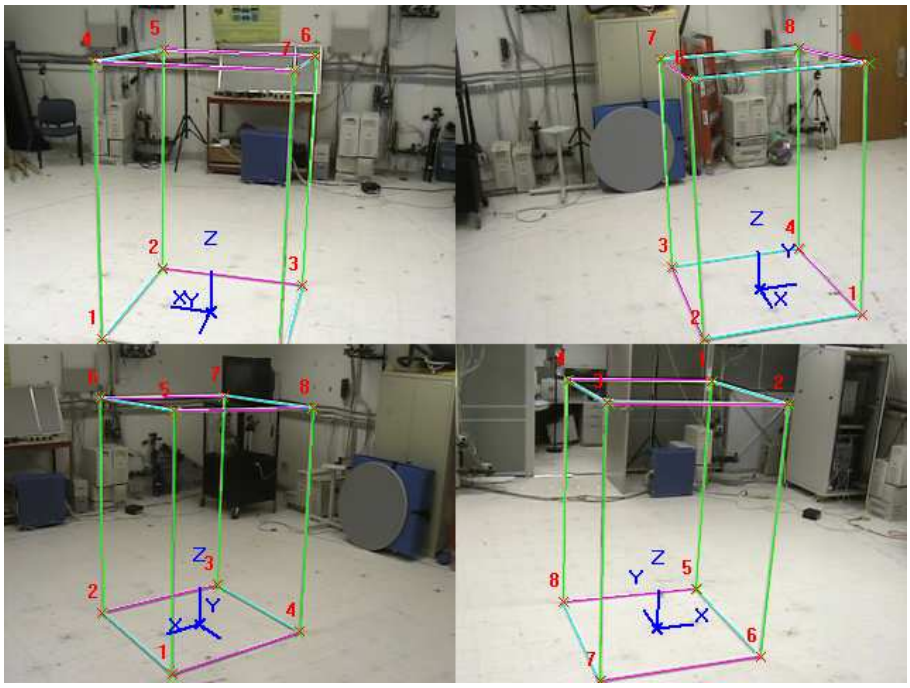


Figure 3.4: Camera calibration using vanishing points method and an accessory structure.

The important issue in the calibration process is the relative accuracy between the cameras. In fact, it is very important that the two 3D line of sights corresponding to a single 3D point and two cameras, pass nearby each other in 3D space. If some of the cameras are not well calibrated, the accuracy of the candidate points would be reduced and the number of candidate points in 3D space would increase; this would degrade tracking performance. The calibration method used is based on *vanishing points* [83, 84] and requires a set of parallel lines in the three directions X, Y and Z and also a few 2D image points with known spatial coordinates.

Figure 3.5: Background subtraction results for a sample frame.

3.5.1.2 Background Modeling

Another preprocessing step is background modeling which is needed for the background subtraction module. Even though this is not an essential step in the process, it helps in reducing the clutter and eliminating background objects with similar color to skin . Also, by removing the background from the scene, we can obtain the silhouettes of human subjects in the scene and use them as a cue in estimating the prior probability of the candidate points as explained in section 3.5.4. We deploy the *color code book* method for background modeling and subtraction [85]. Figure 3.5 shows the outputs of the background subtraction module for a sample frame after applying some *morphological* and *flood-fill* filtering.

3.5.1.3 Skin-Colored Regions Segmentation

The next pre-processing step is to train a module for segmenting the skin-colored regions from the images. For this purpose, we employ the same module as the one explained in section 2.2.1.2 [33]. Note that we may need to train a separate skin color detection module for each camera, because different cameras may have different internal parameters such as brightness, contrast, white balance and so on. In fact, in spite of the stereo camera case, we do not make any attempt to capture similar images from the cameras. The reason is that there is no need to find similar regions across the cameras and the information extracted from various cameras are aggregated only in 3D space using the 3D reconstruction process.

3.5.2 Image Observations and Accuracy Problem

Images are captured simultaneously from all the cameras at 15 fps. Each image is then processed separately and blobs are segmented. If the images are captured from a scene in which N_h human subjects are present, we ideally expect to extract $3N_h$ regions from each image corresponding to the hands and heads of the subjects; however, due to a variety of reasons, the number of extracted regions usually deviates from this ideal number:

1. Due to occlusion, some limbs are invisible in some images. The occlusion problem is especially severe when multiple people are in the scene. Also, shadows and lighting conditions might change the color of some limbs, thereby

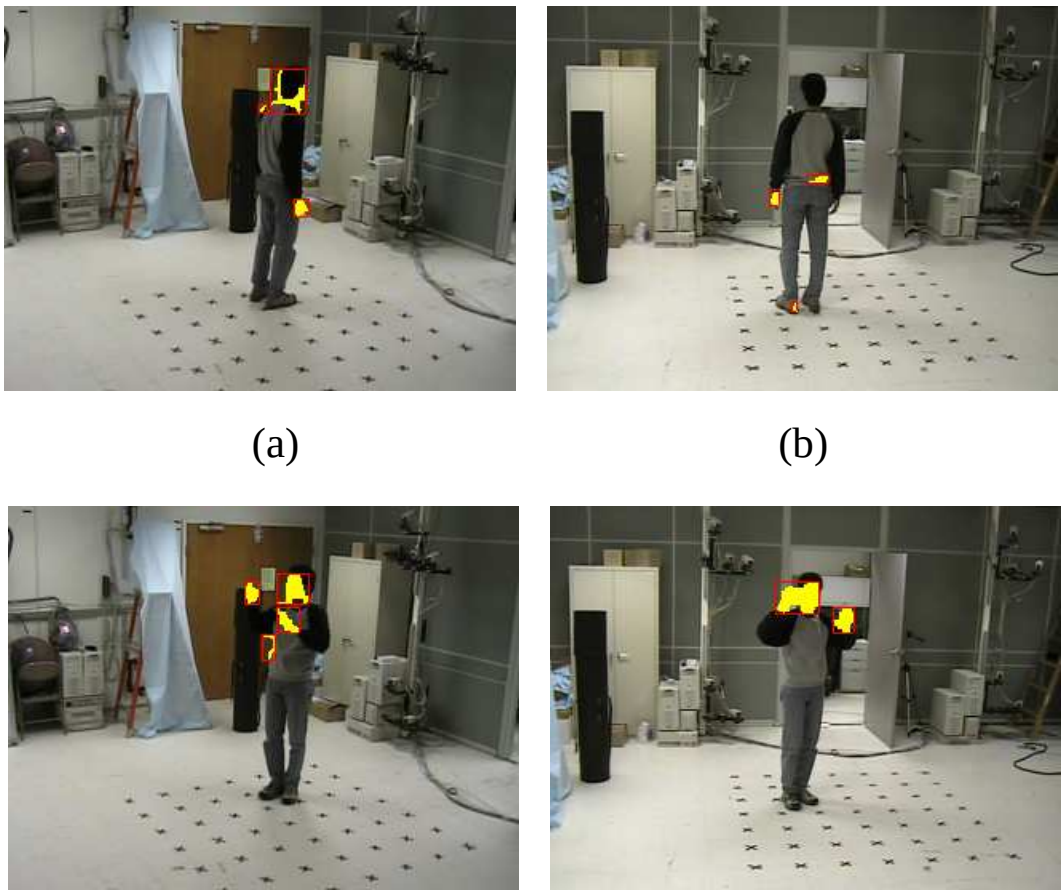


Figure 3.6: Skin-colored regions segmentation and sources of inaccuracy: (a) Occlusion and false alarm, (b) Unsegmented region and false alarm, (c) False alarm, (d) Joined regions.

leave them unsegmented in some images. These cases are shown in figures 3.6(a) and 3.6(b). In figure 3.6(a), the left hand is occluded by the body and in figure 3.6(b), the right hand is undetected due to shadow and its small size.

2. Some of the limbs may be too close to each other or reside on the same line of sight from the camera so that they join and make a single region in the image. Figure 3.6(d) shows a case where a hand and head are segmented as a single region.

3. Even though the background subtraction module filters all the skin-colored objects in the background, there is possibility that some regions other than the hands and head are segmented from the foreground area. These spurious regions may appear on any part of the person's body. Figures 3.6(a), 3.6(b) and 3.6(c) show examples of such regions. Fortunately, these undesired blobs usually appear in non-consistent regions across the images, therefore they are likely to be assigned negligible weights when re-projected to 3D space.

Due to these problems, the actual number of detected blobs is usually different from the ideal number, therefore we retain the $\lceil 3\eta N_h \rceil$ largest blobs with $\eta > 1$ to increase the change of including all desired blobs and discard the remaining segmented blobs. Finally, the blobs with sizes smaller than a certain threshold are filtered and removed. For the m^{th} camera, this leaves N_{om} blobs which will be used in the re-projection process as explained in the following section.

3.5.3 Computing 3D Candidate Points

To re-project the extracted blobs to 3D space, every blob should be represented by a single point. We select the *center of mass* of the blob as the representative point. This selection however, adds another source of inaccuracy as the center of mass of the objects in different cameras are not generally the projection of the same 3D point. This is especially true for the head, which is a larger object. A person's hair also increases the error as it may cover large portions of the head from some viewpoints, effecting the skin-color based segmentation.

As explained in section 3.5.1.1, having a calibrated camera, we find the 3D line of sight in our tracking space corresponding to any single point in the image acquired by a camera. A pair of lines corresponding to the images of a 3D point projected in two cameras are used to reconstruct the point in space. In fact, the point ideally lies on both lines and therefore on the intersection of them in space; however, due to the noise terms introduced earlier as well as digitization effects, the lines will not actually meet in space. As a result, the 3D point is approximated by the closest point in space with respect to the two lines. This point is the midpoint of a line segment which is orthogonal to both lines. Let $(\mathbf{v}_m^j, \mathbf{p}_m^j)$ and $(\mathbf{v}_{m'}^{j'}, \mathbf{p}_{m'}^{j'})$ denote two 3D lines corresponding, respectively, to the projections of the j^{th} selected point on the m^{th} camera and the j'^{th} point on the m'^{th} one and $\mathbf{v}$ and $\mathbf{p}$ be computed using equations 3.25 and 3.26. Therefore, the 3D point $\mathbf{Z}(X, Y, Z)$ can be approximated as

$$\mathbf{Z} = (\mathbf{p}_m^j + \mathbf{v}_m^j c_1 + \mathbf{p}_{m'}^{j'} + \mathbf{v}_{m'}^{j'} c_2)/2 \tag{3.27}$$

where $\mathbf{c} = (c_1, c_2)^T$ can be computed as

$$[\mathbf{v}_m^j \quad -\mathbf{v}_{m'}^{j'} \quad (\mathbf{v}_m^j \times \mathbf{v}_{m'}^{j'})]\mathbf{c} = \mathbf{p}_m^j - \mathbf{p}_{m'}^{j'} \tag{3.28}$$

Also, the distance between the 3D reconstructed point $\mathbf{Z}$ and each one of the 3D lines can be computed as

$$d_{mm'}^{jj'} = \|\mathbf{p}_m^j + \mathbf{v}_m^j c_1 - \mathbf{p}_{m'}^{j'} - \mathbf{v}_{m'}^{j'} c_2\|/2 \tag{3.29}$$

The distance $d_{mm'}^{jj'}$ is a measure of the accuracy of the 3D reconstruction process. For example, if the j^{th} selected point on the m^{th} camera and the j'^{th} point on the

m'^{th} camera do not actually represent the same 3D point in space, this error would be large, whereas for image points representing the same point, the error would be small. Also, the more accurate the image observations are, the smaller this measure would be. As a result, $d^{jj'}_{mm'}$ is included in the prior probability estimation of the 3D candidate points as explained in the next section.

