2005

Towards Full-Body Gesture Analysis and Recognition

Muthukumar B. Puranam
University of Kentucky

ABSTRACT OF THESIS

Muthukumar B Puranam

The Graduate School
University of Kentucky
2005

Towards Full-Body Gesture Analysis and Recognition

ABSTRACT OF THESIS

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
at the University of Kentucky

By

Muthukumar B Puranam

Lexington, Kentucky

Director: Dr. Christopher Jaynes, Associate Professor of Computer Science

Lexington, Kentucky

2005

ABSTRACT OF THESIS

Towards Full-Body Gesture Analysis and Recognition

With computers being embedded in every walk of our life, there is an increasing demand for intuitive devices for human-computer interaction. As human beings use gestures as important means of communication, devices based on gesture recognition systems will be effective for human interaction with computers. However, it is very important to keep such a system as non-intrusive as possible, to reduce the limitations of interactions. Designing such non-intrusive, intuitive, camera-based real-time gesture recognition system has been an active area of research research in the field of computer vision.

Gesture recognition invariably involves tracking body parts. We find many research works in tracking body parts like eyes, lips, face etc. However, there is relatively little work being done on full body tracking. Full-body tracking is difficult because it is expensive to model the full-body as either 2D or 3D model and to track its movements.

In this work, we propose a monocular gesture recognition system that focuses on recognizing a set of arm movements commonly used to direct traffic, guiding aircraft landing and for communication over long distances. This is an attempt towards implementing gesture recognition systems that require full body tracking, for e.g. an automated recognition semaphore flag signaling system.

We have implemented a robust full-body tracking system, which forms the backbone of our gesture analyzer. The tracker makes use of two dimensional link-joint (LJ) model, which represents the human body, for tracking. Currently, we track the movements of the arms in a video sequence, however we have future plans to make the system real-time. We use distance transform techniques to track the movements by fitting the parameters of LJ model in every frames of the video captured. The tracker's output is fed a to state-machine which identifies the gestures made. We have implemented this system using four sub-systems. Namely

1. Background subtraction sub-system, using Gaussian models and median filters.
2. Full-body Tracker, using L-J Model APIs
3. Quantizer, that converts tracker's output into defined alphabets
4. Gesture analyzer, that reads the alphabets into action performed.

Currently, our gesture vocabulary contains gestures involving arms moving up and down which can be used for detecting semaphore, flag signaling system. Also we can detect gestures like clapping and waving of arms.

Key words: Distance Transform, Background Modelling, Link-Joint Model, Golden Section Search, Gesture Vocabulary

(Muthukumar B Puranam)

(Date)

Towards Full-Body Gesture Analysis and Recognition

By

Muthukumar B Puranam

_____
(Director of Thesis)

_____
(Director of Graduate Studies)

_____
(Date)

RULES FOR THE USE OF THESIS

Unpublished thesis submitted for the Master's degree and deposited in the University of Kentucky Library are as a rule open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted, but quotations or summaries of parts may be published only with the permission of the author, and with the usual scholarly acknowledgments.

Extensive copying or publication of the thesis in whole or in part requires also the consent of the Dean of The Graduate School of the University of Kentucky.

A library which borrows this thesis for use by its patrons is expected to secure the signature of each user.

Name	Date

THESIS

Muthukumar B Puranam

The Graduate School
University of Kentucky
2005

Towards Full-Body Gesture Analysis and Recognition

THESIS

A thesis submitted in partial fulfillment of the
requirements for the degree of Master of Science
at the University of Kentucky

By

Muthukumar B Puranam

Lexington, Kentucky

Director: Dr. Christopher Jaynes, Associate Professor of Computer Science

Lexington, Kentucky

2005

Acknowledgments

I want to begin by thanking my thesis chair Dr.christopher Jaynes, for the invaluable guidence and support he provided me during my Master's studies at University of Kentucky. Then I would like to thank Dr.Jane Hayes and Dr.D.Manivannan, members of my thesis Committee for their insightful suggestions and guidence. I thank our team members of Metaverse lab for their support and understanding througout my thesis work.

I would like to thank my friends, Praveen Devabhaktuni, Ganapathy Chidambaram and Inies Raphael who posed in video sequences that were used as sample data. A special thanks to my mother and other family members for their encouragement and support during my studies at University of Kentucky. A very special thanks to my elder brother, Gopalakrishnan Puranam, who supported me financially and morally. I would like to thank my friend vijay venkatesh in helping me editng this report.

I would like to thank all my friends and classmates who made my stay at university a memorable one. I would like to thank everyone at Department of Computer Science with an emphasis to Dr.Grzegorz W. Wasilkowski for his continuous help and support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

We normally interact with computers using devices like a mouse, keyboard, wireless devices, etc. However, present Human Computer Interaction (HCI) devices are not intuitive for common man and it needs some training before using them and also they limit application of computers in daily life as they need the presence of human beings in the vicinity of computers. With computers being embedded almost in every walk of our life, we need more intuitive means of interacting with computers. Since gestures form very important part of our communication, designing HCI devices that can recognize human gestures can be more effective in taking the computer technology closer to man. Hence there is an increased attention in the field of gesture analyzers, wherein computer can "See" the environment and "respond" to the gestures made.

Consider for example a smart living room, where without moving from chair or sofa and without having to carry any kind of remote control devices, one can control electrical appliances like TV, Fan, Radio, etc, using gestures. This shows us that Human Computer Interaction (HCI) devices based on visual inputs is always advantageous over other interfacing devices, as it eliminates the need for the presence of human beings in the vicinity of the device.

1.1 Full-Body Gesture Analysis

Human beings make many subtle gestures using different body parts to communicate. An effective gesture recognition system should track the movements of these body parts. However, tracking every part of the body is nearly impossible because of time and space constraints, self-occlusion and other confounding factors. The current gesture analysis systems typically try to classify a set of gestures made using specific body parts and try to recognize them. There has been significant research dedicated to head, eye, lips, facial expressions, but relatively little in recognizing

large pose. Full-Body gesture analysis is a difficult problem because modeling a full-body segment in 3D or 2D for tracking is very complex and meeting the constraints of real time implementation is very challenging. In this work, we implement a full-body tracking system, which is used for analyzing gestures made at a distance from the computer.

1.2 Gesture Analysis System Design

The design of gesture analyzer and thereby a HCI device based on visual inputs typically consists of the following challenging steps:

* Identifying Human beings in the environment

  To identify human beings in an environment, the system must understand the environment, which requires a robust modeling of the background. Also we require techniques to segment human beings from the background. A robust tracking system requires an adaptive background model that can handle variations in lightnings, scene clutters, and other arbitrary changes to the scene being observed. We can find some of the methods being discussed in [1, 2, 3], etc.

* Tracking relevant body parts

  Tracking involves defining and segmenting the relevant moving object or parts of the moving object in an environment, which is a difficult problem. Some works make use of external markings, feature based tracking or training methods. We, for example, take the location of relevant parts in the first frame as user input and then track limbs automatically as described in Chapter 4.

  The tracking problem is solved by modeling the object being tracked as either 2D or 3D model. We have designed a 2D model called link-joint model which is described in Chapter 3. We will discuss in detail regarding other model based approaches being used to solve the problem of tracking in Chapter 2.

* Analyzing pose-gesture

  Gesturing has been a non-verbal communication mode from the day of evolution of human beings . It is a challenging task to make computers "understand" the gestures being made by human beings. Gesture recognition techniques are mostly Model Based or View based. Gestures are normally classified using rules or by training the system. Gesture analyzers make use of the trackers to identify them.We try to quantize the outputs of the tracker into alphabets and the gestures are read as words by the analyzer.

* Applications

  Designing appropriate applications that can make use of the gestures recognized has always been an interesting problem. In Chapter 2, we discuss some of these applications.

Despite of above mentioned challenges, HCI-enabled applications are making their impact in the market in the following areas.

. Control of consumer electronics

. Interaction with visualization system

. Control of mechanical systems

. Computer games

. Video surveillance

There are HCI applications being built based upon lip movements detection, facial expressions detection, etc. Automatic eye movement detection is used to develop driver safety systems, which raises alarm when the driver falls asleep.

Full-body gesture recognition system is an emerging field. A full-body gesture analysis system can be used in several areas including

. Semaphore sign language detection

. Automatic manned traffic signals detection

. Detection of signals by flagman in airplane runways

. Modeling mannerisms for automatic recognition in security systems.

Hence camera-based gesture recognizing systems play very important role in developing novel HCI devices. However designing such real-time systems is a real challenging task as the output of such systems depend on various sub-systems with which it is made off.

1.3 System Overview

A typical video-based gesture analysis system involves the following challenges.

1. Background Subtraction

2. Modeling the object being tracked

3. Tracking

4. Gesture Detection and Analysis

Each of these challenges represent an active research area in itself. In this work, we have designed and implemented a gesture analyzing HCI system based on distance transform techniques. We have implemented the system using several sub-systems to perform tasks specified in each stage as shown in the Figure 1.1.

1.3.1 Background Modeling

Background subtraction is a complex problem because of factors like

- Varying lighting in different environments.
- Varying Light conditions in same environment over a period of time.
- Presence of specular surfaces in the background.

Hence it necessitates us to impose certain constraints on the background to model it. Our background subtraction model is also designed with certain assumptions about the background. We have discussed in detail about our background model in Chapter[3].

1.3.2 Distance Transform

Distance transform is an operator that is applied on binary images to transform an Image in such a way that the Grey level intensity of points inside foreground regions are transformed to show the distance to the closest boundary from each point. In the figure 1.2, we apply distance transform on a rectangular matrix which represents a binary Image, we can see the output matrix's elements represent the distance from the closest boundary. This operator when applied to an image gives us the skeleton of the Image. Skeleton is defined as the set of points whose distance from the nearest boundary is locally maximum. Figure 1.4 shows how distance transform operator obtains the skeleton of the foreground. Distance transform forms an integral part of our tracking system as explained in Chapter 4. In the system block diagram, shown in Figure 1.1, the block labeled 'A' accomplishes foreground extraction and distance transforms the foreground object.

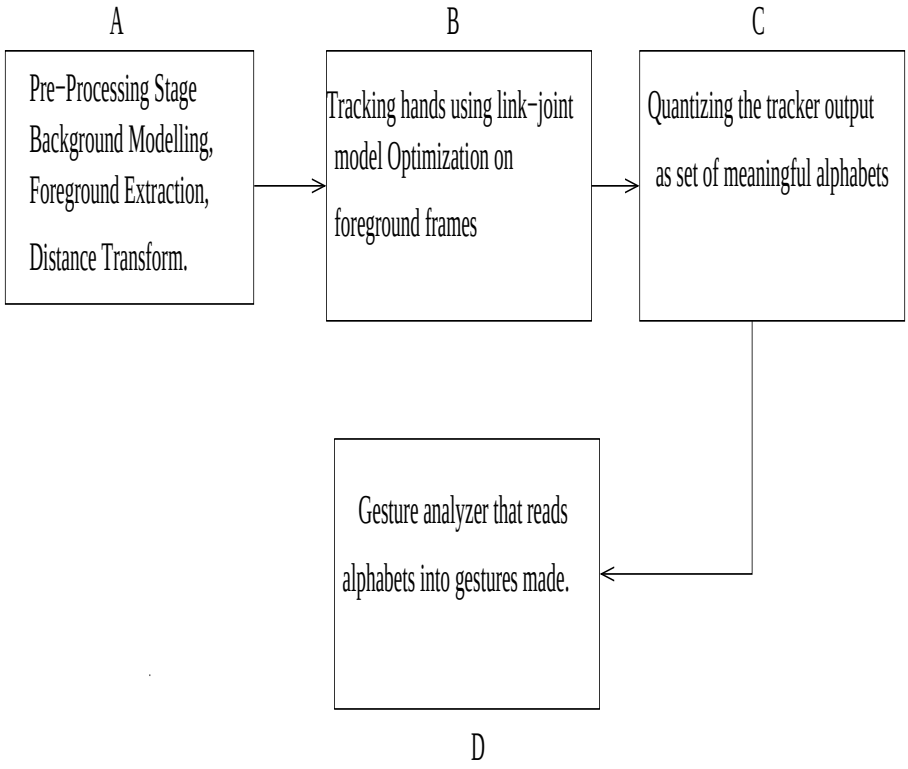

Figure 1.1: System Block Diagram

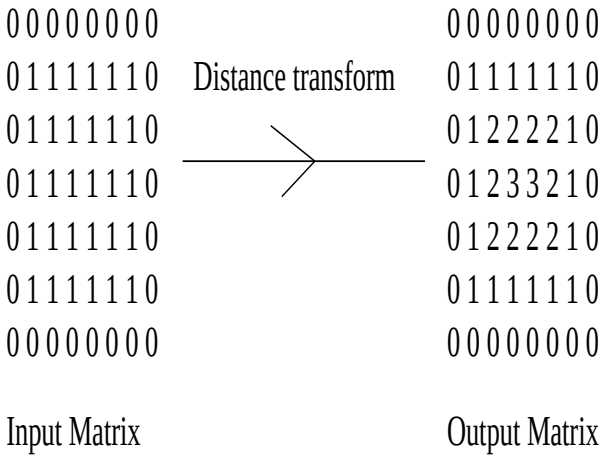

Figure 1.2: Distance transform of simple rectangular matrix

Figure 1.3: Image before transform

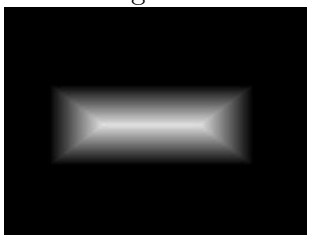

Figure 1.4: Image after transform

1.3.3 Link-Joint Model

As we know, tracking of an object in a video sequence involves modeling the object either as 2D or 3D. In this work we are implementing a full-body tracking system to accomplish gesture recognition. Modeling full-body can be a highly complicated 3D model that represents every joints with every degree of freedom or can be very simple 2D model with very fewer degrees of freedom for each joint. In general, complex model requires more reconstruction time hence forms a bottleneck in real time implementation. However very simple models cannot be effective in replicating the movements of full-body. A typical gesture analysis system design makes some assumptions about degrees of freedom required for the joints based upon actions being recognized to reduce the processing speed. Based on this idea, we are proposing a 2D Link-Joint (LJ) model that represents full body movements. The implementation details of LJ model is discussed in Chapter[3].