3.5.4 Prior Probability Estimation for 3D Candidate Points

In this section, we discuss how to assign a prior probability to all the 3D candidate points computed in the previous section. These prior probabilities will be then used in the likelihood estimation for the particles according to equation (3.17). Let $Z^c = R(z^j_m, z^{j'}_{m'})$ denote the re-projection (reconstruction) of the c^{th} 3D candidate point being observed as the two image points z^j_m and $z^{j'}_{m'}$ in the m^{th} and m'^{th} cameras respectively. According to equation (3.19), the prior probability for Z^c is comprised of three terms corresponding to the image space, 3D reconstruction and its back projection to the image space. All these terms are defined as zero-mean Gaussain functions with some error terms as the variables and tune the variances based on a set of training videos.

The first term, which evaluates the compatibility of a pair of observations, is based on a function $D_v(z^j_m)$ which measures the relative v coordinate of an image point with respect to the range of v coordinates of the silhouette of the person in the image. The prior probability term $p^o(Z^c)$ is defined as

$$p^o(Z^c) = exp(\frac{-(D_v(z^j_m) - D_v(z^{j'}_{m'}))^2}{2\sigma^2_v}) \tag{3.30}$$

based on the fact that the relative v coordinates of z_m^j and $z_{m'}^{j'}$ should be close if the two points correspond to the same 3D candidate point. Note that this relies on the assumption that all the cameras are located such that 3D points with larger Z coordinates will have smaller v coordinates in all the images. Different configuration of the cameras requires modifying this term.

The second term, $p^t(Z^c)$ measures the accuracy of the 3D re-projection and is a function of the distance $d_{mm'}^{jj'}$ computed in equation (3.29). $p^t(Z^c)$ is defined as

$$p^t(Z^c) = exp(\frac{-(d_{mm'}^{jj'})^2}{2\sigma_d^2}) \tag{3.31}$$

which decays exponentially as the re-projection error $d_{mm'}^{jj'}$ grows. Even though $p^t(Z^c)$ seems a reasonable means to rule out invalid points; cases remain where the image points corresponding to two different objects in the scene reconstruct a candidate point with small distance error $d_{mm'}^{jj'}$. For example, when two limbs lie along the same line of sight from the perspective of one of the cameras. To detect such points, we rely on the other two terms $p^o(Z^c)$ and $p^b(Z^c)$ which are based on information in the image domain.

The last prior probability term, $p^b(Z^c)$ is a function of $D_s(z'^c_m)$, the distance of z'^c_m , the projection of the candidate point Z^c on the m^{th} camera image to the silhouette of the person segmented by the background subtraction module and is defined as

$$p^b(Z^c) = \prod_{m=1}^{M} exp(\frac{-(D_s(z'^c_m))^2}{2\sigma_s^2}) \tag{3.32}$$

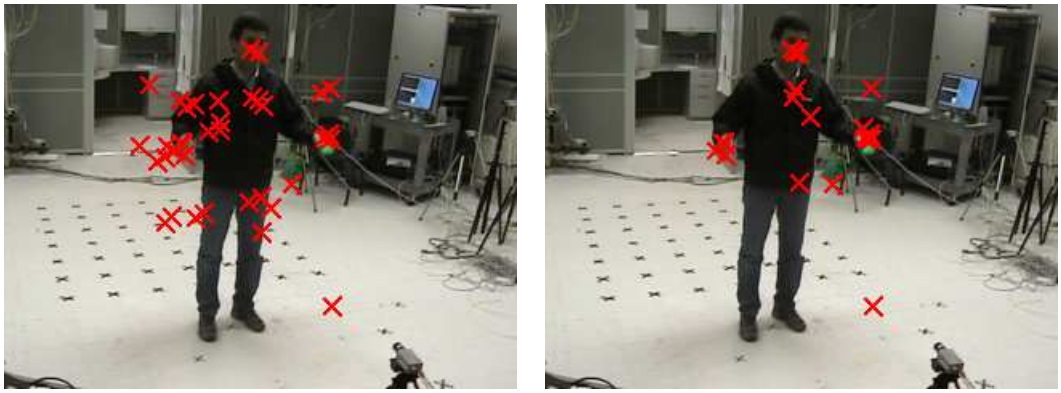
This term is especially useful to identify those candidate points which relate two inconsistent image points, yet make a small distance error in 3D space.

Figure 3.7(a) shows all the 3D candidate points projected into one of the camera images for a sample frame. As can be noticed, the number of candidate points is more than desired. The reason is that the combination of all pairs of points are considered. One way to filter most of these points is through thresholding the prior probabilities. For example, figures 3.7(b),(c),(d) show the points which pass a threshold of 0.1 for $p^o(Z^c)$, $p^t(Z^c)$ and $p^b(Z^c)$ respectively. Figure 3.7(e) also show the points with final prior probability $p(Z^c)$ larger than 0.1.

Since the re-projection process only considers *pairs* of image points and there are usually more than two cameras involved in the tracking, it is very common to have several candidate points representing a target. Close candidate points are combined to save redundant computations by eliminating the weaker point and boost the prior probability of the other one. The rate of boosting is a fraction of the prior probability of the eliminated point. Figure 3.18(f) shows the final three candidate points after filtering the points based on the prior probabilities and joining close points. It is worth noting that in spite of the effectiveness of the prior probability estimation, some spurious 3D candidate points may appear in some of the frames; however, the particle filtering method and the history it creates for the trajectories is usually robust enough to avoid the confusion caused by these points.

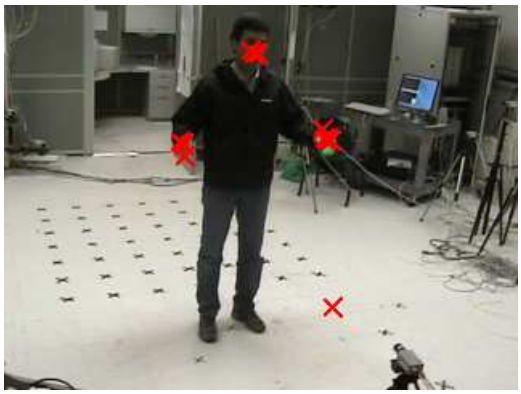
3.6 Application: Activity Recognition for Visual Surveillance

The proposed hand/head tracking system can be applied to visual surveillance and human-computer interaction. In visual surveillance applications, we are usually

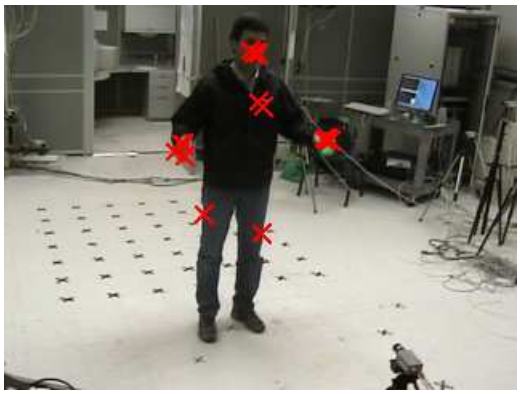


(a)

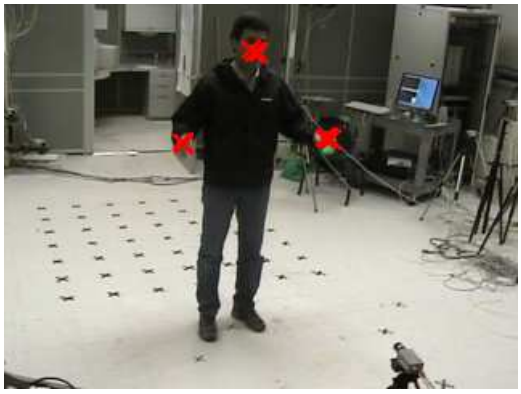
(b)



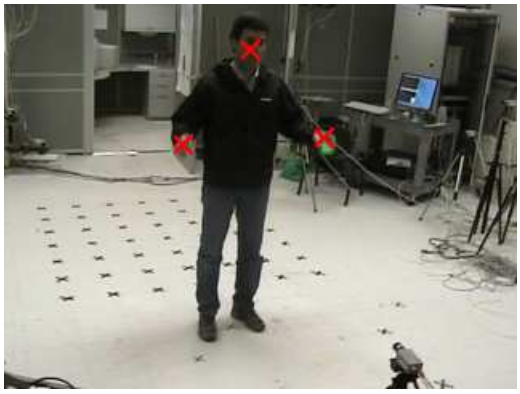
(c)



(d)



(e)



(f)

Figure 3.7: 3D candidate points projected back to the image: (a) All the candidate points, (b) Candidate points passing the $p^o(Z^c)$ threshold, (c) Candidate points passing the $p^t(Z^c)$ threshold, (d) Candidate points passing the $p^b(Z^c)$ threshold, (e) Candidate points passing the $p(Z^c)$ threshold, (f) Candidate points after filtering by $p(Z^c)$ and joining close ones.

interested in recognizing the type of activities happening in the scene. Even though solving this problem in general is an extremely difficult problem, in this section it is shown how to use the information generated by the system to recognize and classify a certain class of activities. Our goal is to classify activities merely by knowing the trajectory of the hand and head motions throughout the sequence, estimated using our tracking method.

There are two classes of activities we are interested in: the first class includes activities which involve only the motions of the hands and/or the head and interactions between them. Examples of these activities are clapping and hand shaking and are discussed in section 3.6.1. The second class of activities involves carrying objects with the hand. Examples are object exchanges between hands of a person or two people, picking up an object from the scene, and placing an object in the scene and those discussed in section 3.6.2. For the activities of this class, we need to detect at certain time instants during the act whether the person holds an object or not.

3.6.1 Activity Classification based on Limb Interactions

To classify the activities which involve interaction between the limbs, we detect the moments two or more limbs coincide or separate as well as the type of limbs involved in that coincidence. This information is provided by the proposed tracking system. Coincidence of two limbs happens when the trackers associated with them make the transition from the normal state to the combined state (see section 3.4.2).

Also, they separate when the trackers make the transition from the combined state to the unlabelled state. In the unlabelled state, the problem is how to label the targets so they can return to the normal state. As explained in section 3.4.2, some application based cues are required to label the targets in the unlabelled state. All the activities considered in this section share the property that the coincident limbs subsequently separate from each other. Therefore, by computing the angle between the two 3D vectors connecting the two targets before joining and after separation, the targets can be labelled.

Knowing the type of the interacting limbs, we can limit the list of possible activities. For example, if two hands of the same person join and separate, this is a cue for clapping or object exchange between the hands. Also, if two hands of two different people join and separate, it could be a hand shake or object exchange. To choose the right activity from these alternatives, we detect if any of the hands holds an object and whether this object moves from one hand to another after the separation.

3.6.2 Carrying Object Detection

The proposed technique for detecting whether the person carries an object or not is through color analysis of the pixels around the hand region. The detection steps are as follows:

1. As a first step, it is determined for each hand whether it is visible in a camera or not. the 3D estimated location of the hand to each camera is back projected

and verified whether the projected point resides within the boundary of any segmented skin-colored blob in the image. If a blob is detected, it is considered as the region of the desired hand.

2. For each visible hand in an image, an appropriate region around the corresponding blob is selected for color analysis. The color histogram of the pixels in that selected region is constructed and normalized. The modes of the color histogram are extracted and, for each mode, it is verified whether it corresponds to a connected region in the image. If such a region is found, the mode is retained, otherwise it is eliminated.

3. The modes of all the histograms acquired at different times by different cameras are combined and a *weighted average distribution* is constructed. The weight of each histogram depends on the quality of the selected region and is determined base on the relative position of the hand with respect to the body.

4. The weighted average distributions of different hands are compared at two different time instants, called *decision points*, to detect the carried object, its color and the potential exchange. The first time instant is when the hands coincide with each other and the possible object exchange occurs. The second time instant is when the hands separate above a minimum pre-defined distance. This usually creates a few frames of delay which is required to collect enough data for updating the color histogram of the hands, thereby identifying whether an exchange has occurred or not.

   The following subsections explain these steps in detail.

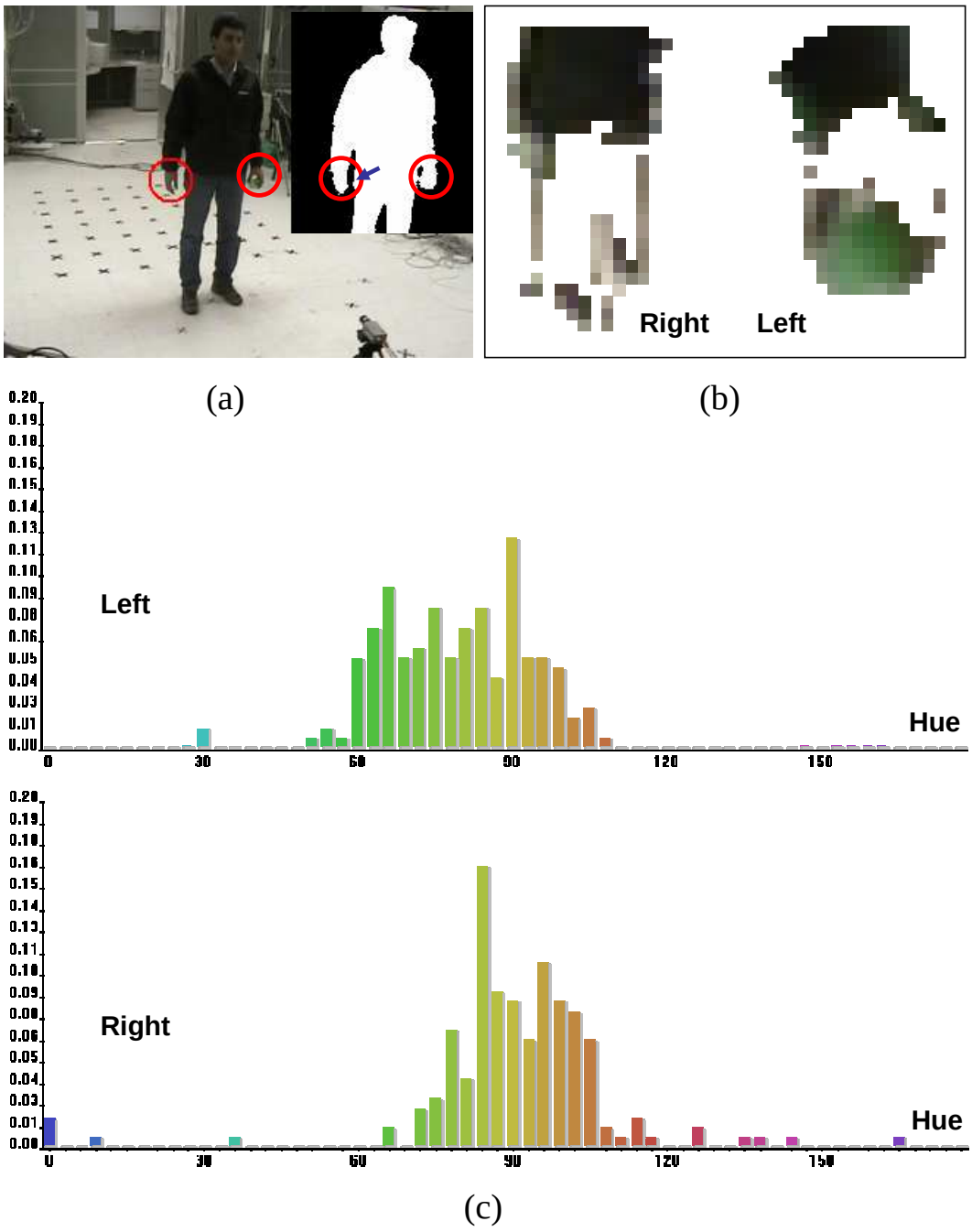


Figure 3.8: The selected regions for color analysis: (a) Input image and silhouette and the selected circular regions, (b) The selected regions after filtering, (c) Normalized color histogram of the two selected regions..

3.6.2.1 Hand Region Selection

At each image acquired by the cameras, if a skin-colored blob corresponds to a tracked hand, a circular region around that blob is selected to be used in the color analysis. These circular regions are then filtered using the silhouette of the body generated using the background subtraction module. Also, if the area inside the circle contains more than one connected foreground region, the region containing the skin-colored blob is preserved and the others are eliminated. The skin-colored blob is also excluded from the region and the remaining pixels presumably correspond to an object, the sleeve or dress of the person (depending on the relative position of the hand to the body) and noise pixels. Next, the color histogram of the remaining region is generated and analyzed. The *hue* is used as the color indicator.

Figure 3.8(a), depicts the regions selected for the two hands in a sample frame image in which the person holds a small green object in his left hand. The region of the pant which is pointed to by arrow is eliminated as it is not connected to the hand. Figure 3.8(b) shows the filtered version of the two selected regions in 3.8(a). Figure 3.8(c) shows the normalized color histogram of the two selected regions. As can be seen, the histogram of the left hand shows large values around the green color (e.g. $hue \approx 65°$). Also note that although the detected skin-colored region is filtered, the amount of skin-colored values in the histogram is still significant due to failure of the color detection module.

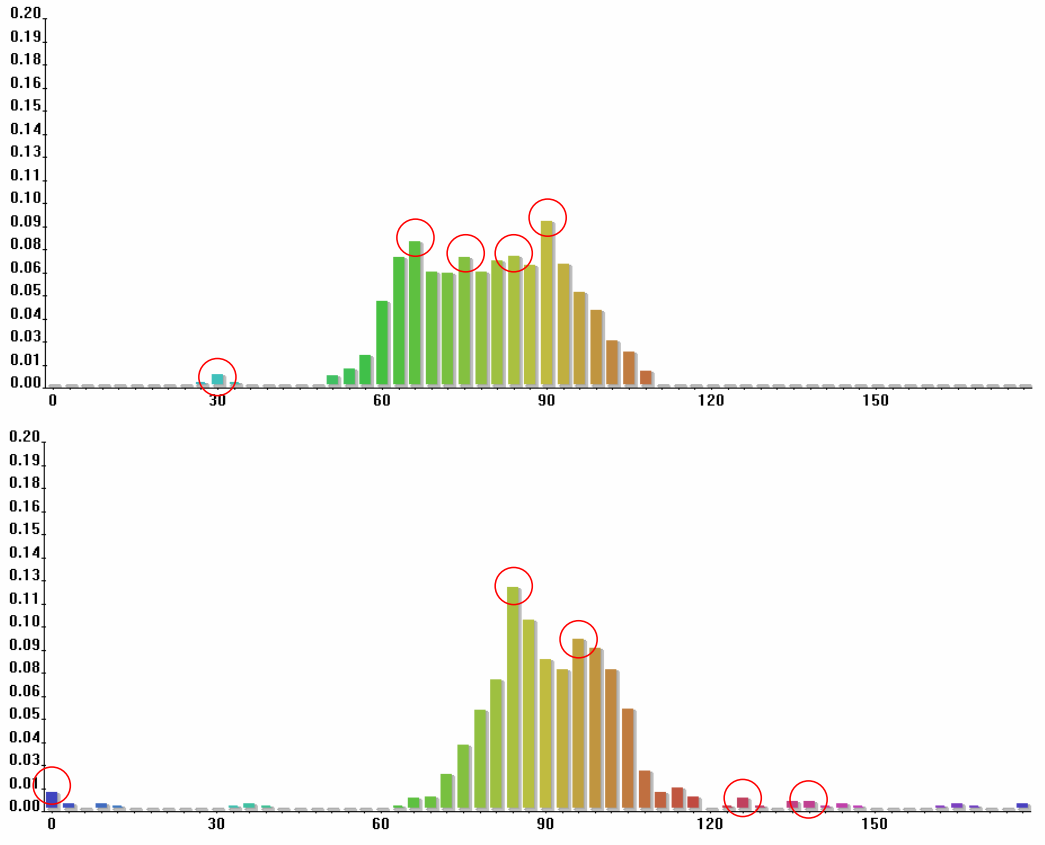


Figure 3.9: The normalized color histogram of the selected regions in figure 3.8 after kernel density estimation and the location of the modes.

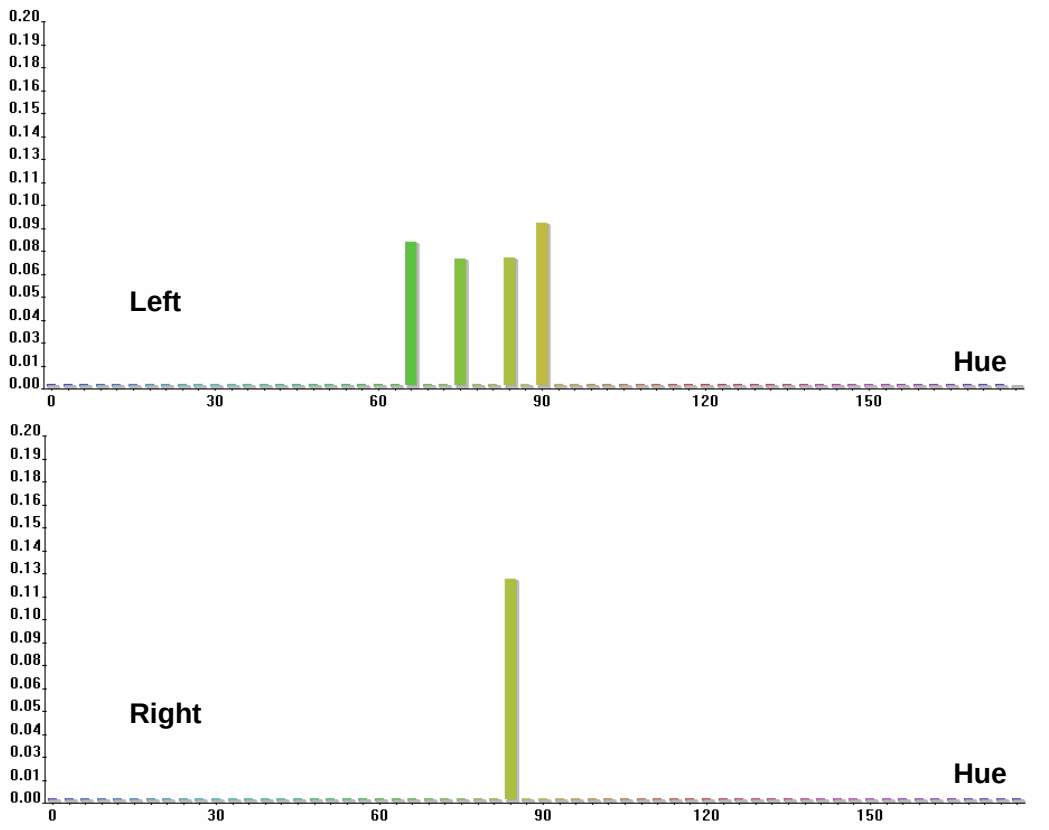


Figure 3.10: The modes of the histogram in figure 3.9 after connectivity verification.