1.3.4 Tracking Module

Robust tracking system forms the backbone of a gesture analyzer. Tracking can be very effective if we use more than one camera, however it may not be practical in real-time situations. Hence tracking using a single camera is an important research area in the field of computer vision. We are proposing a monocular non-intrusive tracking system to track full body movements using

the LJ model. The tracking system is discussed in detail in Chapter 4. In the system block diagram shown in Figure 1.1, the block labeled 'B' accomplishes the full-body tracking using the LJ Model.

1.3.5 Gesture Recognition Module

As we know, human gestures are innumerable and they make use of each and every part of the body. Hence it is nearly impossible to make a complete set of gestures and there is no technical knowhow to design a system that recognizes them in real time. Hence any gesture detection is done only upon defined set of gestures. There have been lot of work on detecting facial expressions by tracking movements of eyes, lips, head, etc. However, there are relatively less non-intrusive methods available to track full-body.
We have designed a full-body tracker with the help of distance tracking techniques. We have implemented a gesture analyzer which detects the following gestures

1. Both hands moving up
2. Both hands moving down
3. Right hand moving up
4. Right hand moving down
5. Left hand moving up
6. Left hand moving down
7. Clapping
8. Waving both hands
9. Waving Right hand
10. Waving Left hand

In the system block diagram Figure 1.1, the block labeled 'C' converts the tracker's output into a form that could be read by gesture analyzer and the block labeled 'D' is responsible for recognizing the gestures.
Please refer to Chapter 5 for a detailed analysis of our gesture recognition system. In a nutshell, we have designed and implemented a non-intrusive gesture analysis system that detects gestures made using full-body movements. We have also proposed methods to make our system work in real time in future.

1.4 Organization of the thesis

We have implemented this gesture analysis system using sub-systems as shown in the figure 1.1. We have devoted chapters that discuss implementation details of each sub-system. Chapter 3 deals with pre-processing stages where we discuss our background modeling and Link-Joint model. In Chapter 4 we discuss our tracking technique and its implementation. In Chapter 5 we discuss our gesture vocabulary and state diagram, used to identify gestures from the output of tracker. We discuss our results in Chapter 6 and we provide a road map for real-time implementation in Chapter 7. In Chapter 2, we make a brief review of existing gesture analyzing systems by comparing and contrasting with implementation of sub-systems shown in Figure 1.1.

Chapter 2

Related Work

Human Computer Interaction (HCI) using gesture analysis is a complex procedure, which involves integration of several sequential processing stages. Typically a gesture based HCI will have the following four different stages.

. Background subtraction

. Tracking

. Gesture Analysis

. HCI applications

Please refer to the block diagram shown in Figure 1.1, which explains our gesture analyzer. Each of these stages represents a highly active research area in the field of computer vision. In this Chapter, we discuss existing methods of implementation of each of these sub-systems and compare and contrast with our methods.

2.1 Background Subtraction

Accurate background modeling is the basic requirement of object-tracking and segmentation problems. There have been very robust techniques proposed for both indoor and outdoor backgrounds. Xiong et al. [1] propose a multi-resolution modeling of dynamic scenes using weighted match filters. They have very promising results under several indoor and out door conditions (8%-12% false positive rates). But we don't have multimodal background, instead we have unimodal background with slightly changing foreground. So we propose a pixel based Gaussian Distribution Model for modeling the background. For future works we can extend our background to multi

modal GMM as in [3]. Pfinder system [7], uses similar background modeling approach in YUV space, while we use RGB color space.
Our algorithm works in two levels

. Pixel Level

. Region Level

We find similar approach by Omar Javed et al. [2], where they process in three levels. They process in an additional level called as "Frame Level" processing. Since we are not processing for outdoor environments we have not considered adding additional layer of processing. However, we use median filters for region level processing, where [2] uses a gradient -based approach. We find our approach highly successful for the constraints we have for our background. We have discussed the modifications being considered for lesser constrained environments in Chapter 7

2.2 Tracking

Tracking human beings can be broadly defined as the ability of a system to locate body parts and follow them. This problem is solved using

. Monocular approach or

. Multi view approach.

The monocular approach is challenging because we cannot obtain depth information from the images and it makes the problem ill-conceived. In multiple cameras, we can effectively reconstruct the object being tracked in 3D by establishing correspondence between images from different cameras and we can address problems created by occlusions, depth ambiguities, etc. n real-time situations multi-view approach is not practical because of time and space constraints and monocular approach is preferred even though it involves applying certain constraints over the background model and motion patterns being tracked. Monocular tracking is very active research area in computer vision. There have been many deterministic continuous, discrete and stochastic approaches based on particles proposed. Please refer to [8] for a detailed survey on tracking techniques. ormal tracking techniques try to reconstruct 2D or 3D models of either whole human body or body segments. We can see 2D reconstruction in [7] or 3D human reconstruction [9]. The models proposed can have constraints regarding movements that could be tracked. Based on these constraints, the complexity of the model differs among proposed models. However, we are proposing a tracking approach where

we are not reconstructing any 2D or 3D models, instead we try to fit our proposed link-Joint model over the distance transformed images.
As we know, tracking is typically done in the following two steps.

1. Initialization

2. Optimization and Estimation

We will compare and contrast our approach in each of these steps with existing trackers.

2.3 Initialization

In any tracking application, getting the initial positions of moving segments is a very difficult problem. The pfinder [7] uses a 2-D blob model based on color distribution of the pixels, to represent various parts of body like head, arms, torso, etc. Some trackers use external markers as in [4] to segment the corresponding body parts to be tracked. The above mentioned techniques do not require any initial user inputs in locating the interested body parts, but they have their own constraints like subjects carrying markers to segment relevant parts, Skin region exposed so that the texture can be used for tracking, etc. Instead, we require the initial arms position as user input and construct the LJ model.
The proposed Link-joint (LJ) approach models the arms and shoulder part as link objects joined by spring objects. We discuss the design and implementation of LJ model in Chapter 3. Initially the user is asked to draw the model over the first frame and LJ model is built is constructed based on the inputs. We track the arms by fitting the model over the distance transformed images of corresponding sequences.Here we don't use any external markers as in [4] and we did not try to build a 3D or 2D human models as in [7] or [9].

2.4 Estimation and Optimization

Estimating the position of segments being tracked is a difficult problem. The approaches for solving this problem are broadly classified as

. Deterministic and

. Stochastic

Deterministic approaches can be discrete or continuous. Continuous approaches typically use locally linearized model approximations and unimodal Gaussian error distributions. The distributions are often propagated in time using Extended Kalman Filter [4]. Stochastic approaches are based on Particle Filter as in [6]. Also stochastic approaches need training data for complex motion models.

We follow a 2D approach similar to Pfinder [7]. But Pfinder uses a state machine model to estimate the new position of the segments being tracked. Also Pfinder needs a learning phase to know its background. Our model assumes only the movement of arms and the foreground is parallel to the image plane.We capture a video sequence where the subject is making gestures and we analyse each frames to identify the gestures made. In this process, we obtain the parameters of Link-Joint model for initial frame as user input and we track the changes in the parameters subsequent frames. We have designed a step function to estimate the new positions of the arms in each frame, please refer to Chapter 4. We define the movement of each link using a single variable non-linear function, hence we go for numerical optimization methods like Golden Section Search method [15]. We use Distance Transform (DT) to fit links of LJ model.

2.5 Degree of Freedom (DOF)

Complex modeling of human beings are required because arms and legs are very kinematic. According to [12] a typical human 3D model has around 30 kinematic Degrees of Freedom(DOF). But Based on the application we try to track less number of DOF. For example Pfinder [7] tracks three DOF. Since we follow a Link-Joint model, we have each link having their own DOF. The table 3.1 gives the DOF allowed for different links.

2.6 Gesture Recognition and HCI

In this information age we have bigger role for computers in every walk of life. People of all age groups and with varying physical abilities or disabilities have to interact more with computers. This makes the traditional way of interactions like mouse, keyboard, etc. are very limiting as they require the presence of user in the vicinity of interacting device. Vision based interaction techniques are more attractive as they remove such bottlenecks. There have been a lot of research in the area of gesture analysis in recent times. We find a survey of vision-based Hand gestures in [13] and according Derpanis [13], gesture recognition techniques are broadly classified as

1. Rule-based methods, where the gestures are manually encoded.

2. Learning approaches that use machines to infer models of gestures from a set of exemplars.

Our approach belongs to first class, where we try to quantize tracker results into alphabets and we later try to read them as gestures in a hierarchical way. We can find a similar approach in [14]. We can find that the gesture analyzers are used for controlling real-time applications. MIT Media Lab has implemented applications like Smart Room, Smart Desk, etc. A real time implementation application discussed in [6] demonstrates the application of vision based HCI devices. These results promise vision-based devices playing a major role in altering the way humans interact with computers in near future.
We wanted to demonstrate the utility of distance tracking in tracking. Nevertheless, we find distance transform is computationally expensive process. However, there are research works like [16], that deal with inventing simpler algorithms to calculate distance transform. In future we hope the cost DT will be very less and can be used as a tool for real-time tracking.

Chapter 3

Background Subtraction and Link-Joint Model

Tracking an object in a video sequence, typically involves modeling the object either in 2D or 3D. We have designed a system to track full-body movements using a 2D model called Link-Joint model. However, to make use of the model, we need to segment the foreground from the background. Hence We need to model the background to meet the robust demands of background subtraction. We have designed a Gaussian model that represents our background.
We extract the foreground by comparing the current image with the Gaussian background model. The extracted foreground is distance transformed and we try to fit the Link-Joint model to the new foreground. In this chapter we discuss implementation details of our Background model and Link-Joint model. We discuss the experimental results of our model along with advantages and disadvantages of the model.

3.1 Background Modeling

Background subtraction is a key problem in tracking. It needs to address various issues like changing light conditions, effects of moving elements in the background, objects that enter and leave the scene, etc. Hence background modeling has been very active research area and there are many adaptive techniques developed to address these issues [1, 2, 3]. The complexity of the problem grows with relaxing of the constraints imposed on the background. We consider an indoor background where we don't have moving elements other than the foreground human object. The variation in the background is caused by the electrical lights used in the room. To offset the noise introduced by the lighting we model each background pixel with a Gaussian distribution for every

R, G, B channel.

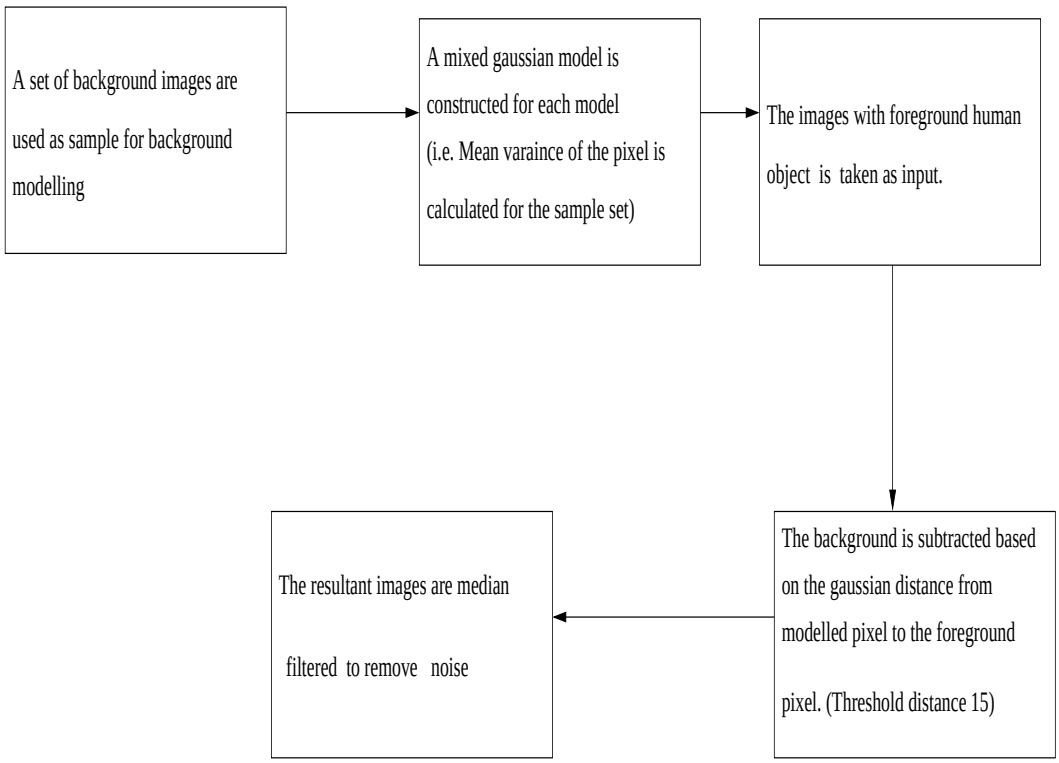

Figure 3.1: Flowchart for Background Subtraction

We collect the background image samples over a period of time (Typically for 10 secs at 30 fps). In the RGB space, the vector (r,g,b) represents both color and luminance. Luminance can be removed by normalizing the component values as