3.6.2.2 Color Histogram Modes Extraction

To detect the modes of the distribution, *kernel density estimation* with a gaussian kernel is applied to the histogram and local maxima are selected. Several modes may be found in the histogram; however most of these modes appear due to scattered noise. To remove these, the modes of the histogram are back projected to the image to determine whether they form a connected region or not. If a connected region of minimum n pixels (e.g. $n = 10$) is found in which all the pixels have the hue value in the range $[hue - \alpha, hue + \alpha]$ with $\alpha \approx 3°$, then the mode hue is retained otherwise it is eliminated. It is worth noting that after all the filtering, there may still be modes which appear due to noise. These modes will be filtered by the subsequent step. Figure 3.9 shows the histograms of figure 3.8(c) and the position of the modes after kernel density estimation. Figure 3.10 shows the retained modes after the connectivity verification process.

3.6.3 Estimating the Relative Hand Position

The modes of the color histograms extracted from the individual cameras are combined to increase the reliability of the process. However, at each time instant, some cameras are located in a better position with respect to the person and collect more reliable data. By detecting those cameras, a larger weights can be assigned to them in the weighted average distribution.

A selected region around a hand is considered more reliable when the number of pixels not belonging to potential object is minimal. This occurs when the hand

is stretched out from the body and the camera views the body from an appropriate angle. To determine the reliability of a selected region, the pixels of the body silhouette which lie on the bounding circle (see figure 3.11) are extracted and a n-tuple vector $\mathbf{v}_c$ of 0's and 1's is created. Values 0 and 1 correspond to foreground and background pixels respectively. The reliability factor μ is then defined as

$$\mu = \frac{n - \sum_{i=1}^{n} v_{ci}}{n} + \mu_{min} \tag{3.33}$$

where μ_{min} is a constant positive number to be used as the minimum reliability factor. The larger the value μ is, the more reliable the selected region would be. For instance, in the case of figure 3.11(a), the majority of the undesired pixels belong to the sleeve of the person's dress which does not constitute a big portion of the region. Also, knowing the location of the sleeve from the 1's in the vector $\mathbf{v}_c$, we are able to remove them. As a result, the selected region in figure 3.11(a) can be considered a reliable selection. This consideration is reflected in the large value of μ. In contrast, in the case of figure 3.11(c) with the hand residing inside the silhouette of the body, many pixels belonging to the person's dress may be contained in the region inside the circle. This reduces the utility of the color analysis, therefore, a small value μ is assigned to this region.

Another point to note is that an iterative process can be added to determine the appropriate radius of the circle. In fact, we can start from a small circle, which merely covers the skin-colored blob and grow the circle gradually until a sufficiently small or large value for μ is measured or the circle reaches a pre-defined maximum

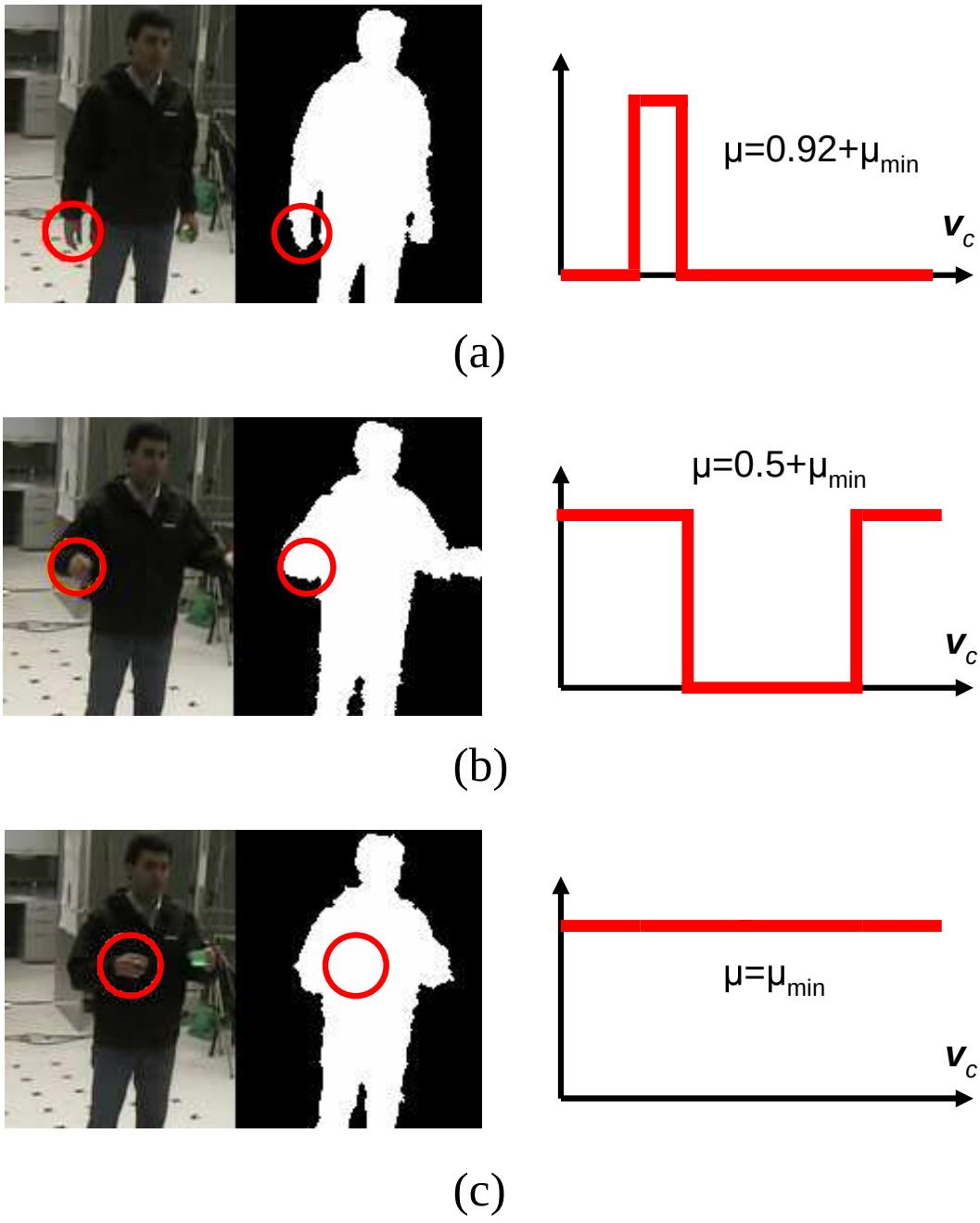


Figure 3.11: Reliability measurement of the selected regions: (a) Hand outside the body, (b) Hand in in-out position (c) Hand inside the body.

size. When the value μ is determined, the circle size is modified accordingly; for example, if μ is very small (the hand is inside the silhouette), the circle size is modified to the smallest possible size to exclude as many pixels of the person's dress as possible.

3.6.3.1 Object Exchange Detection

To detect the exchange of an object between two hands or gaining or ceasing to carry an object in case of pick and place actions, the weighted average color histograms are analyzed at two different time instants called the *decision points*. The first time instant called the *climax point* is when two hands coincide or when a hand stops moving for a few moments to pick up or place an object. The exchange moment can be detected using the FSM used for tracking. To recognize the climax point of the pick and place activities, which are categorized as *ballistic movements* [86], motion field estimation is deployed. The second decision point in which the histograms need to be examined is when the hands are separated by a minimum pre-defined distance from each other. This is usually a few frames after separation of the two hands or after the still moment for pick and place cases. From the modes of the averaged color histograms of the hands, it is determined whether the hand holds an object or not. Also, comparing the modes of the hands at the two decision points, we can recognize whether the objects has been transferred from a hand to another or not. This information can consequently lead us to the recognition of the type of activity. Table 3.1 shows this categorization for a set of activities involving

the interaction of two hands. For example, if there is no object in either of the two hands (indicated by 0 in the table) before of after the climax point, the activity is either a hand shake or a clap depending on the number of involved people. Or, if an object is detected in one hand before the climax point and in another one afterwards, an exchange has happened. If there is only one hand involved in the activity, the presence of an object in the hand before or after the climax point distinguishes the two pick and place activities.

First Hand (Before)	Second Hand (Before)	First Hand (After)	Second Hand (After)	Activity Type
0	0	0	0	Hand Shake or Clap
0	1	1	0	Exchange
1	0	0	1	Exchange
0	-	1	-	Pick
1	-	0	-	Place

Table 3.1: Activity Recognition based on the Object Location Before and After the Climax Point

To determine whether a hand holds an object or not, the modes of the color histograms detected in section 3.6.2.2 from different images are combined. This is useful due to the observation that the hand or the carried object may be invisible or partially visible in one camera or another. The fusion method used here is the *Beamforming* method [87] which asserts that if a signal is transmitted over several noisy channels, the optimal way of combining the outputs of the channels is computing the weighted average of all the channels such that the weight of each channel is inversely proportional to the variance of noise of the channel. If $p^j_{mk}(c)$ denotes the pdf of the modes of the color distribution for the selected region around the j^{th} limb on the m^{th} camera at time instant k, a weighted average pdf $\overline{p}^j(c)$ is

defined as

$$\overline{p}^{j}(c) = \frac{\sum_{k=K}^{K'} \sum_{m=1}^{M} \mu_{mk}^{j} p_{mk}^{j}(c)}{\sum_{k=K}^{K'} \sum_{m=1}^{M} \mu_{mk}^{j}} \tag{3.34}$$

with μ_{mk}^{j} denoting the reliability measure μ for the j^{th} limb as computed in section 3.6.3. The reliability factor μ_{mk}^{j} can be considered inversely proportional to the noise of the channel (i.e. camera). The time period $k = [K, K']$ is a period before the decision point. Also the period begins after or ends prior to the instant the circular regions of the two limbs intersect. This avoids the case in which the object appears in both selected regions. Another possible enhancement is to attenuate the noise level of the distribution by subtracting the distributions of the two hands of the same person from each other and removing the negative distribution components. This can be especially effective to remove the color of the dress from the histogram. Performing the above process on the two interacting hands at the two decision points and comparing the modes of the distributions, it can be determined if there has been any object in a hand and whether it has changed its position between the hands. An example is shown in the experimental results section.

3.7 Experimental Results

The performance of the tracking method was measured with a set of video sequences including one person or two interacting people. Our system processes around 4 frames per second on a 3GHz PC. To measure the accuracy of the particle filtering technique deployed for tracking, the estimated 3D coordinates of the two

hands and the head of a person were compared with ground truth values. The ground truth measurements were obtained by selecting the locations of the limbs manually in all the images of sample frames and projecting them to 3D space. Figures 3.12 and 3.13 compare the X, Y and Z coordinates of the 3D limbs obtained by the two methods in two different video sequences. As seen in the graphs, the tracking error is always less than $100mm$. The two sequences used for this evaluation show a person moving hands and head in space while standing or sitting behind a desk. The average distance error in the standing sequence for the head, left and right hands are 52.5, 37.4 and 43.9 millimeters respectively. For the sequence with a person doing normal activities while sitting at a desk, the average distance error for the head, left and right hands are 45.0, 37.9 and 33.9 millimeters respectively. The larger errors for the head are expected as the head is a larger object and is often partially occluded by the hair from certain directions. Figures 3.14 and 3.15 show a few selected frames from the two video sequences. The analysis of the desk sequence shows that the moment the person picks up a cup to drink can be detected by color analysis of the frames before and after that moment.

In another experiment, the two hands of a person were tracked when the person clapped. Utilizing the finite state machine and heuristics for labelling the targets after separation, the system successfully retains the tracks and distinguishes the two hands before and after the clapping point. Figure 3.16 shows three sample frames before, during and after the clapping.

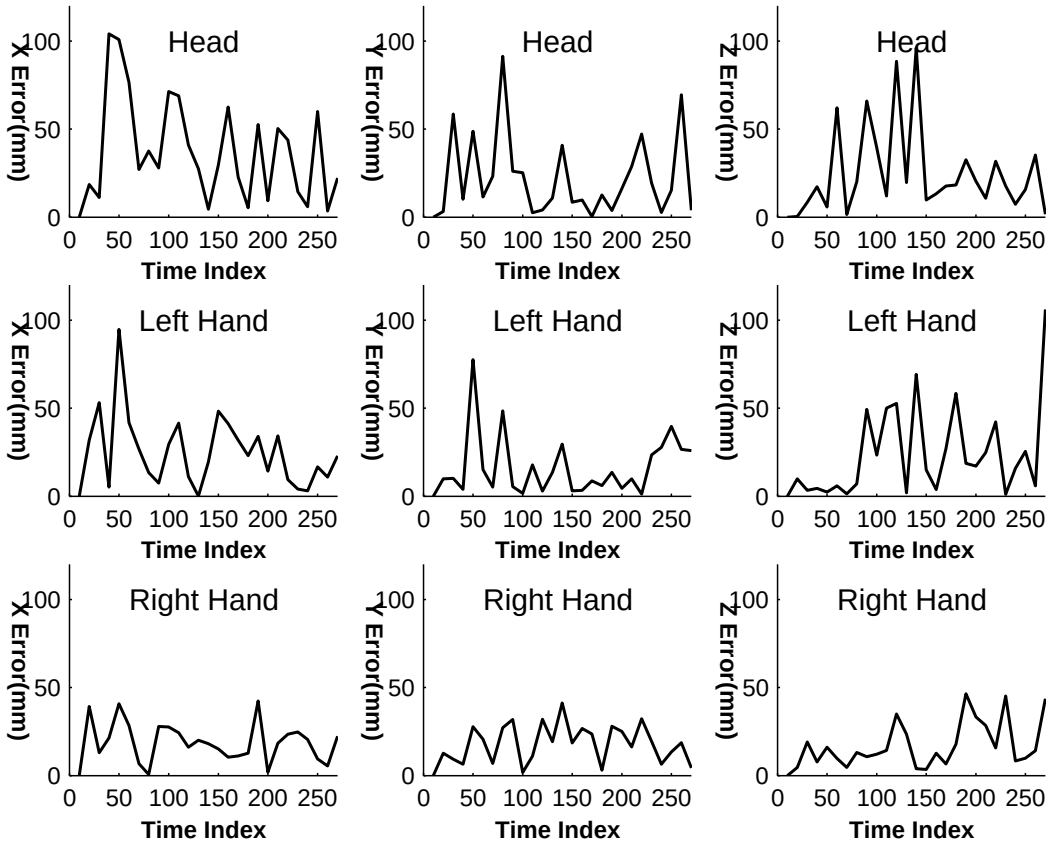


Figure 3.12: Performance evaluation of the tracking system. The tracking error is shown as the difference between the ground truth values and the actual estimated values.

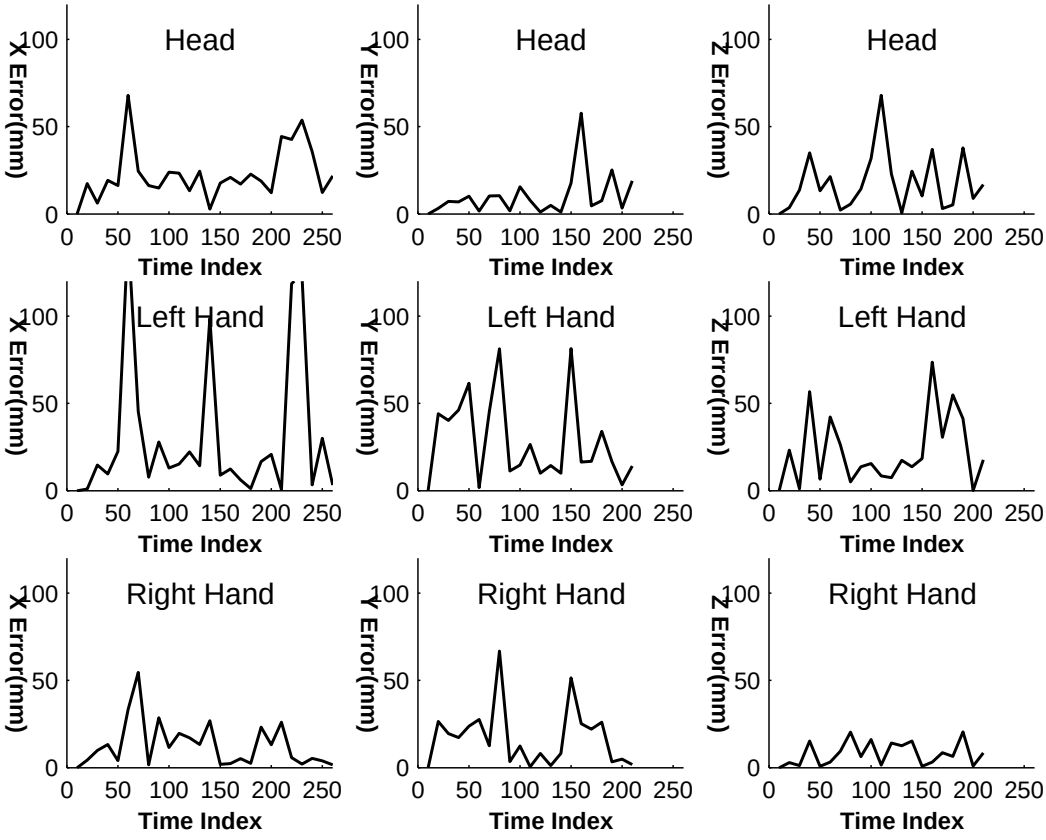


Figure 3.13: Performance evaluation of the tracking system. The tracking error is shown as the difference between the ground truth values and the actual estimated values.

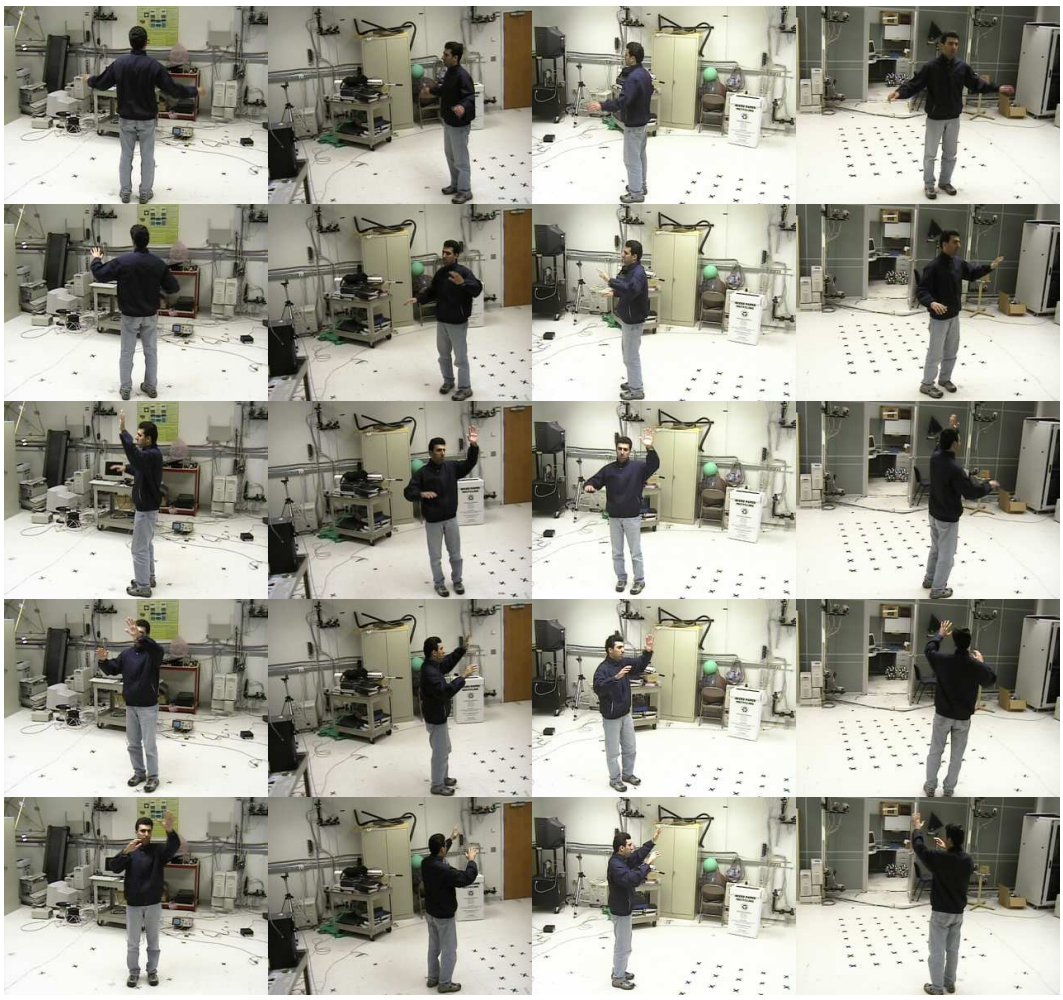


Figure 3.14: Sample frames of a sample sequence.

Figure 3.15: Sample frames of a sample sequence with a person working at a desk.

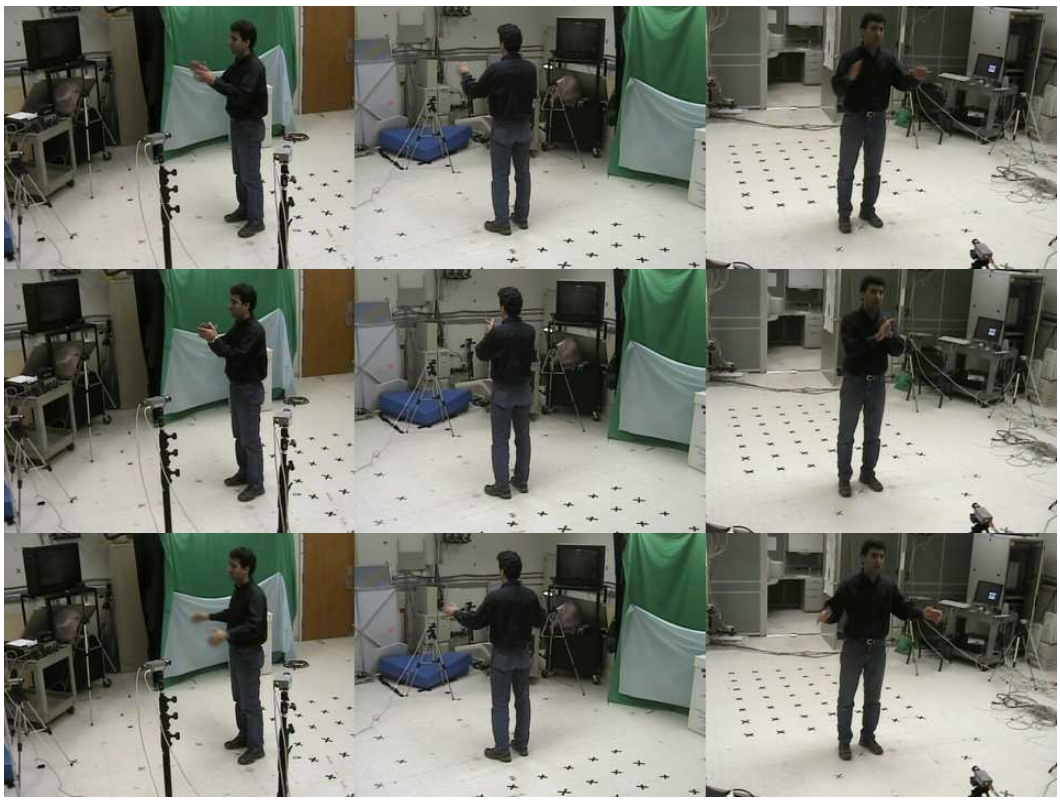


Figure 3.16: Sample frames of the hand clapping sequence.