$$\mathrm{r} = \frac{r}{r+g+b}$$
$$\mathrm{g} = \frac{g}{r+g+b}$$
$$\mathrm{b} = \frac{b}{r+g+b}$$

we calculate the mean μ and covariance Cov for each background pixel from the sample images, where

$$\text{Mean: } \mu = \mathrm{E}\ \{\ \mathrm{X}\ \} \text{ where X=(r g b) } ^{T}$$

$$\text{Covariance: } Cov = \mathrm{E}\ \{\ (\mathrm{x} - \mu)\ (\mathrm{x}\text{-}\mu)\ ^{T}\}$$

$$\mathrm{p}(x) = \frac{1}{(2\pi)^{\frac{1}{2}}(|Cov|)^{\frac{1}{2}}} exp\{-\frac{1}{2}(X-\mu)^{T}Cov^{-1}(X-\mu)\}$$

where p(x) is likelihood of pixel belonging to background. We use Gaussian distance, i.e. the distance of the pixel from the mean value, as a criterion to segment the background pixels. Typically the pixels that are away from mean value of background pixel by five times the standard deviation is considered as foreground pixel. The resultant image is subjected to two levels of median filtering to remove noise introduced in the previous stages. The various steps followed in background subtraction are shown in Figure 3.1. The sequence of images in Figure 3.2 shows the results obtained in each step of our algorithm. Since our background is assumed as mostly stationary with no moving objects, the proposed model performs reasonably well. The drawback of this model is with specular surfaces, which might be falsely interpreted. So we assume there are no specular surfaces in the background. As future work, we want to implement more dynamic background subtraction algorithms discussed in [1]. We can use mahalanobis distance criterion as an alternative. But it is computationally complex and for our background constraints we feel simple Gaussian distance is more suitable. Figure 3.2 represents the set of input images to the background subtraction model. The output is a binary image with pixels identified as foreground are white, rest are black, are shown in the Figure 3.3.

3.1.1 Robustness

The background model designed is to help us solve the problem of tracking human arms. We need little more than just subtracting the foreground object from the background. The model we propose must be able to track the arms when it is moving over trunk region. This is done by using a reference image as mentioned in the previous sections, to segment out the arms when moving over the trunk. This aspect of our model is dealt in Chapter 4.

3.1.2 Limitations of Background Model

Even though we try to have a background model that has zero constraints over the background, because of highly dynamic nature of backgrounds like change in light conditions, specular surfaces, etc., such a model is not yet a reality. Hence it is imperative to model our background based on certain assumptions about it. In this section, we discuss about the assumptions and limitations of our model. ur model charecterizes each background pixel as 3D Gaussian distribution in RGB space as discussed in section 3.1. While performing background subtraction, we apply a threshold of five times the standard deviation on either side of the mean to decide on the foreground objects. In case of specular surfaces in the background, this method might introduce noise in the detected foreground which can be reduced by median filtering the subtracted image. As mentioned previously every

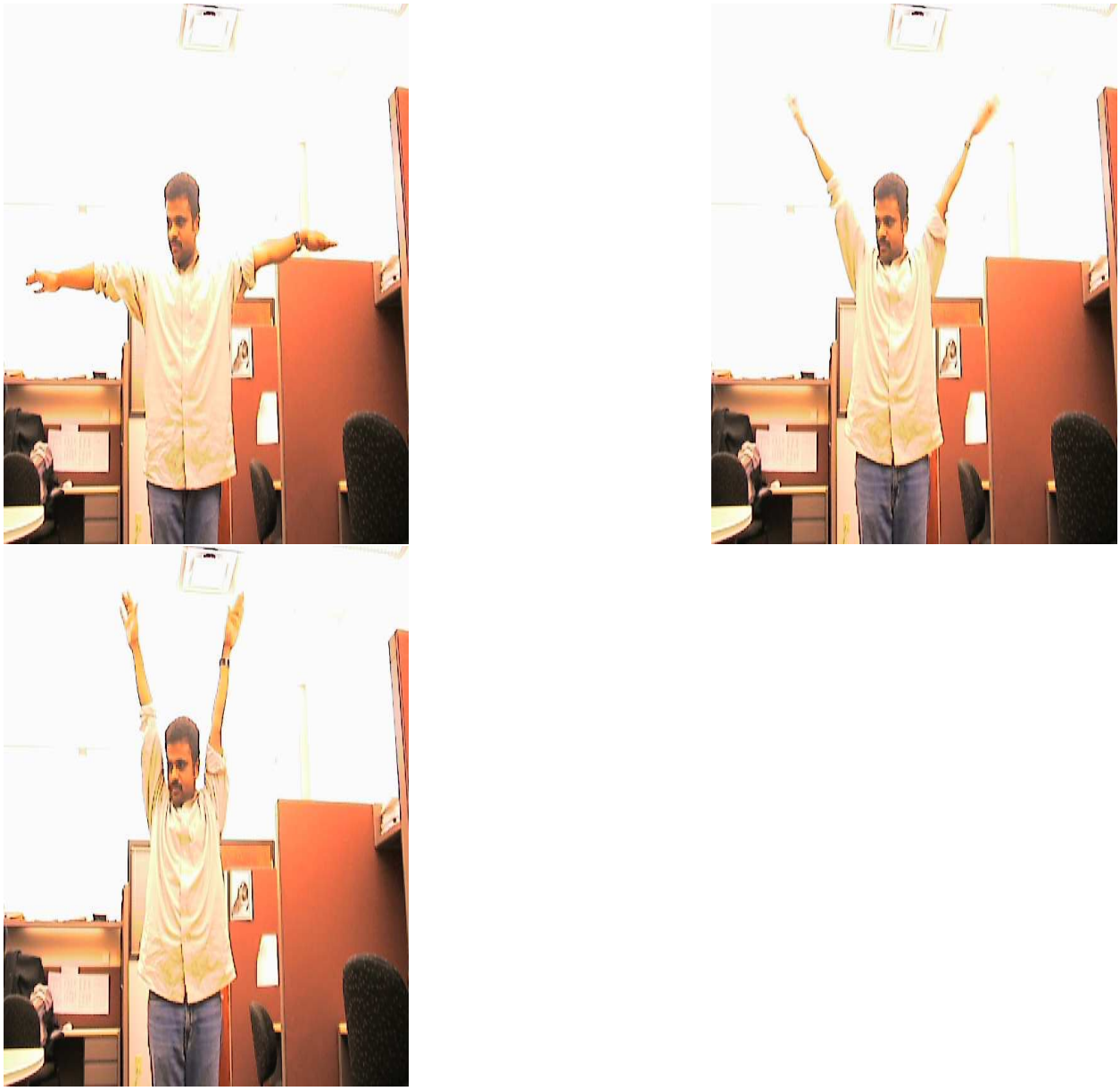

Figure 3.2: Input Images for Foreground extraction

Figure 3.3: Output binary Images where foreground pixels are white

background model assumes certain constraints over the background. Our background modeling has the following assumptions

. The lighting of the background do not have considerable variations other than noise introduced by the electric lights used in the background.

. The background has no moving objects.

. The background does not occlude the foreground.

. There are no specular surfaces in the background or foreground.

These assumptions limit our background subtraction model to a constrained environment. Since our aim is to implement a gesture analyzer using a fixed set of motions, that could be tracked by algorithms implemented over distance transformed foreground, our background model serves our purpose.

3.2 Human Model

The human body is highly deformable object with many degrees of freedom and hence it is a challenging problem to model the human body or human body parts. Tracking involves, capturing the video sequence and locating the interested segment in each frames. The main objective of modeling the human body is to track the body or body parts by estimating the parameters of the model so that it fits best in a new frame. The model can be either 3D or 2D based on what we try to track and the constraints imposed. A typical 3D model has around 30 kinematic D.O.F. with supplementary parameters to include internal proportions [12]. Such 3D models are designed to offer several advantages in terms of high-level interpretation and occlusion prediction. Normally 3D models require time consuming initialization and updating procedures. There are many 2D models proposed because of its inherent simplicity and less computational overhead [6, 7]. But 2D models can function only under constrained environments such as in absence of occlusions, restricted movements, etc.

Our aim is to track human arms using distance transformed technique for the purpose of gesture analysis, where we assume the person stands in front of camera and there are no occlusions. The movements made are simple gestures defined in our dataset. So we do not use complex 3D models, instead we propose a 2D Link-Joint model to track arms.

Link-Joint model is designed to accommodate the deformities naturally occurring to the arms while making movements. We see human arms as linked pair of lines with spring like joints that connects

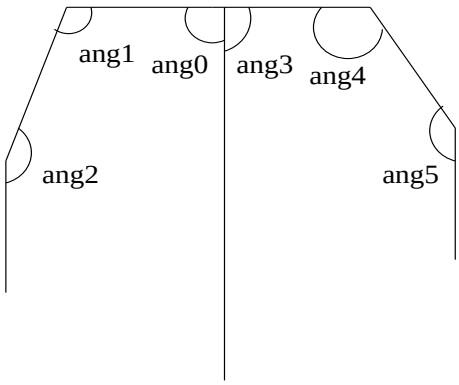

Figure 3.4: 2D model

the links as shown in Figure 3.4. The joints are designed like a spring to represent the angular movements of re-volute joints and also the change in length perceived in the perspective projection when arms are making angular movements. Figure 3.5 shows our proposes model.

Our proposed human model has seven Link-Joints as shown in Figure 3.4. They are interdependent in such a way that change in the position of any link may affect the locations of one or more links. The model has a "Base Point", which is invariable in a model and the locations of every link is defined with respect to base point. The constraint of base point being invariable restricts the geature vocabulary from having gestures that involves 3D motion patterns like walking, running, etc. The implementation details of Link-Joints and human model are discussed in the following sections.

Vector Parameters and Estimation

As shown in Figure 3.4, the LJ model is made of seven link objects and the tracker has to fit each of these links in the distance transformed image and thereby fit the whole model. The characteristics of link object is discussed in 3.2.1. The important parameters of a link object are

- Link's starting position in the image
- Angle at which it is connected to other links
- Length of the link

These parameters are estimated by minimizing the weight of the link object and are discussed in detail in the section 3.2.1.

3.2.1 Implementation

We have implemented LJ model using two objects named

1. Link Object
2. Spring Object

These two objects constitute a Link-Joint in a LJ model and a LJ model has seven Link-Joints as shown in Figure 3.5. In this section, we will discuss the characteristics of Link-Joint object and the algorithm used to fit each Link-Joint over a distance transformed image and thereby we obtain best fit for a LJ model.

Link Joint

A Link-Joint object is made up of a spring object and link object. Spring denotes the joint in the arms and it represents the angle between adjoint links. When arms are bent we see a reduction in length in the image because of projection. To handle such a situation spring has the characteristics of changing its length by fraction of its length. Hence spring is described by the following characteristics.

1. Starting position of the spring
2. Angle between two adjoint 'Links", A
3. Length of the Spring, L (Fixed)
4. Differential length of spring DX (variable)

Similarly the characteristics of a link is explained below

1. Length of the link he length of the link is fixed and the flexibility is obtained with the help of spring.
2. End point of the link t is determined by the angle 'A' of the spring and its 'DX'.

A Link-Joint (LJ) object is composed of a link and spring object. The operations of a Link-Joint(LJ) are as follows:

1. Initialize the link joint object: o initialize a link joint we need to know start point of the link, Spring Object and Length of the link.

2. Change the angle of LJ he angle of a LJ is the angle of the spring.

3. Change the length of the spring (DX) alculate Error of the Link

Each LJ has an energy associated with it. Energy of the link is defined as the sum of the pixel values along the length of the link normalized by the total length of the link including DX, in the Distance Transformed (DT) Image

Energy = $\sum$ Intensity of the Pixels along the link in binary DT image /(length of th link+DX)

We know in the distance transformed image, the axis will have minimum intensity as it is the darkest. Hence, we try to minimize the energy of each links. As observed, the parameters that determine the best fit of LJ object are 'A' and 'DX'. We have kept 'DX' also as fixed parameter for the current application and we try to track the full-body by estimating "A" for each LJ. The estimation technique is discussed in Chapter 4.