Several videos with two interacting people were also evaluated. One expected problem in this type of scene is the high rate of occlusion. The occlusion becomes worse when the two subjects approach each other. Increasing the number of cameras in the scene and uniformly distributing them around the room can reduce the occlusion significantly; however these improvements will also increase the required computation. Also, having six targets in the scene instead of three (four hands and two heads) increases the number of candidate points in the tracking space, thereby increasing the number of false alarms. However, our system worked well in most cases. The only inevitable problem is the occasional track loss due to a shortage of observations. Figure 3.17 shows sample frames of the video of the subjects shaking hands. The colored markers show the location of the tracked limbs back-projected to the images. As depicted in the last row, the two shaking hands were distinguished and labelled successfully after separation. Meanwhile, as can be seen, the left hand of a person is lost in the 2nd and 3rd rows due to the shortage of observations. Table 3.2 shows the states of the tracked limbs at each row of the figure 3.17. The letters H, L and R stand for head and left and right hands respectively. The indices are used to indicate the two subjects.

Row	Normal	Combined	Lost
Top	L_1, R_1, L_2, R_2, H_1, H_2	-	-
Middle	L_1, H_1, H_2	R_1, R_2	L_2
Bottom	L_1, R_1, L_2, R_2, H_1, H_2	-	L_2

Table 3.2: The state of the hands and heads in different rows of figure 3.17: H, L and R stand for head and left and right hands respectively. The indices are used to indicate the two subjects.

To test the performance of the proposed object exchange detection scheme, a

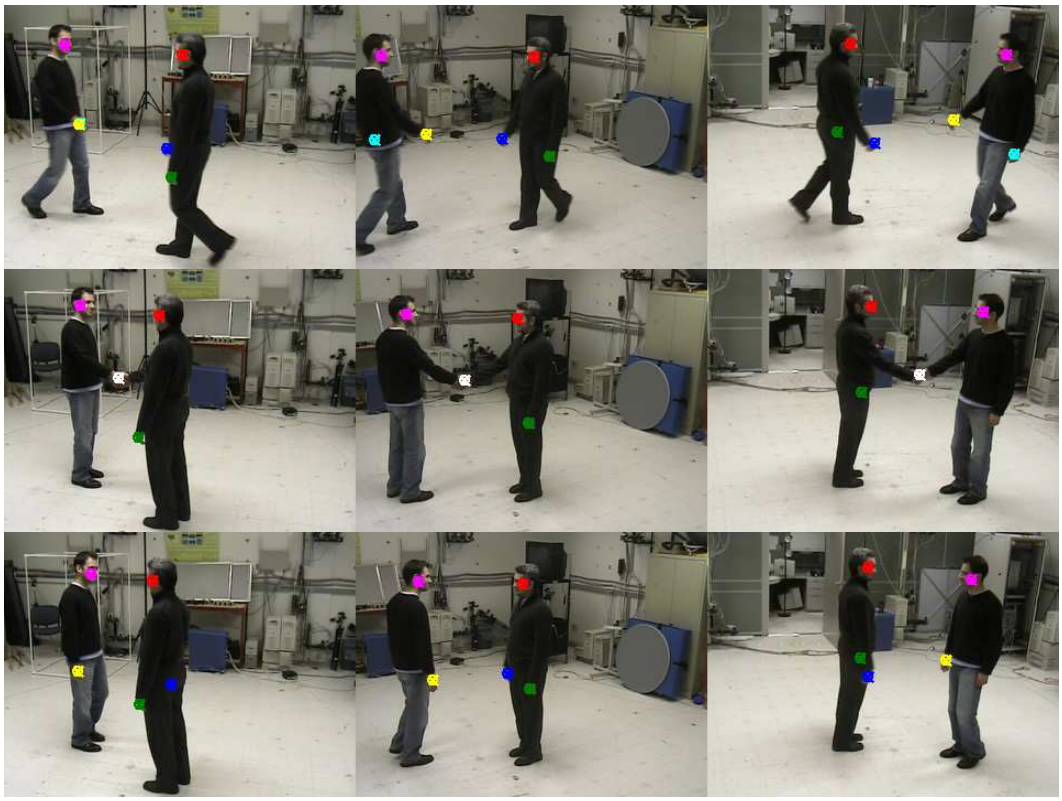
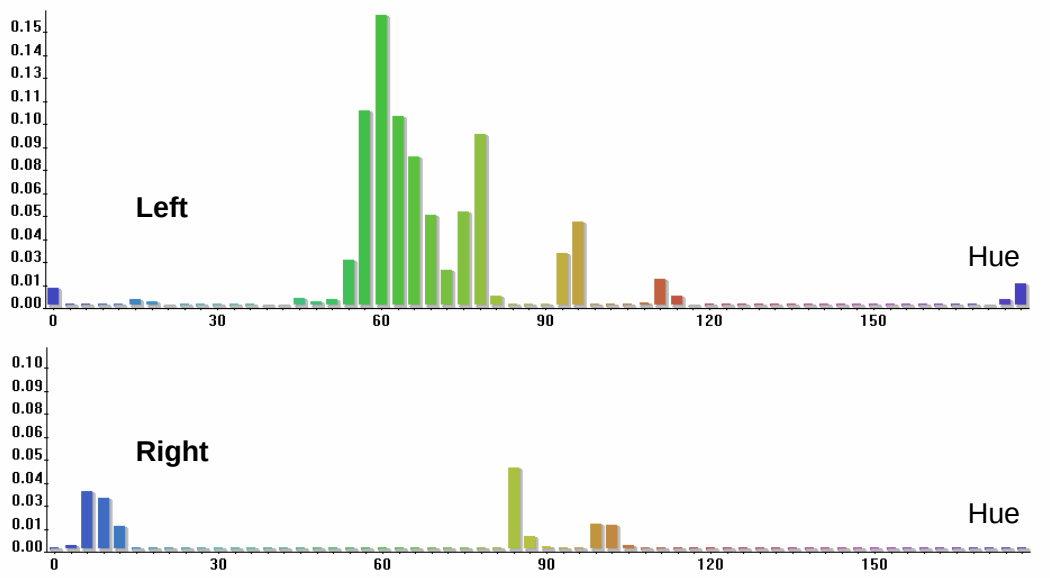
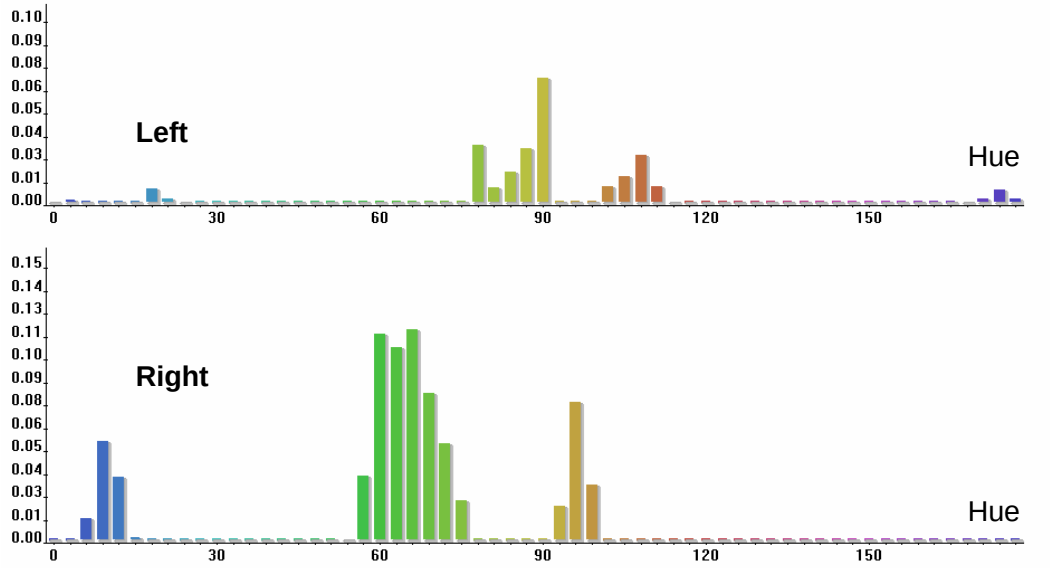


Figure 3.17: Sample frames of two people do the hand shaking. The white markers show the estimated location of the limbs. Left hand of a person is lost in the 2nd and 3rd rows due to shortage of observations



(a)



(b)

Figure 3.18: Weighted average distributions at two decision Points: (a) At the climax point, (b) A few frames after after the climax point.

sequence was examined in which a person exchanged a small green object between the two hands. Figure 3.18 shows the $\overline{p}^j(c)$ for the two hands at two time instants. Figure 3.18(a) shows the average histograms for the two hands at the moment they join. Figure 3.18(b) shows the same histograms a few frames after the hands separate. By looking at this figure, it is clear that the green object (e.g. $hue \approx 60°$) has been moved from the left hand to the right hand at the time the hands met. This can be inferred easily using analysis of the modes of the four distributions. Figure 3.19 shows several frames of the same video sequence. The average histograms of 3.18(a) and 3.18(b) are measured at the frames of the 3rd and 5th rows.

3.8 Summary

In this chapter, a general approach for multiple target tracking using multiple sensors was presented. The approach includes a low-level particle filtering layer for tracking individual targets and a high-level finite state machine for analyzing the interaction between the targets as well as appearance and disappearance of each target. A multiple hand/head tracking system was then designed based on this two-level tracking method. Some application specific heuristics were added to improve the performance of the system. An object detection scheme was also presented based on the color analysis of regions around the hands. It was shown that fusion of the information acquired by all the cameras improves the quality of object detection. As an application, a set of activities involving interaction between people and objects were investigated.

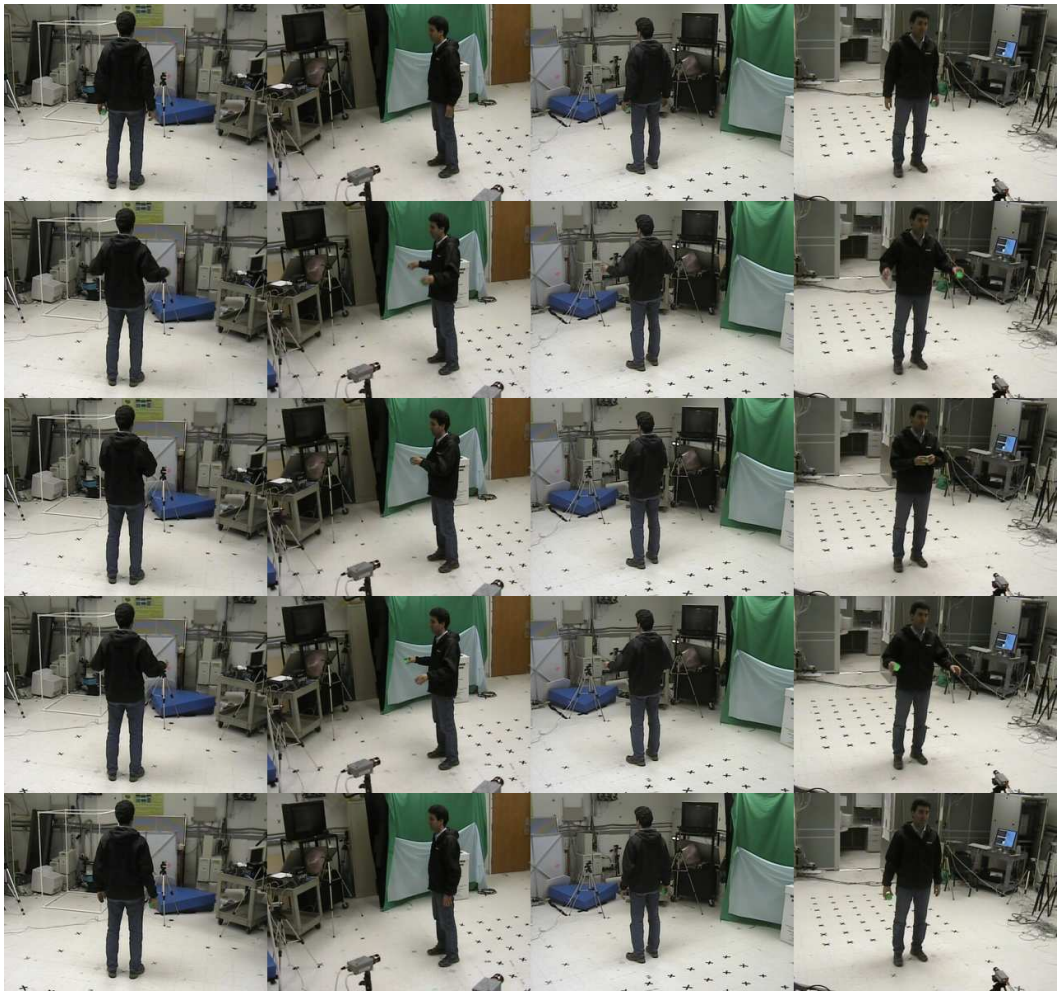


Figure 3.19: Sample frames of the object exchange sequence.

The set of experiments performed show the potential and also limitations of the approach. In fact, the proposed system can be effective in tracking the hands and heads of people and detecting the objects they may carry. This can be very helpful in visual surveillance applications; however, adding some modeling and detection modules focusing on the body structure of the people can increase the quality of the system.

Chapter 4

Conclusion

In the previous two chapters, we addressed two different problems in the area of human body tracking. Both chapters discussed hand tracking while in chapter 3, head tracking was also added.

In Chapter 2, the absolute position and orientation of a hand were estimated while the hand was viewed by a stereo camera set. Theses parameters were extracted from a planar model created based on disparity map. It was then shown how modeling the motion components of the hand can be used to fix a 3D coordinate frame on the hand thereby tracking the hand over time. It was explained that these modelings are critical for hand tracking as the hand is a low-textured object and classical disparity and motion estimation methods are very inaccurate.

In chapter 3, a two-layer tracking method was proposed for the problem of tracking hands and heads of one or several human subjects while they are viewed by multiple cameras. The novel approach included a set of parallel particle filters for tracking the desired limbs and a finite state machine to model the interaction between the limbs. Also, a color analysis method for detecting objects carried by the hands and their exchange between the hands was presented and a novel approach for combining the color information gathered by different cameras was proposed. The color combination method was based on the reliability factor of each camera

and the position of the hand with respect to the body.

As explained in this dissertation, visual tracking of the human body has two major sets of applications. The first set includes human-computer interaction in which a person or a group of people move in view of a single or a set of cameras with the goal of communicating some commands to a computer system. Several applications have been proposed and developed on this subject. In chapter 2, three sample applications in this area were described. In these applications, the 3D position and orientation of a hand were estimated and used. The first one introduced virtual drawing in space in which a user communicates letters and drawings in the same way as when she draws them on a piece of paper. This application eliminates the need for using a limited vocabulary as required in similar tools such as sign language. It shows how a vision-based system can allow people to use a system in a natural manner. In fact, one of the goals in human-computer interaction systems is to comply with the way people prefer to interact with machines and this application tried to fulfill that goal.

By monitoring and interpreting the actions of a user, a system can create the possibility of working with virtual devices instead of real ones. This adds flexibility in a variety of aspects. The marble game application presented in chapter 2 demonstrates some of these flexibilities. It enables the user to play with a new maze and a new map every time she plays with the virtual game. Also, changing the coefficient of friction shows how vision-enabled virtual devices can provide features which are not easily made available with physical devices.

Finally, we presented a 3D object construction application in which a user can

track the edges of a physical or virtual object and create its 3D model.

One important point to note is that almost all the human-computer interaction systems assume some degree of cooperation from the user. In the applications presented in this dissertation, if the motion of the hand is beyond certain limits, the system cannot track it reliably. In fact, as the system models the palm or the back of the hand as a single plane and estimates its position and orientation, if the hand is too tilted and a sufficient portion of the hand is not visible, modeling fails and the system does not respond appropriately.

Visual surveillance is another application of human body tracking which was addressed in this dissertation. In this application, the movements of people in front of the cameras are measured to categorize their activities. Chapter 3 showed how the hands and heads of human subjects can be tracked and used for activity classification. Using multiple cameras made the system more robust and effective, allowing us to measure human movements over a broad range of viewing conditions. Also, 3D tracking helps in classifying the actions more accurately. However, action classification only based on the movements of hands and heads is not sufficient - additional analysis is required. For example, by detecting the presence of an object in the hand, an action like hand clapping can be distinguished from similar hand movements associated with an object exchange between hands. Information about limbs of the body can expand the range of activities that can be classified.

4.1 Future Work

As future work on the topics addressed in chapter 2, the following directions may be taken:

1. Modeling the hand with more complex surfaces than a plane. This can result in more accurate estimation of the position and orientation of the hand, which is especially helpful for cases in which the hand makes complicated shapes and gestures. For instance, when the hand is closed as a fist, a quadratic surface can model its surface more accurately. Also, if the hand is very tilted, most of the pixels in the region of the hand are projection of points on the side of the hand. In this case, if the system can model the curvature of the hand region, it can effectively use all the pixels instead of considering most of them as outliers.

2. Adding extra sources of information to the model. These sources may include the silhouette of the hand, the edge map of the hand region and information from the fingers. For example, in some applications, it is important to detect if the region is from the back of the hand or the palm. The lines appearing in the palm region and the curvature of the hand can help in this decision. In addition, if it is known whether the person uses the left or right hand, the information from the fingers can also help to distinguish the palm from the back of the hand. Depending on the application, this process may be reversed and information from the fingers along with the side of the hand can be deployed to distinguish right hand from the left.

3. Creating new applications using the information provided by the hand tracking method. A useful application could be a virtual panel with a set of push buttons, turning knobs and sliding sliders being controlled by a hand. This virtual device can eliminate the need for carrying physical devices. Also, the type, position and purpose of the buttons and knobs can be modified instantly depending on the application or context.

4. Using the hand position and orientation to recognize the hand gesture more accurately. One of the common approaches in hand gesture recognition is extracting a set of features from the image and comparing them with the entries of a database to find the best matched gesture. However, due to the complexity of the hand shape and the degrees of freedom, this approach can be highly ambiguous. By including the information of the position and orientation of the hand, the range of the potential gestures is narrowed and the probability of finding the correct match could be improved.

Also, the work done in chapter 3 can be extended in the following directions:

1. Deploying cues other than skin color to segment the hand and head regions. This can be helpful in cases where a person wears a glove or the lighting condition causes the skin detection module to fail. These cues may include the shape of the hand and the position of the other body parts. Note that if the distance between the cameras and the subjects is large (as it was in our experiments), then shape analysis may not be a reliable source of information.

2. Adding information about the elbow and shoulder to the model. This can increase the accuracy of the tracking and can be helpful for cases where hands are not visible. One way to deploy this information is to augment the state vectors in particles to include them. Also, the dimensions of the limbs can be estimated from the height of a person. This information can be used to rule out some of the candidate positions and physically impossible body poses.

3. Including a broader range of activities to classify and increasing data sources to make their classification possible. Classifying the type of the object a person carries can be helpful in expanding the range of classified activities. For instance, in the desk example shown in chapter 3, recognizing the cup, paper and pan can be used to classify drinking and writing applications. Adding shape analysis to the presented color analysis can be useful in object recognition too.

4. Taking a hybrid approach for limb tracking. As explained in chapter 3, as the number of people in the scene increase, the probability of occlusion increases drastically. Therefore, tracking of the limbs, especially the hands, becomes very hard as their visibility diminishes. Using some parallel methods such as body modeling can help in increasing the level of accuracy. Also, some strategies should be developed to re-detect and label the limbs which lose their tracks. Proximity between the old and new positions as well as the similarity between appearances can be used as cues.

5. Complex activity classifications using the trajectories of the hands and heads tracked using our method. For instance, in a store scenario, the cashier or

the customer may be expected to perform some particular actions or interact with particular devices such as a credit card reader or a bag of goods. The information extracted from our system may be used for anomaly detection in such scenarios.

4.2 Final Word

Computer vision, along with other fields of technology, can provide easier and more convenient methods of using machines and devices thereby increasing the power and comfort of people. These technologies can also create safer and more secure environments to live. Any scientific discovery and innovation is a step in this direction.

Bibliography

[1] Kyungnam Kim and Larry S. Davis. Multi-camera tracking and segmentation of occluded people on ground plane using search-guided particle filtering. In *ECCV (3)*, pages 98–109, 2006.

[2] Aravind Sundaresan and Rama Chellappa. Multi-camera tracking of articulated human motion using motion and shape cues. In *ACCV (2)*, pages 131–140, 2006.

[3] Z. Yue and R. Chellappa. Pose-normalized view synthesis from silhouettes. *IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP)*, March 2005.

[4] Dimitrios Tsoumakos, Konstantinos Bitsakos, Yiannis Aloimonos, and Nick Roussopoulos. A framework for distributed human tracking. In *PDPTA*, pages 863–868, 2005.

[5] Jian Li, Shaohua Kevin Zhou, and Rama Chellappa. Appearance modeling under geometric context. *iccv*, 2:1252–1259, 2005.

[6] Vinay D. Shet, V. Shiv Naga Prasad, Ahmed M. Elgammal, Yaser Yacoob, and Larry S. Davis. Multi-cue exemplar-based nonparametric model for gesture recognition. In *ICVGIP*, pages 656–662, 2004.

[7] A Sundaresan, A RoyChowdhury, and R Chellappa. Multiple view tracking of human motion modelled by kinematic chains. *International Conference on Image Processing*, 2004.