Model Object

The proposed 2D Link-Joint Model is used to represent the arms of humans standing perpendicular to the axis of the camera. We use seven LJ objects to represent the model. Each LJ has its own Degree of freedom. Please refer to the Figure 3.5 for our proposed model. The table 3.1 gives the DOF of each link. The Model object functionalities include

1. Initializing a Model
2. Enabling the Movement of individual LJ
3. Calculating Errors
4. Providing a user interface by drawing the LJ on a given Image
5. Adjusting the angles of Each LJ
6. Setting the Base Point of the Model

Link-Joint Model API

The main objective of using a object oriented design is code reuse. This is made possible by providing a API reference to the code, so that developers who want to use the model need not have to bother about the implementation details. Our L-J Model has its API reference for each of its objects namely

$LinkNum$	DOF	Desc
0	2	Right/Left
1	2	Up/Down
2	4	Right/Left Up/Down
3	4	Right/Left Up/Down
4	2	Up/Down
5	4	Right/Left Up/Down
6	4	Right/Left Up/Down

Table 3.1: DOF of Each Link

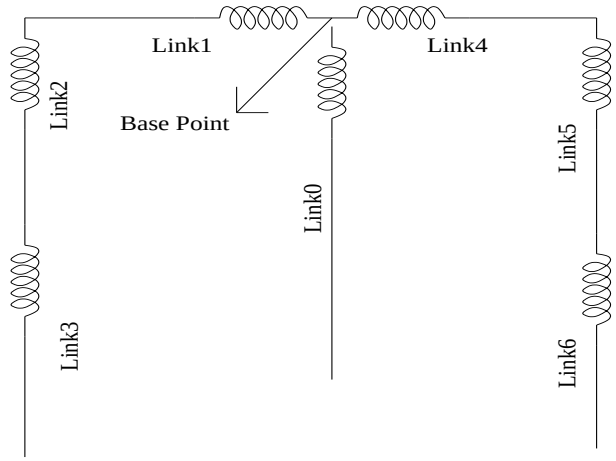

Figure 3.5: Proposed Link-Joint Model

. Spring

. Link-Joint

. Human-Model:

The API references are provided in tables 3.2, 3.3, 3.4.

The Model object has seven Link-Joint objects as its components. Hence it is intuitive that the model object API will make use of Link-Joint's and spring object's API. The API for spring object is shown in table 3.2 and Link-Joint object is shown in table 3.3. The API for human model object is shown in the table 3.4

Object	*API*	Description
Spring	Spring (double length, double angle, double DC)	Constructor
	setSpringStart (Vec start), Vec getSpringStart ()	Get and set for the start point of the Spring
	setSpringEnd (Vec end), Vec getSpringEnd()	Get and Set for End point of the Spring
	setSpringLine (), Line2D getSpringLine()	Spring Line is the line obtained from the begin and end points of the Spring. This line is needed when calculating the Error of the Link-Joint.
	setDX (double dx), double getDX()	Set and Get for DX of the Spring
	setLength(double), double getLength()	Get and Set for the length of the Spring
	double getAngle(), setAngle(double angle)	Get and set for the angle of the Spring

Table 3.2: API References for Spring Object

3.3 Assumptions and Constraints

Tracking of human or human parts is complex problem and it always involves modeling the tracking regions. One can use 3D human models to represent all deformities and can handle occlusions using more than one camera. Meanwhile, there are needs to track simple movements which does not involve occlusions and huge deformations in the images captured, where we can make use of simple 2D model. We can decide our complexity level based on the problems we try to solve. The problem that we are considering is to track the arm motion of a person right in front of the camera, perpendicular to the camera axis. We have the following assumptions.

* The person will not move back and forth.
* He will make only the movements specified in our gesture database.
* There won't be any occlusions.
* Since the experiment is performed in indoor environment we don't assume strong changes in the lighting.
* The person performs actions at normal speed. No rapid movements of arms that could not be caught by 30 fps camera.

Since our final objective is to demonstrate distance transform techniques can be used for tracking human movements and thereby to implement a gesture recognizer of fixed set of gestures, we think

Object	*API*	Description
LinkJoint	linkObject (Vec start,Spring model-spring,double modellength)	Constructor
	linkObject (Vec end,double angle, double len,double modellength)	Constructor
	void setSpringStart (Vec model_start)	sets the start point for the spring object associated with this link joint.
	Vec getStart ()	Returns the start point
	Vec getEnd ()	Returns the end point of the link joint
	void setEndPoints()	This method sets up the start and end points of the Link-Joint whenever there is a change in either spring or link
	double getLength ()	Returns the length of Link-Joint
	Spring getSpring ()	Returns the Spring Object of the Link-Joint
	double getSpringLength ()	Returns the Spring length
	double getSpringDX ()	Returns the DX of the Spring of link-join
	double getSpringAngle ()	returns angle of the Link-Joint
	void setSpringAngle (double ang)	sets the angle of the Link-Joint. One has to call setEndPoints method when angle is changed to update the link.
	Vec getSpringStart ()	Returns starting of the spring and thereby the link.
	void changeSpringDX (double dx)	Controls the DX of the Spring
	void setLinkError (double error)	Sets the link Error of the link joint
	double getLinkError()	Gives the Link Error of the link joint
	Line2D linkToLine ()	The Link-Joint has start point and end point. The line constructed from the link is used for Calculating the error associated with it and also it is drawn over the given image to help user know the position of the links.

Table 3.3: API References for Link_Joint Object

Object	*API*	Description
Model	modelObject(double *points)	Constructor, Takes 8 pairs of points as input. Each point represents the co-ordinates of the pixels at joints.
	void setLinkObject (int i, linkObject lo)	sets up the link of given index for as the given link-object.
	linkObject getLinkObject (int i)	gets the link object of given index.
	double getLinkAngle(int index)	get the spring angle of link of given index.
	void setLinkAngle (int index, double angle)	sets the spring of the given index
	Vec getSpringStart (int index)	gets the starting point of the spring of the link of given link
	Vec getLinkEnd (int index)	gets the end of the link of given index. A link-object starts from start of the spring and ends at link end.
	Vec getRoot()	gets the root of the whole model. It is the point where link0, link1 and Link4 meet.
	void setRoot (Vec rootpoint)	sets the root point
	Line2D getLinkToLine(int index)	This method gives the link line of the link of given index
	void drawRGBModel (RGB_Image image)	This method draws the model over the given RGB Image

Table 3.4: API References for Model Object

these constraints are genuine. Also a tracker cannot be computationally complex as it has to deal with frames captured over a period of time at very high rate (> 25 FPS). ur Link-Joint model is suitable for our proposed gesture vocabulary. In the following chapters we discuss tracking the human arms using our model. Our Link-Joint model has more methods than what has been used in our HCI implementation. We can make use of them as we enlarge our gesture library.

Chapter 4

Robust Tracking

A full-body gesture analyser needs to track human motion with the help of a 2D or 3D model which represents highly deformable and dynamic nature of the human body. As discussed in the review paper [10], the dynamic parts of gestures have the following three characterisctics.

. Movement,

. Activity and

. Action.

Movements are typically atomic and most primitive form of action, Activity is sequence of movements and Actions are high level entities used to describe what is happening. As we can see movement forms the fundamental entity of gesture analysis, we need an efficient and robust algorithm to track the movements of human body parts. In real time conditions, we cannot use many cameras to track as it will increase the computation complexity and in many situations it is not practical. Hence researchers of vision community have been interested in tracking using monocular techniques.
Monocular Tracking has been very active research area. The common tracking techniques include

* Having Artificial Markings [4]

* Optical Flow based methods [14]

* Limb tracking based on 3D models [5]

* Limb Tracking based on 2D Models [7]

Tracking of human arms requires the system to identify the arms in the initial frame and tracking the corresponding segment in the subsequent frames. The process of identification can be varied from primitive techniques like having artificial markings to training the system to identify the arms without any constraints, even under occlusions. The method of carrying artificial markers adds an overhead to the user while the method of training the system to identify limbs under unconstrained environments is an idealistic case. Invariably researchers apply certain constraints while implementing the trackers, while the goal is to make it as ideal as possible. There has always been a trade off between performance and complexity.

We have designed a 2D model-based tracker that uses distance transform technique for tracking. As described in previous chapters we obtain the background subtracted images and we manually draw our Link-Joint(L-J) model over the initial frame. We use our tracking algorithm to fit our L-J model in the subsequent frames. As discussed in previous chapters we model the limbs as L-J model and fitting the model for a given frames involves fitting the link parameters (namely Angle A, Position P, Spring's incremental length DX) for each link of L-J model in those frames. We use optimization techniques to fit each link to the new image by minimizing the weights of the links. Please refer to the section 3.2 for details related to implementation of Link-Joint model.

In the following section, we present the assumptions we make regarding our movements and arrive at a conclusion that it is only the angle A that needs to be fixed for each frame. In every frame, we can have the link angle incremented or decremented or remain unchanged for every new frame. Hence we have designed a differential estimator that decides whether any of the three changes mentioned above has happened. The amplitude of angle change is quantized to a step size and will be fixed using a fitter based on optimization techniques on a distance transformed foreground Image. The algorithm is discussed in detail in the following section.

4.1 Tracking Algorithm

The first step in the tracking is drawing the Link-Joint model over the initial image which we call as reference image. We have to track the movement of the arms by fitting the Link-Joint model over the subsequent frames. We make the following assumptions about the process.

. The images are captured at more than 25 fps.

. The system captures each frame and does not drop any frames.

. There are no occlusions over the foreground images.

. The human arms do not occlude each other while moving.

The term tracking means tracking the changes in the parameters of each link of our Link-Joint model. The parameters of our link are length and angle of the links. Since, in our experiments, we are not considering the stretching and bending of arms we have only the angle of the links to be tracked. We characterize the movement of arms in the following terms.

. The movement of the arms are limited to 2D-frame and we are not tracking motion in 3-D.

. The links can move with any velocity that could be captured by the camera we use for tracking. Since the angle "A" of each link is the only varying parameter that charecterizes the movement, we try to make an estimation of rate at which "A" changes for each frame and that estimated value is called "Step Size"of a link.

. The step size for each link is fixed dynamically so that our tracker is robust enough to track any arbitrary arm movements.

We estimate the step size using history of the link's movements in the previous frames. We adjust the step size by applying optimization techniques, where we try to minimize the energy of a link in a image segment called "Search region". We explain the process in the following sections.

4.1.1 Search Region

Our tracking algorithm works by fitting each link in a given frame of a video sequence and thereby we fit the L-J model. The initial parameters for the L-J model is obtained as user input. In the subsequent frames we have to estimate a search area for each links where we look to get the best fit for the link. The search area is estimated based on the angular velocity of each link, i.e. rate of which the parameter 'A' per frame . Hence the search area is robust enough to handle the dynamic changes in the velocity at which arms are moved. Figure 4.1 shows the segment of the image where we show the search region around the link bounded by green lines and the Link-Joint is marked as red line. We construct a search area by drawing a arc with starting point of the spring as center and with an angle proportional to the step size of the link.

Search area = area subtented by the arc of angle K * DA
with center as start point of the spring and radius as length of the link, where K is a constant

Figure 4.1 shows the search region bounded by green lines.

Figure 4.1: Search Region: The search regions are bounded by green lines, the red line marks the Link-joint. The foreground is represented by white pixels while the background is represented as black

4.1.2 Fixing the Step Size

The idea of tracking the changes using step function is derived from digital encoding techniques like delta modulation, pulse modulation etc. This method is used for tracking changes where the successive samples of objects being tracked do not change considerably. In this method we make a prediction assuming the step size is same as the previous sample and apply correction algorithms to adjust the difference between actual and predicted values. Thus this method is essentially a two tier technique named as prediction and correction.

Our tracker also has the same approach. The initial step of our tracker is to fix the step size for each of the links to some value, say 0.01 radians. We try to track the subsequent position within the 0.01 radians to -0.01, if we are not getting the best fit in the range we apply the correction technique which in this case is based on numerical based fitting technique called as "Golden Section Search in One Dimension" [15]. The flow chart Figure 4.2 explains the algorithm we follow to track the link. Hence the prediction technique to predict angle A of a link is as follows.

$\text{Step}_{Predict}(\text{t}_i) = \text{Step}_{Predict}(\text{t}_{i-1})$

We assume the step size will be the same as before the last frame and we select the best fit of +Step or -Step. If the weight of the link is beyond the tolerance level which is a user defined parameter, we say the error is infinite. In our case we fix the tolerance level in such a way that not more than twenty percent of the link can lie outside the foreground region. In case the weight of the link with this step is infinite, we have to apply correction technique. The idea is to always try to keep the

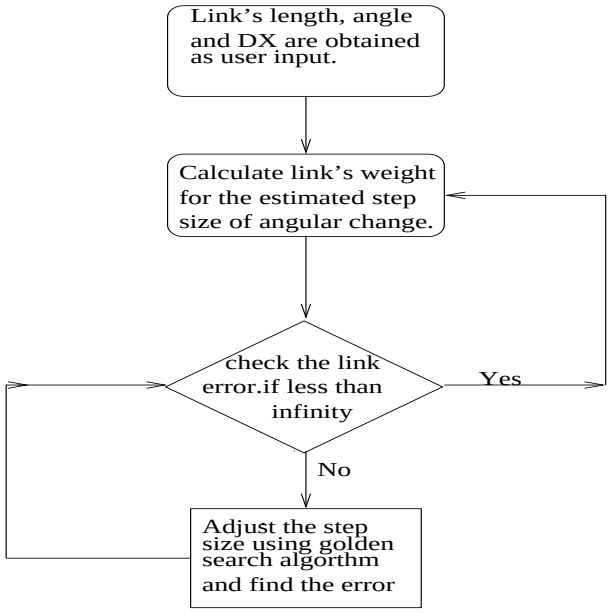

Figure 4.2: Flow chart for fixing the step size

links with in the foreground image and try to get the best fit.
Sometimes, because of failure in foreground extraction, we may not be able to find a fit in the search region. In such cases, we make use of the history of the movement patterns. We average out the step sizes in the last five frames to make an estimate about current step.