[8] Larry S. Davis, Vasanth Philomin, and Ramani Duraiswami. Tracking humans from a moving platform. In *ICPR*, pages 4171–4178, 2000.

[9] Ismail Haritaoglu, David Harwood, and Larry S. Davis. An appearance-based body model for multiple people tracking. In *ICPR*, pages 4184–4187, 2000.

[10] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Comput. Surv.*, 35(4):399–458, 2003.

[11] Ying Wu and Thomas S. Huang. Vision-based gesture recognition: A review. *Lecture Notes in Computer Science*, 1739:103+, 1999.

[12] D. J. Sturman and D. Zeltzer. A survey of glove-based input. *IEEE Computer Graphics and Applciations*, 14:30–39, 1994.

[13] J. Lee and T.L. Kunii. Model-based analysis of hand posture. *IEEE Computer Graphics and Applications*, pages 77–86, September 1995.

[14] V. Pavlovic, R. Sharma, and T. Huang. Gestural interface to a visual computing environment for molecular biologists, 1996.

[15] J. Segen. Controlling computers with gloveless gestures. *Proc.of Virtual Reality Systems*, 1993.

[16] J. J. Kuch and T. S. Huang. Vision based hand modeling and tracking for virtual teleconferencing and telecollaboration. *International Conference on Computer Vision and Pattern Recognition*, pages 666–671, 1995.

[17] Vladimir Pavlovic, Rajeev Sharma, and Thomas S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):677–695, 1997.

[18] J. M. Rehg and T. Kanade. Digiteyes: Vision-based hand tracking for human-computer interaction. *IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, 1994.

[19] J. M. Rehg and T. Kanade. Visual tracking of high dof articulated structures: an application to human hand tracking. *3rd ECCV*, volume II, May 1994.

[20] T. Heap and D. Hogg. Towards 3d hand tracking using a deformable model. *International Conference on Automatic Face and Gesture Recognition*, 1996.

[21] Q. Delamarre and O. Faugeras. Finding pose of hand in video images: a stereo-based approach. *Proceedings of FG'98*, April 1998.

[22] R. Rosales, V. Athitsos, L. Sigal, and S. Sclaroff. 3d hand pose estimation using specialized mappings. Technical report, Los Alamitos, CA, July 9–12 2001.

[23] Vassilis Athitsos and Stan Sclaroff. 3D Hand Pose Estimation by Finding Appearance-Based Matches in a Large Database of Training Views. Technical Report BUCS-TR-2001-021, CS Department, Boston University, October 22 2001.

[24] V. Athitsos and S. Sclaroff. Estimating 3d hand pose from a cluttered image. *IEEE Conference on Computer Vision and Pattern Recognition*, 2(18-20):II–432–9, June 2003.

[25] B.P.L. Lo and S.A. Velastin. Automatic congestion detection system for underground platforms. *Proc. of 2001 Int. Symp. on Intell. Multimedia, Video and Speech Processing*, pages 158–161, 2000.

[26] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting moving objects, ghosts and shadows in video streams. *IEEE Trans. on Patt. Anal. and Machine Intell.*, 25(10):1337–1342, October 2003.

[27] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder:real-time tracking of the human body. *IEEE Trans. on Patt. Anal. and Machine Intell.*, 19(7):780–785, 1997.

[28] C. Stauffer and W.E.L. Grimson. Learning patterns of activity using real-ime tracking. *IEEE Trans. on Patt. Anal. and Machine Intell.*, 22(8):747–757, 2000.

[29] S. Zhou J. Shao and R. Chellappa. Simultaneous background and foreground modeling for tracking in surveillance video. *Proc. Intl. Conf. on Image Processing*, October 2004.

[30] A. Elgammal, D. Harwood, and L.S. Davis. Non-parametric model for background subtraction. *Proc. of ICCV '99 FRAME-RATE Workshop*, 1999.

[31] B. Han, D. Comaniciu, and L. Davis. Sequential kernel density approximation through mode propagation: applications to background modeling. *Proc. ACCV -Asian Conf. on Computer Vision*, 2004.

[32] N. M. Oliver, B. Rosario, and A. P. Pentland. A bayesian computer vision system for modeling human interactions. *IEEE Trans. on Patt. Anal. and Machine Intell.*, 22(8):831–843, 2000.

[33] X. Yin, D. Guo, and M. Xie. Hand image segmentation using color and rce neural network. *IJRAS*, 34:235–250, March 2001.

[34] V. Vezhnevets, V. Sazonov, and A. Andreeva. A survey on pixel-based skin color detection techniques. *Proc. Graphicon-2003*, pages 85–92, September 2003.

[35] K. Abe, H. Saito, and S. Ozawa. 3d drawing system via hand motion recognition from two cameras. *Proceeding of the 6th Korea-Japan Joint Workshop on Computer Vision*, pages 138–143, January 2000.

[36] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, 2001.

[37] Peter J. Huber. *Robust statistics*. John Wiley and Sons, 1981.

[38] John Fox. *Robust Regression: Appendix to An R and S-PLUS Companion to Applied Regression*. SAGE Publications, 2002.

[39] S. Geman and D. E. McClure. Statistical methods for tomographic image reconstruction. *Proc. of the 46-th Session of the ISI, Bulletin of the ISI*, 52:5–21, 1987.

[40] Michael J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *CVIU*, 63(1):75–104, Jan 1996.

[41] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[42] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *Proc. of 7th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 674–679, 1981.

[43] A. M. Takalp. *Digital Video Processing*. Prentice Hall, 1995.

[44] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.

[45] Anil K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, 1989.

[46] L. Bretzner and T. Lindeberg. Use your hand as a 3-d mouse ... *European Conference on Computer Vision*, 1998.

[47] N. Jojic, B. Brumitt, B. Meyers, S. Harris, and T. Huang. Detection and estimation of pointing gestures in dense disparity maps. *International Conference on Automatic Face and Gesture Recognition*, 2000.

[48] Y. Cui and J. Weng. A learning-based prediction-and-verification segmentation scheme for hand sign image sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1999.

[49] Z. Zhang, Y. Wu, Y. Shan, and S. Shafer. Visual panel: Virtual mouse, keyboard and 3d controller with an ordinary piece of paper. *Workshop on Perceptive User Interfaces*, 2001.

[50] Y. Nam and K. Wohn. Recognition of space-time hand-gestures using hidden markov model. *ACM Symposium on Virtual Reality Software and Technology*, 1996.

[51] D. Huttenlocher, D. Klanderman, and A. Rucklige. Comparing images using the Hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, September 1993.

[52] Shuo Wang, Xiaocao Xiong, Yan Xu, Chao Wang, Weiwei Zhang, Xiaofeng Dai, and Dongmei Zhang. Face-tracking as an augmented input in video games: enhancing presence, role-playing and control. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, New York, NY, USA, 2006. ACM Press.

[53] Tollmar Konrad, David Demirdjian, and Trevor Darrell. Gesture + play: full-body interaction for virtual environments. In *CHI '03: Proceedings of the SIGCHI conference on Human Factors in computing systems*, New York, NY, USA, 2003. ACM Press.

[54] R. Karlsson and F. Gustafsson. Monte carlo data association for multiple target tracking, 2001.

[55] D. Schulz, W. Burgard, D. Fox, and A. Cremers. Tracking multiple moving targets with a mobile robot using particle filters and statistical data association, 2001.

[56] C. Hue, J. Le Cadre, and P. Perez. Tracking multiple objects with particle filtering, 2000.

[57] S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House Radar Library, 1999.

[58] R. L. Streit and T. E. Luginbuhl. Maximum likelihood method for probabilistic multi-hypothesis tracking. *Proceedings of Signal and Data Processing of Small targets*, 2235:394–405, 1994.

[59] Yaakov Bar-Shalom and William Dale Blair. *Multitarget-Multisensor Tracking Applications and Advances - Volume III*. Artech House, 2000.

[60] Zia Khan, Tucker R. Balch, and Frank Dellaert. An mcmc-based particle filter for tracking multiple interacting targets. In *ECCV (4)*, pages 279–290, 2004.

[61] Simo Särkkä, Aki Vehtari, and Jouko Lampinen. Rao-blackwellized monte carlo data association for multiple target tracking, 2004.

[62] Simo Särkkä, Aki Vehtari, and Jouko Lampinen. Rao-blackwellized particle filter for multiple target tracking. *Preprint submitted to Elsevier Science*, September "2005".

[63] Avitzour D. Stochastic simulation bayesian approach to multitarget tracking. *IEE Proceedings - Radar, Sonar and Navigation*, 142.

[64] N. Gordon. A hybrid bootstrap filter for target tracking in clutter, 1997.

[65] Mark Morelande and Darko Musicki. Fast multiple target tracking using particle filters, December 2005.

[66] Changjiang Yang, Ramani Duraiswami, and Larry Davis. Fast multiple object tracking via a hierarchical particle filter. *iccv*, 1:212–219, 2005.

[67] Michael Isard and John MacCormick. BraMBLe: A bayesian Multiple-Blob tracker. pages 34–41.

[68] G Qian, R Chellappa, and Q Zheng. Bayesian algorithms for simultaneous structure from motion estimation of multiple independently moving objects, 2005.

[69] Caifeng Shan, Yucheng Wei, Tieniu Tan, and Ojardias Ojardias. Real time hand tracking by combining particle filtering and mean shift. *Face and Gesture Recognition*, 00:669, 2004.

[70] Carine Hue, Jean-Pierre Le Cadre, and Patrick Perez. A particle filter to track multiple objects. *womot*, 00:0061, 2001.

[71] Arnaud Doucet, Nando De Freitas, and Neil Gordon. *Sequential Monte Carlo Methods in Practice (Statistics for Engineering and Information Science.* Springer, 2005.

[72] A. Doucet, S.J. Godsill, and C. Andrieu. On sequential simulation-based methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[73] A. Papoulis. *Probability, Random Variables, and Stochastic Processes, 2nd Edition.* McGraw-Hill, 1984.

[74] A. T. Bharucha-Reid. *Elements of the Theory of Markov Processes and Their Applications.* McGraw-Hill, 1960.

[75] Peter S. Maybeck. *Stochastic models, estimation, and control. Volume 1.* Academic Press, 1979.

[76] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking, 2002.

[77] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[78] John MacCormick and Andrew Blake. A probabilistic exclusion principle for tracking multiple objects. *International Journal of Computer Vision*, 39(1):57–71, 2000.

[79] Michael Isard and Andrew Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[80] S. Zhou, R. Chellappa, and B. Moghaddam. Visual tracking and recognition using appearance-based modeling in particle filters, 2003.

[81] Niclas Bergman. *Recursive Bayesian estimation : navigation and tracking applications.* Link¨ping, 1999.

[82] A. Doucet. On sequential monte carlo sampling methods for bayesian filtering, 1998.

[83] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[84] Antonio Criminisi, Ian D. Reid, and Andrew Zisserman. Single view metrology. *International Journal of Computer Vision*, 40(2):123–148, 2000.

[85] Kyungnam Kim, Thanarat H. Chalidabhongse, David Harwood, and Larry S. Davis. Real-time foreground-background segmentation using codebook model. *Real-Time Imaging*, 11(3):172–185, 2005.

[86] V. S. N. Prasad, V. Kellokumpu, and L. S. Davis. Ballistic hand movements. *Proceeding of Conference of Articulated Motion and Deformable Objects (AMDO)*, July 2006.

[87] Andrea Goldsmith. *Wireless Communications*. Cambridge, 2005.