4.2 Golden Section Search

The "Golden Section Search" algorithm is similar to bisection method used to detect the roots for functions in one dimension. The root is supposed to be bracketed in interval (a,b) and one tries to find a intermediate point x between a and b to reduce the interval to either (a,x) or (x,b). This process is repeated until the bracket interval is acceptably small. In "Golden Section Search method", we try to bracket a minimum instead of a root. A root is bracketed between a pair of points (a,b) when the values of the function f at these points a and b have opposite signs. A minimum by contrast is contracted between triplet of points a, b, c in such a way that $a < b < c$ or $c < b < a$ and f(b) is lesser than both f(a) and f(b). Now we know that there has to be minimum between a and c, if the function is non singular.
The analog to bisection is to choose a point either between a and b or b and c. Suppose we make the latter choice. Then we evaluate f(x). If $f(b) < f(x)$, then the new bracketing triplet of point is

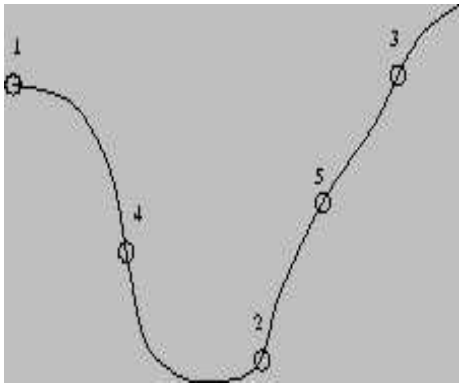

Figure 4.3: Golden Section Search with initial bracket (1,2,3) becomes (4,2,3) (4,2,5)...

(a,b,x) otherwise the new bracketing interval is the triplet (b,x,c). We continue the bracketing until the distance between the two outer points of triplet is tolerably small. Figure 4.3 shows how we go about finding the minimum of a function. This method does not need any information regarding the derivative of the function. If such information is available we can use it for fixing new point x in our algorithm.

In the L-J model, the most robust links are link numbers 2, 3, 5 and 6. The step size of those links vary considerably and the fitter algorithm fits them. Please refer to Figures 4.4, 4.5, 4.6 and 4.7 which show the varying step sizes for links 2, 3, 5 and 6 of frames taken from a sample video sequence.

4.3 Advantages

The advantage of the model lies in its simplicity and robustness. The tracker is based on 2D and it does not involve any 3D reconstruction. There are no complex calculations involved. As long as there is no rapid variation in the movement of arms we do not even require to use the golden section search. Most of the time the tracking is done assuming that the changes happen only within the $\pm$step size. It is robust in the sense that any rapid change in speed or direction of movement of the arms is tracked using our correction techniques. Our model has certain advantages over existing trackers. Many trackers use external markings to segment the parts that are being tracked. We are not using any such markings. Pfinder tracks the human arms by reconstructing a 2D blob model, while we are not reconstructing anything. This avoids computational overhead. We are not using any complex filters like particle filters or Kalman filter. Some trackers involve constructing 3D arm

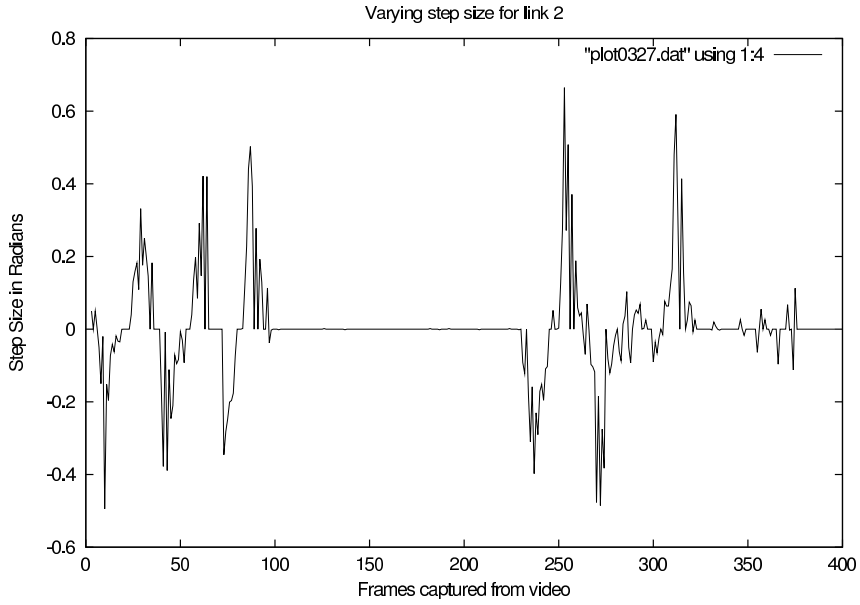

Figure 4.4: Varying step size for link 2 in radians for different frames

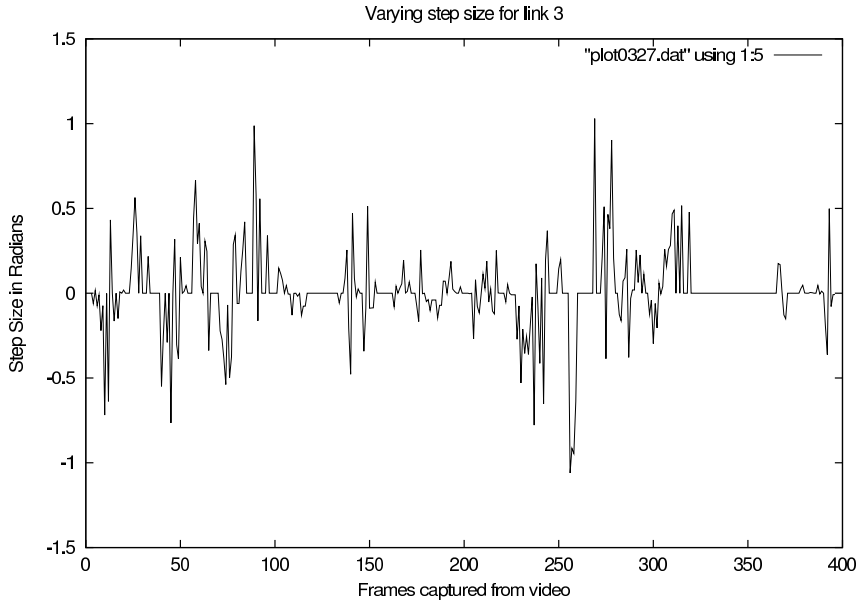

Figure 4.5: Varying step size for link 3 in radians for different frames

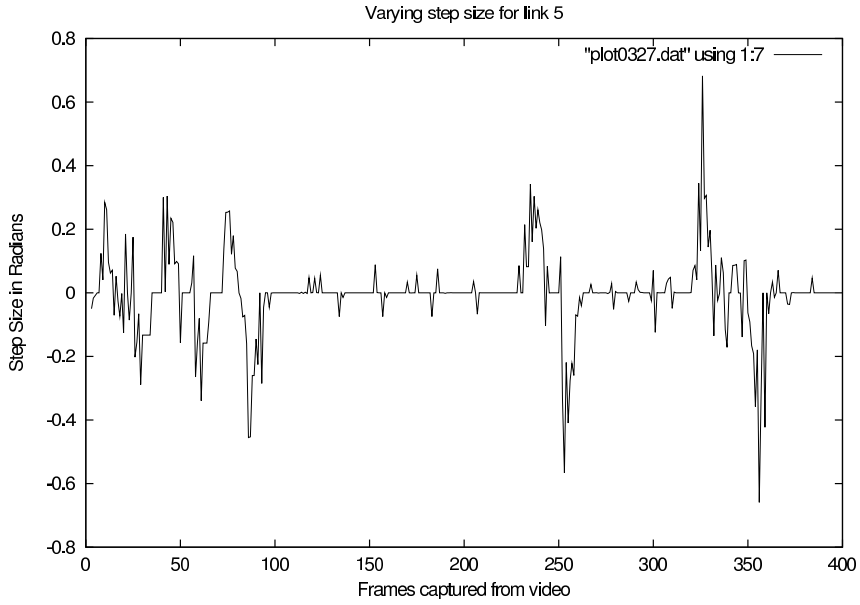

Figure 4.6: Varying step size for link 5 in radians for different frames

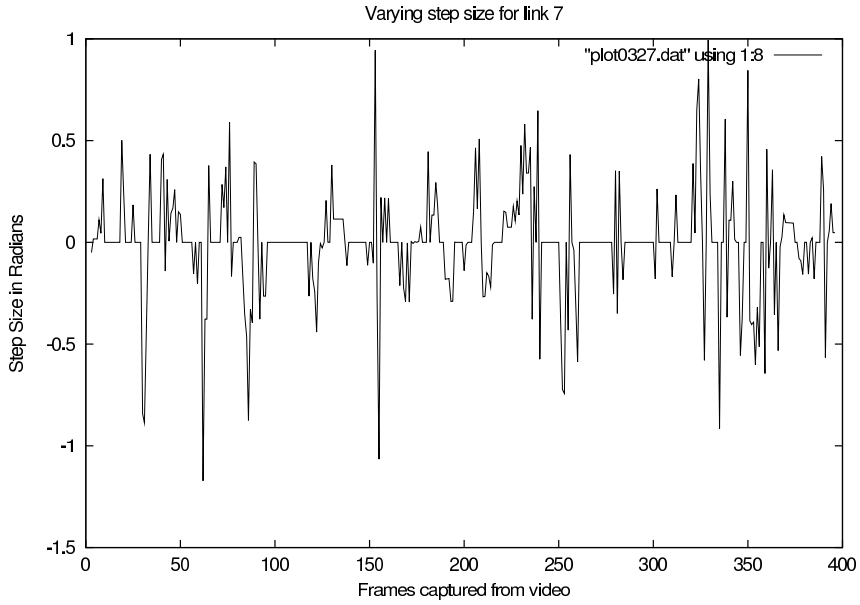

Figure 4.7: Varying step size for link 6 in radians for different frames

model which will make the system computationally complex.

4.4 Limitations

Our model makes certain assumptions about the background and makes certain restrictions regarding the movement of arms. We assume an indoor environment without sudden changes in illuminations. The only moving object in the scene is the foreground. The arms do not self occlude. The foreground objects can move only the hands and there may not be other movements like turning around or moving back and forth etc. We expect that there will be only one person in the foreground. ight now the system is not implemented for real time tracking, though it has the ability to be converted into real time. We discuss about the possibility of making our tracker real time in Chapter 7. We obtain the initial values for the parameters of L-J model as user input. Also we assume that there are no significant frame loss during the capture phase. These are some of the limitations of our proposed model. Despite of these limitaions we find the performance of the tracker satisfying the needs our goal which is to identify the gestures made in the constrained environment. However, as discussed in Chapter 7, we are working on reducing the constraints.

Chapter 5

Gesture Detection and Analysis

Gesture is defined as movement of part of the body with or without intention to mean something. Gesture forms the important aspect of interaction between man-man or man-machine. As we have discussed in the previous chapters HCI involves designing a system that can understand natural modes of communication used by human beings like speaking, writing, gestures etc in their day-to-day life. Gesture analysis system exploits the gestures made by human beings to improve HCI. Hence the primary goal of gesture analysis system is to make the computer identify the gestures and to use them to convey information or for device control.

There are variety of gestures made by human beings. Gestures can be made using face, arms, hands etc. It is not possible to list out all forms of gestures and hence their identification. It is a general approach to have a set of gestures defined for a system and the system trying to identify them. For e.g., American Sign Language (ASL) involves a set of hand gestures. Similarly we try to define a set of gestures made by movement of arms and our system tries to identify them.

A typical gesture analysis involves designing a 3D or 2D human model to track the movement of concerned body parts. The trackers output is quantized and is used for developing a gesture vocabulary. Gesture vocabulary is used for identifying the state of the arms. We have developed a state diagram that is used for identifying the gestures made. Our gesture analysis system involves the following steps,

. Design a Human Model (either 2D or 3D).

. Track the concerned body parts by fitting the model to new positions.

. Define the set of gestures that could be identified by the system.

. Develop a quantizer the quantizes the output of the tracker.

. Define a state diagram in which different gestures are reached in stages.

. Use the quantizer output to define the states of the model.

. Using the state diagram detect the gestures made.

5.1 Gesture Vocabulary

Gestures form the very important part of human communication. We make different kinds of gestures using every part of our body and it is not possible to list out each and every gestures made by human beings. Hence gesture analysis system tries to define their own set of gestures that it tries to identify. We are trying to identify the gestures made by arm movements. The set of gestures our system can identify are

. Both Arms moving up

. Both Arms moving down

. Right hand moving up

. Left hand moving up

. Right hand moving down

. Left hand moving down

. Right hand waving

. Left hand waving

. Both the hands waving

. Clapping

We identify the gestures in a hierarchical way. The upward or downward movement of arms is identified by tracking the links 2 and 5 of L-J model. The waving of the arms are identified using the tracker outputs of links 3 and 6. Any tracking is done with respect to the reference frame as described in previous chapters. We have implemented a state diagram to determine the gestures. The implementation details are discussed in the next chapter.

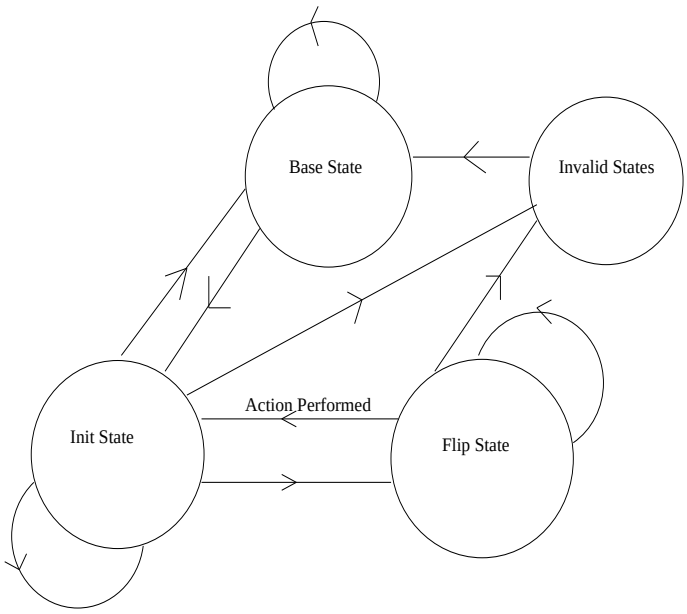

Figure 5.1: State Diagram to determine if any action performed

5.2 Implementation

The actions listed in the previous section can be classified into two types.

. Basic Actions Basic actions are complete by their own. E.g., either or both hands moving up or Down, either or both forearms moving towards or away from the body.

. Composite Actions Actions that are performed by the combinations of basic actions. E.g. Clapping hands, waving either or both hands.

A close observation of our gesture vocabulary shows that the actions are performed with either or both hands raised or lowered. Also we have to determine random hand movements which are not intended for any gestures. Hence we have to design a system that determines whether any hand movements are made with an intention to make gestures allowed in our vocabulary and should also ignore the invalid ones. We have designed a state machine model to identify the gestures which is shown in the figure 5.1. Let us explain each of the states mentioned in our state diagram

. Base State he tracker requires the human, whose arms need to be tracked to pose in the reference position. We can track either arms moving “UP” or “Down” based on the angular displacement of the links numbered 2 and 5 in Figure 3.5. The model is said to be in “Base

State" when in reference position or when we track movements that suggests the arms are near the reference position.

. InitState s mentioned before, we identify gestures that are performed with arms raised or lowered. Hence when we determine the arms are either upwards or downwards we start tracking forearm's (links 3 and 6) motion. The links 3 and 6 can either move towards the body or away from it. The initial state is described as the state where the links 2 or 5 or both are determined to have moved either upwards or downwards and the link 3 or 6 or both are established in either moving inwards or outwards.

. Flip State he flip state occurs only after the initial state is established. Flip state is described as the state in which links 3 or 5 or both move in a direction exactly opposite to that in initial state. A gesture is said to have occurred when there is a transform from flip state to initial state. In this state we can have the same gesture occurring repeatedly.

. Invalid State he invalid state is the state in which the system is not able to establish whether the arms are moving upwards or downwards. This state can be reached when the arms are moved randomly or when the arms move downwards from upper position or vice versa.

The unexplained part here is how we determine the movement of the arms like "moving up", "moving down". We have quantized the movements tracked in each frame using a set of alphabets namely a,b,c,d,e,f,g,h,A,B,C,D,E,F,G,H. The table below explains our quantization process.

The small letters of link 2 and capital letters of link 5 marks the upward movement. The small letters for link 3 and capital letters of link 6 marks the arms moving towards the body. The vice versa for the above said case also holds good for our model. Thus we have developed a language that represents the trackers output into gestures. Thus the gesture analyzer makes use of the trackers output and converts them into a set of alphabets that can convey the gestures made. Like any other system, our gesture analyzer has its own advantages and disadvantages. We discuss them in the next section.

5.3 Advantages and Limitations

The field of gesture recognition is vast and many different approaches have been tried with different goals. But we can say that any gesture recognition system typically has two entities, as mentioned in [13].

Angle in Radians	Alphabet
0.00 -.05	A
0.05 -.1	B
.1 - .15	C
.15 - .2	D
.2 - .3	E
.3 - .4	F
.4 - .5	G
> .5	H
0.00 - -.05	a
-0.05 - -.1	b
-.1 - -.15	c
-.15 - -.2	d
-.2 - -.3	e
-.3 - -.4	f
-.4 - -.5	g
< -.5	h

Table 5.1: Quantization of angular movements into alphabets

. Feature Extraction.

. Classification of gestures.

The process is explained in Figure 5.2. In this section,we discuss the advantages and disadvantages of our implementation of each stage of the processing.

5.3.1 Feature Extraction

Feature Extraction with respect to the arms involves tracking the pose of arms and the joint angles. This is normally done by using model-based approach [9], [7] or by vision-based approach [14]. Since vision-based methods are suitable only when we have more than one camera, for monocular approaches model-based techniques are mostly used. We have already discussed the advantages of using our 2D link-joint model and the simplicity with which we track the movements. As mentioned earlier, though the model help us track without the aid of external training systems or complex filter, it is limited with respect to the mobility in 3D. We track the movements confined only to the plane.

5.3.2 Gesture Classification

Feature extraction is followed by gesture classification. As suggested in [13] we can classify gestures using

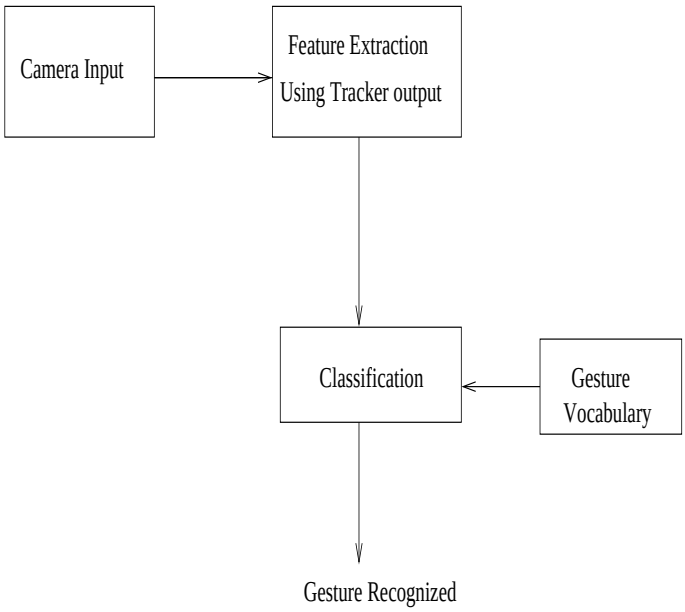

Figure 5.2: Gesture recognition Process - Block Diagram

. Rule-Based approach his approach is based on a set of manually encoded rules between feature input points.

. Learning-Based approach his approach makes use of machine learning methods like Hidden Markov Models, to find mappings between high dimensional feature sets and gestures.

We make use of the first method. We have quantized the output of the tracker into a set of alphabets and we use that to "read" the gestures performed. We have developed a gesture vocabulary that is used for recognition of gestures. This method is very simple and applicable where we have very well defined rules that marks a gesture. Learning-based approaches are computationally expensive, but they are very useful in high-dimensional feature sets. since we are not dealing with complex 3-D models, rule-based approach is best suitable for us.

5.4 Sample results

Here we are providing some of the sample images where gesture recognizer has identified some of the gestures in the vocabulary. The Figures 5.4 and 5.5 are part of the sequence where the gesture recognizer has decided that the gesture made was classified as both hands moving down. Figures 5.6, 5.7 and 5.8 are part of the sequence where a clapping action was identified. As

Figure 5.3: Initial position with arms stretched

mentioned in the section 5.2, when both hands move down and when both hands simultaneously move towards and away from the body, we classify that as a clapping action. When the same pattern is identified with hands raised high we classify that as waving action. There are occurrences where our recognizer fails totally because of loss of tracking. We will discuss such cases in Chapter 6.

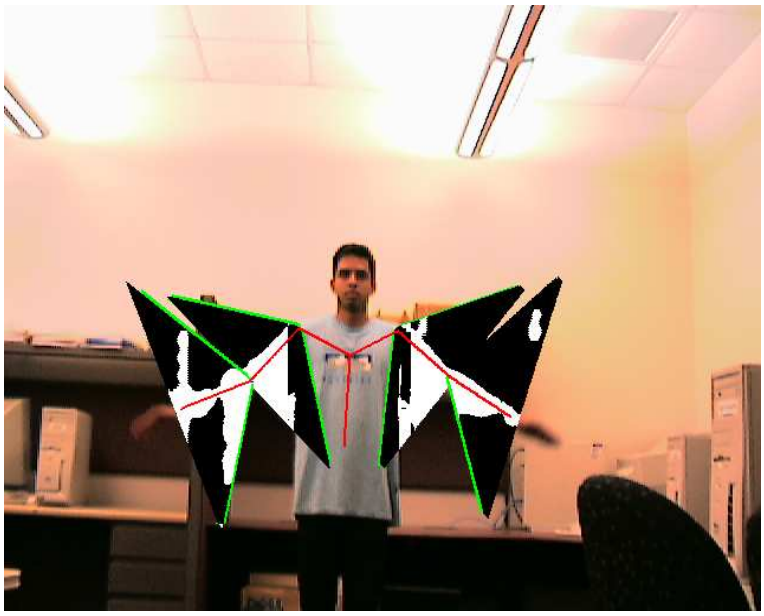

Figure 5.4: Hands Moving Down

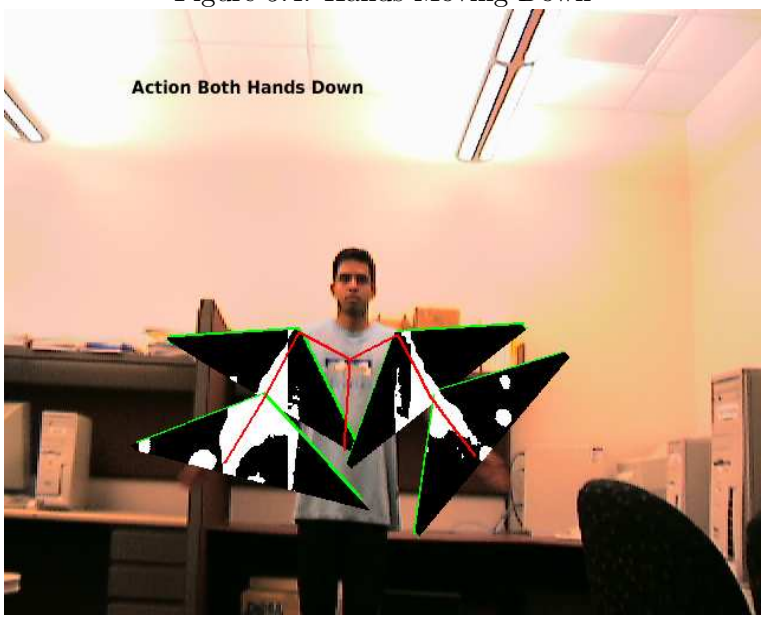

Figure 5.5: The part of the sequence where the gesture analyzer decides both hands are coming down. The red line in the image indicates the tracker, the green lines mark the boundary of the search region for best fit of the link and white area marks the search region

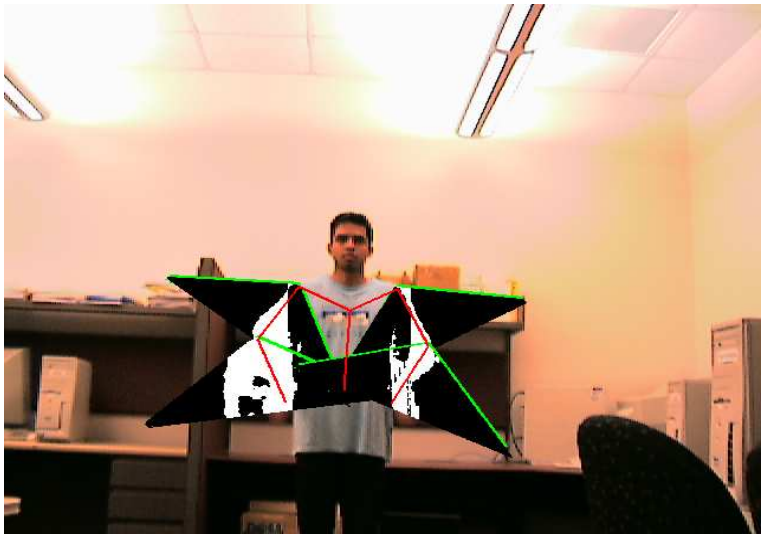

Figure 5.6: The part of the sequence where the gesture analyzer decides clapping has occurred. The red line in the image indicates the tracker, the green lines mark the boundary of the search region for best fit of the link and white area marks the search region

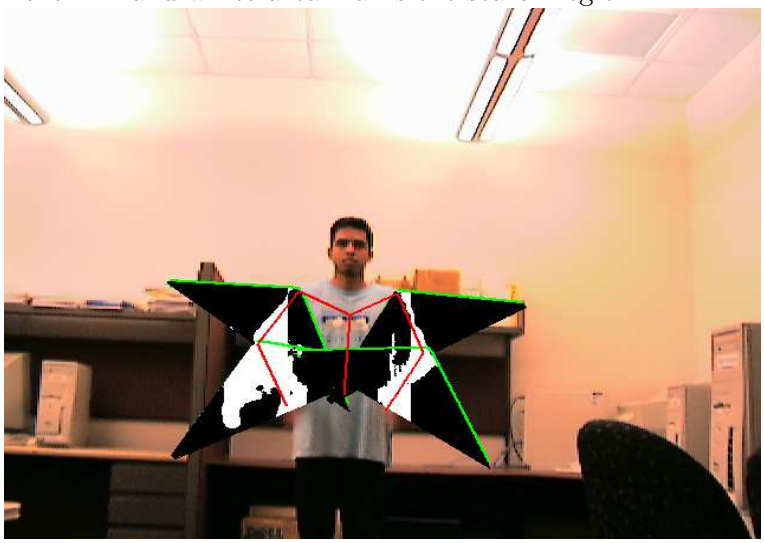

Figure 5.7: Clap action - Being tracked

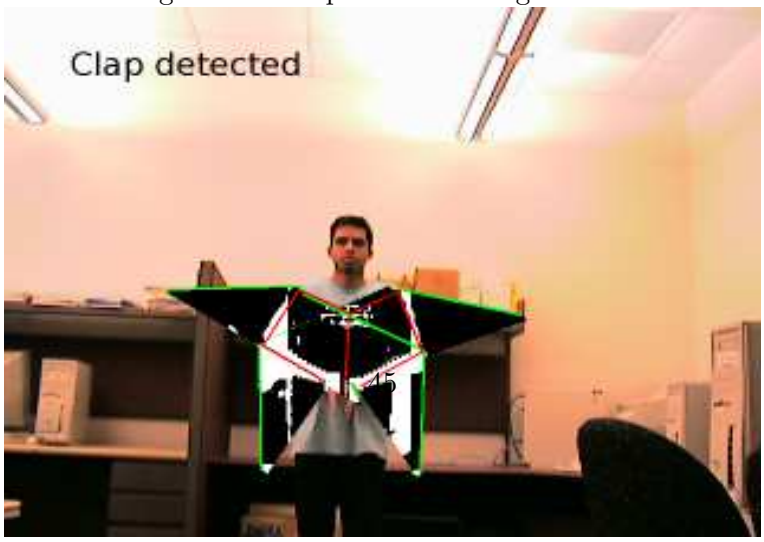

Figure 5.8: Clap detected

Chapter 6

Results and Discussions

Our approach has been tested under conditions like indoor offices and hallways, where the conditions are somewhat controlled but nonetheless are challenging. We used Sony DFW VL500 camera to collect the video sequences. The camera used had a resolution of 640x480, with a frame rate up to 30 FPS. However, higher resolution cameras with higher frame rate can lead to greater accuracy. We captured videos in an indoor office environment where we don't find many highly specular surfaces and where we don't anticipate drastic changes in the illumination. These two factors, namely specular surfaces and illumination changes can cause false positives in the background subtraction phase and thereby might affect the overall performance of the system.
Our experiments typically involve capturing a video sequence of a person performing various gestures specified in our vocabulary. There is no particular order in which the gestures have to be performed but the action sequence must start from position as shown in Figure 5.3. We grab the video sequence and analyse each frame in different stages as shown in Figure 1.1 . The output of each of these stages propagate to next level and affects the final results. However, the results of tracking plays very important role in the gesture analysis as the higher end processing are totally dependent on our robust tracker's output.
As dicussed in previous chapters, the tracker's output is greatly affected by the following factors

. Presence of specular surface in the background.

. Changes in illumination

. Angle between pose and camera.

In the next section we discuss the performance of our gesture analysis system with the help of various metrics defined.

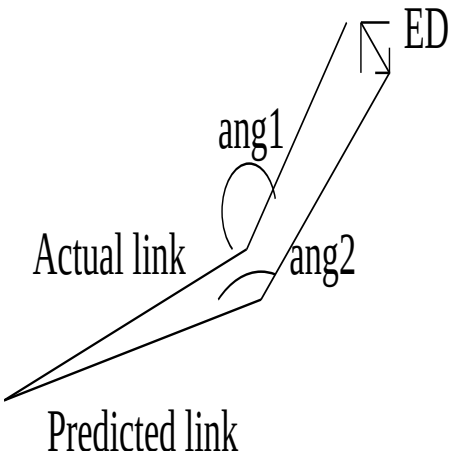

Figure 6.1: The euclidean distance between ends of the actual link and predicted link is called as End point distance (ED). Ang1, Ang2 represent angle between the links in the actual image and in the link-joint model

6.1 Error Metrics

We track full-body movements using a 2D model called link-joint (LJ) model.The system tries to fit the LJ model over each frame of the video sequence captured. We measure the error in the fit using a metric called "End Point Distance" (ED). ED is defined as euclidean distance between the end points of the links of LJ model and the ends of the joints in actual image. The Endpoint metric is best explained in Figure 6.1. The error metric is defined as sum of ED of each of the links of LJ model.

$$ED = \sum_{link=0}^{link=6} ED_{link}$$

Similarly, we define another error metric called "Angular Distance" (AD), which is difference in angle between the neighboring links of LJ model and in the actual image. This error metric is defined as sum of AD of each of the links of LJ model.

$$AD = \sum_{link=0}^{link=6} AD_{link}$$

However, each of the metric is influenced by our sources of error. We have conducted out experiments mostly based on ED as it is easy to measure it in the actual image with greater accuracy.

Figure 6.2: The background with relatively more specular objects

6.2 Experimental Results and Analysis

The gesture analysis system consists of various sub systems as shown in Figure 1.1. Hence the end result is definitely influenced by the outputs of

- Background detection
- Tracker
- Quantizer of the tracker output
- Gesture recognizer

However, we find tracking has the most influence on the end results than any other system. We also find tracking is greatly hampered by the presence of specular objects in the background.

Specular Background

We studied the effect of specular background by performing certain experiment in two different environments. We capture a video sequence where a person makes a set of gestures specified in the gesture vocabulary, in two different background environments where one had very little specular object and another had relatively more specular objects as shown in the Figures 6.3 and 6.2.

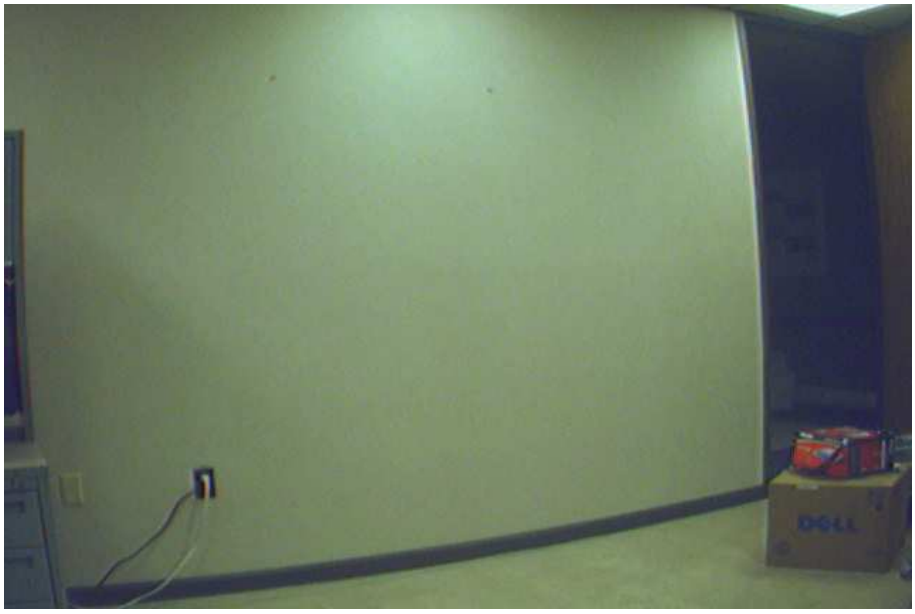

Figure 6.3: The background with relatively lesser specular objects

We have plotted the results in terms of ED observed in these environments. We can easily observe from Figure 6.5, how the presence of specular objects induce errors in the tracking and there by affecting the overall performance.

Angle between camera and pose

One of the requirements of our gesture analysis system is that plane of the camera is parallel to the plane of pose. However, our system is robust enough for small changes in the angle. To study the impact of the change in the angle of the pose, we conducted an experiment where same gesture was performed but at different angles between the camera and pose. We have deduced that the safety region falls between zero degree and ten degrees as shown in the Figure 6.4

Confusion Matrix

As with any recognition system we must try to find the accuracy of the system using a confusion matrix. To construct this matrix we ran the experiment over 600 frames in a environment which has less or no specular objects, with pose being parallel to the cameras. We found our system is very reliable as suggested by the below confusion matrix table.

From the confusion matrix we find that the false alarm rate is bit higher than normal standards,

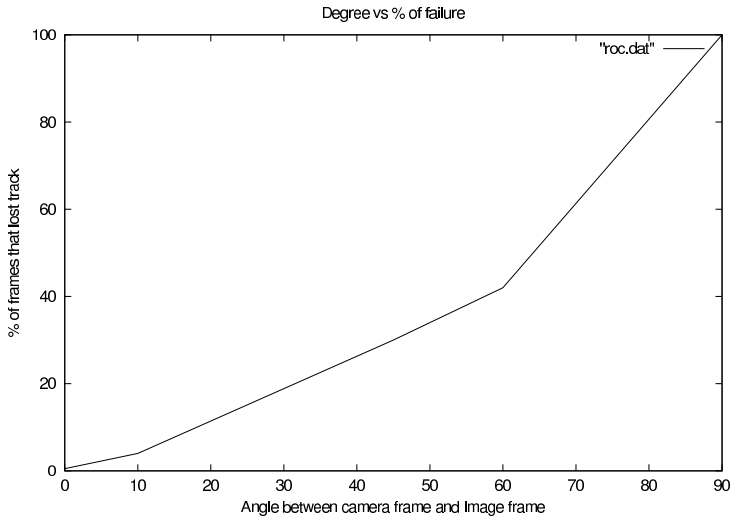

Figure 6.4: The angle between the camera and pose has an impact on the performance. Also we see the error increasing exponentially

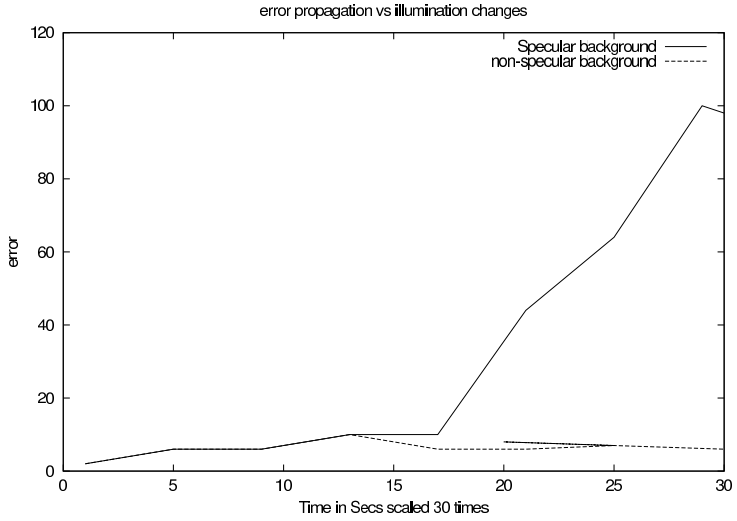

Figure 6.5: We can see that performance is highly improved when there is lesser specular objects in the background

observed/Predicted	True	False
True	.91	.09
False	.25	.75

Table 6.1: The confusion matrix of gesture analyzer constructed over 600 Images captured from video sequence

otherwise it has very good accuracy.

6.3 Performance

As discussed in our previous chapters the gesture analyzer is implemented using a model based tracker. The Model we used is a 2D Image based model developed using distance transform techniques. The performance of the gesture analyzer is totally dependent on the tracker's outcome. Hence it is appropriate to study the results of our tracker in detail to estimate the performance of the gesture analyzer.

Our tracker implementation does not perform any calibration or reconstruction to match the observed arm with the model. This reduces the complexity of the model but can lead to failure when 2D tracking fails. It is important that that the failure modes of the tracker are well understood.

The user is required to exhibit the initial position as described in Chapter 3, with his arms stretched out to initiate tracking. We have asked the users to make the allowed movements for over a few seconds to until few minutes. We have tried to capture every kind of action described in our vocabulary. Depending upon the tracker's output gestures are identified.

We have discussed in detail, the tracker's characteristics and results in Chapter 4. However, given its importance, we will throw more light on its performance in the context of full system implementation. s discussed in the previous chapters, the system estimates the foreground by subtracting the background which is modeled as a Gaussian distribution. We capture the images with the resolution of 640x480 pixels at the frame rate of 30 fps. However, the system could be operated under higher resolution and frame rate, but at increased costs of processing them. In particular, The background subtraction and distance transform are very costly pixel based processing which are time consuming.

We tested the application using a Intel Pentium M 1.5 GHZ processor, as one of the aims is to implement the tracking applications in laptops that are used for wide area surveillance in closed room environment. However, we have found that the processing time for each frame takes around 10 secs, that is ill affordable for real time implementation. The time can be reduced by intelligent

use of resources like using threads and multi-processing. For example when the same process is run using two threads the processing came down to 8.071 secs. Since our aim is to establish the capability of distance transforming technique as a tool for tracking, this work did not concentrate on reducing the processing time at this stage. Nevertheless Chapter 7 throws more light on the ways to reduce the processing time of this stage of the system.

Robustness

Another characteristic of the tracker that needs mentioning is its robustness. As mentioned in Chapter 4, the output of background subtraction affects the tracker's performance. The background detector, because of lighting variations, misclassifies objects. This leads to incorrect tracking in certain frames of a sequence as shown in Figure 6.6. The system can recover itself if the errors in background detection algorithm does not persist over a significant period of time. We achieve this by preventing the links from overlapping on the other links by using repellent method. The repellent method adds more weight to the areas closer to other links so that the minimization functions will not fit into them. Of course, the tracker looses track if the lighting variations are too high or we have specular surfaces in the background that affects the foreground detection. However, we find that the system recovers itself from the loss of tracking in the consequent image frames as shown in Figure 6.7.

Dependence of Gesture Analyzer on Tracker

In section 5.4, we have some of the sample sequences where we had the gestures that were identified successfully. However, there are cases where gesture analyzer fails to detect the gesture. The failure occurs because of lose of tracking of the arms. Figure 6.11 demonstrates one such case, where we show a part of a sequence where the system failed to identify the gesture made.

Also we find an interesting observation from the confusion matrix as shown in table 6.1. We find that whenever the system has identified a gesture, it is highly accurate. However, when there is a conclusion of "No Action", the results are highly unreliable as we observe in the confusion matrix the "No Action" results are true only with the probability of .75. Also we attribute gesture classification failures mostly to failure of the tracker, because the analyzer is based upon reading the alphabets from the tracker, which are nothing but quantized variations of LJ parameters.

Figure 6.6: Specular surfaces seen at the background affects the tracking. As we see the red line that marks the LJ model fit over the foreground image has lost track. The marked white regions represents search area for each links to look for a fit, however in the above images the search areas do not contain the foreground regions we are interested in.

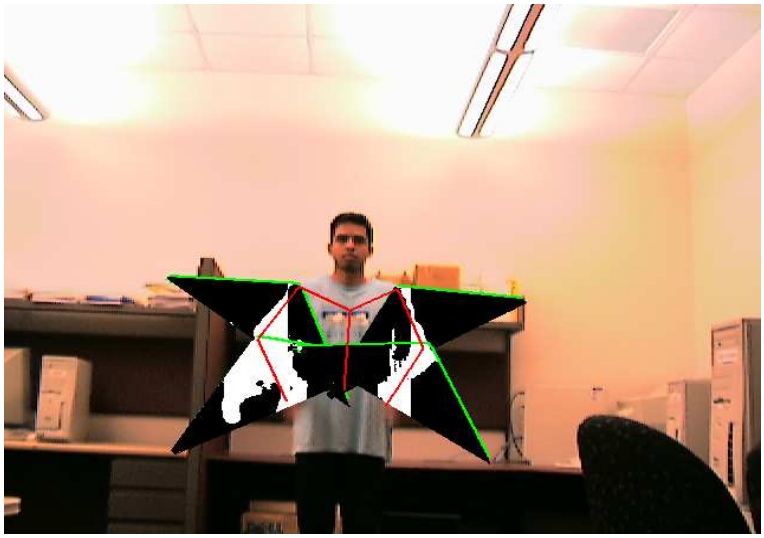

Figure 6.7: We can recover the tracking in case noise introduced are not persistant over time

Figure 6.8: The part of the sequence where the gesture analyzer decides no action has occurred. However, the person is making the 'clapping' gesture. The red line in the image indicates the tracker, the green lines mark the boundary of the search region for best fit of the link and white area marks the search region

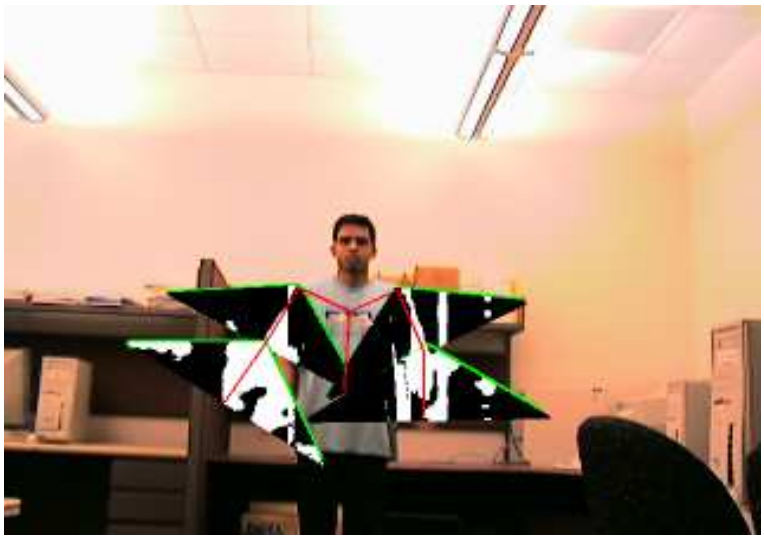

Figure 6.9: Lose of tracking while Capturing Clap Action

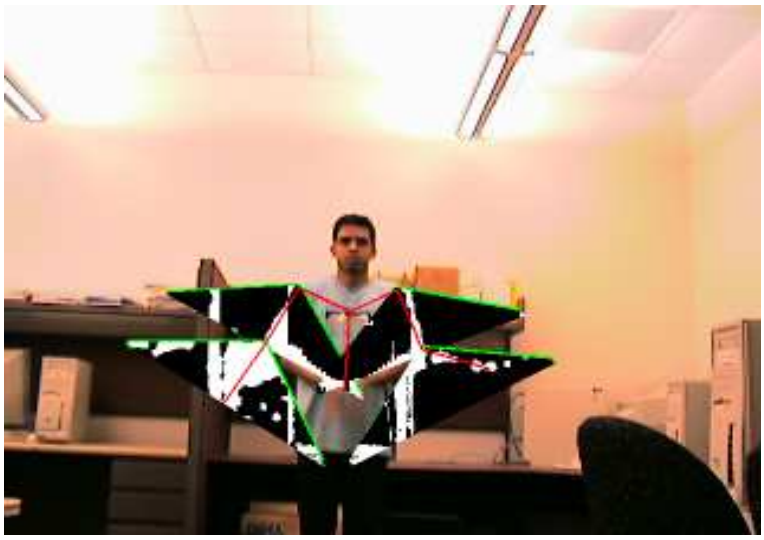

Figure 6.10: Lose of tracking while Capturing Clap Action

Figure 6.11: No Action Reported when a Clap was performed

6.3.1 Summary

As shown in Figure 1.1, our gesture recognizer is built in four stages . The implementation details and performance of each stage are discussed in previous chapters. We have proved the capability of distance transform

tool to track human beings and we find the implementation successfully tracking the arms. Still, we have to improve the system to track in real time and analyze the gestures made. In the following chapter we will suggest the future work needed to make this system more reliable and complete for real time applications.

Chapter 7

Future Work

Our objective is to implement a full-body gesture recognition system using a tracker which is based on distance transform technique. We find that the system is capable of recognizing the gestures specified in our gesture vocabulary. However, as discussed in Chapter 6 the performance of the system does not meet the requirements of real-time implementation. Our future work aims at making the system works in real time. As discussed in the earlier chapters, we know the tracker forms the backbone of the system and the tracker uses a 2D Link-Joint model for tracking. We have discussed the implementation of L-J model in the chapter 3. Our future work aims at addressing the limitations of L-J model and thereby make it more robust. As discussed in Chapter 3, the Link-Joint Model is made up seven Link-Joint objects. The tracking is done by fitting the parameters of each link in the frames of captured video sequence and the process is explained in Chapter 4. We can see that fitting of links are inter-dependent and the error in fitting one link affects the other links. We have to design a method which detects such lose of track and initializes the tracking by re-estimating the "Base Point" of the L-J model, either by alerting for user input or by some automated methods. ince a misfit occurs mostly because of noises introduced from background subtraction sub-system, there is more possibility of tracker losing track if there is a specular background. Even though, as discussed in Chapter 4, the tracker can recover itself from momentory lose of tracking, it fails to track in case the noise due to background subtraction persists over a period of time. Our future work involves designing a adaptive system, that can recover from prolonged loss of tracking. This can be acheived by using a more dynamic background model as discussed in [1]. he current system tracks does not allow the user to bend his arms. Our L-J API has methods that can handle the changes in the length of the links in 2D projections caused by the bending of the arms. Our future work aims to add more degree of freedom to the links. The model requires the user to be standing at a place and start with a initial gesture. If there is a shift in the reference plane because of

involuntary movements, there is possibility of errors in the foreground detection. Over the period of time the errors can accumulate and cause loss of tracking. Our future work involves designing a adaptive foreground detector using expectation maximization (EM) techniques. Also we want to add more gestures to our gesture vocabulary so that it could be used for different purposes.

7.1 Towards Real time

The implementation of our gesture recognizer has timing constraints that prevents it from being a real time tracker. Our aim is to develop a tracker that makes use of distance transforms. But distance transform being pixel based operation, it is expensive to compute in real time. Hence it is highly imperative to find faster algorithms for distance transforming the images. However we find the work of Mohamad Seaidoun's research on distance transform [16] provides us a solution for this problem. Our future work will involve integrating his algorithm into our system.

As we know our gesture analysis system performs the foreground extraction and distance transforms the foreground using one sub system and tracks the object using another sub system. However we can integrate the two sub systems by using more adaptive background models.Our future work aims at integrating the two steps into one less time consuming process by using back ground modeling discussed in [1] and DT technique discussed in [16].

We have successfully demonstrated the ability of distance transform technique as a tool for tracking and have implemented a gesture analyzer based on our tracker. But still we are few steps away from making it real time implementation. Our future work is directed towards that goal.

Bibliography

[1] Quanren Xiong and Christopher Jaynes, "Multi Resolution Modeling of Dynamic Scenes Using Weighted Match Filters" IEEE Conference on Multimedia, 2004.

[2] Omar Javed , Khurram Shafique and Mubarak Shah, "A Hierarchical Approach to Robust Background Subtraction using Color and Gradient Information", IEEE Workshop on Motion and Video Computing, Orlando, Dec 5-6 2002

[3] C. Stauffer and W.E.L. Grimson," Adaptive Background Mixture Models for Real-time Tracking." In CVPR99, pages II:246–252, 1999.

[4] N.Alberto Borghese and Paolo Rigiroli, "Tracking Densly Moving Markers", Proc. IEEE First International Symposium on 3D Data Processing and Transmission, Padova giugno 2002, pp. 682-685.

[5] Bullock, D., Zelek J.S., "Automated Model Initialization for 3D Monocular Visual Tracking of Human Limbs in Unconstrained Environments", WSEAS Transaction on Computers, Nov 2004 pp.1604-1610

[6] Ivan Laptev and Tony Lindeberg, "Tracking of Multi-state Hand Models using Particle Filtering and a Hierarchy of Multi-scale Image Features" Technical report CVAP245, ISRN KTH NA/P–00/12–SE. Department of Numerical Analysis and Computer Science, KTH (Royal Institute of Technology), S-100 44 Stockholm, Sweden, September 2000.

[7] Christopher Wren, Ali Azarbayejani, Trevor Darrell, Alex Pentland, "Pfinder: Real-Time Tracking of the Human Body", IEEE Tran. on Pattern Analysis and Machine Intelligence, 19(7), Jul. 1997, pp. 780-785.

[8] D.Gavrila, "The Visual Analysis of Human Movement:A Survey", CVIU, Vol.73, No.1,pp.82-98, 1999.

[9] L. Goncalves, E. Di Bernardo, E. Ursella, P. Perona, "Monocular Tracking of the Human Arm in 3D", Proceedings of the Fifth International Conference on Computer Vision, pp. 764, 1995.

[10] Ying Wu, Thomas S. Huang, "Vision-Based Gesture Recognition: A Review", The 3rd Gesture Workshop, Gif-sur-Yvette, 1999.

[11] L. Bretzner, I. Laptev and T. Lindeberg, "Hand Gesture Recognition Using Multi-scale Colour Features, Hierarchical Models and Particle Filtering", in Proc. 5th IEEE International Conference on Automatic Face and Gesture Recognition Washington D.C, May 2002.

[12] Cristian Sminchisescu, Bill Triggs, "A Robust Multiple Hypothesis Approach to Monocular Human Motion Tracking", Rapport de recherche de l'INRIA numro RR-4208 , Jun 2001

[13] Konstantinos G.,Derpanis, "A Review of Vision-Based Hand Gestures",International Report, Feb 2004.

[14] Ross Cutler and Matthew Turk. "View-based Interpretation of Real-time Optical Flow for Gesture Recognition", IEEE International Conference on Automatic Face and Gesture Recognition, April 1998, Nara, Japan.

[15] "Numerical Recipes in C : The Art of Scientific Computing" by William H. Press, Brian P. Flannery, Saul A. Teukolsky, William T. Vetterling.

[16] Mohamad Seaidoun "A Fast Exact Euclidean Distance Transform with Application to Computer Vision and Digital Image Processing", PHD Thesis, 1993.

Vita

Date and Place of Birth: Madurai, India, October 11, 1976

Professional Positions: Assistant System Engineer
Tata Consultancy Service
Mumbai, India, 1999–2001

Programmer Systems Analyst Senior
Ag. Communications Services/CALE Lab
University of Kentucky
Lexington, Kentucky, USA, June 2005 – till date