

# **Categorising the Abnormal Behaviour from an Indoor Overhead Camera**

**A PROJECT REPORT**

*Submitted in partial fulfilment of the requirement for the award of the degree of*

**BACHELOR OF TECHNOLOGY  
IN  
ELECTRONICS AND INSTRUMENTATION**

*By*

**GURKIRT SINGH**

**(06BEI080)**

*Under the Guidance of*

**Dr. R. B. Fisher**

**University of Edinburgh**

**Dr. P. Arulmozhivaraman**

**VIT University**



**SCHOOL OF ELECTRICAL ENGINEERING**

**VIT University**

**VELLORE – 632014, Tamil Nadu, India**

**June 2010**

# ACKNOWLEDGEMENT

I am grateful to my **Division leader** and the **Director of SELECT**, for giving me the permission to carry out this project.

I would like to sincerely thank **Dr. Bob Fisher** for giving me the opportunity to carry out my project work in the **School of Informatics at University of Edinburgh**. I express my heartfelt and deepest gratitude to **Dr. Bob Fisher** for his guidance through the completion of the project and for providing us with the necessary inputs. I would also like to thank Steven and Michal for their help and support.

I would like to thank my guide, **Dr. P. Arulmozhivarman** for his guidance and co-operation.

I extend my gratitude to all the teachers who have ever taught me. Without the knowledge that they imparted over the 7 semesters, completion of this project would not have been possible.

Finally I want to thank my parents for their help and supporting morally and financially to carry out my work in Edinburgh.

**Gurkirt Singh**

**(06BEI080)**

*To my parents and family*

*My father Mr. Rakha Singh and mother Mrs. Gobind Kaur and my Dear  
nephews and niece.*

## **Declaration**

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

## **Abstract**

This dissertation describes a B.Tech project for which the purpose was to develop a system that could be used for automated surveillance. The main novelty is the use of a vertical camera. The project investigates whether such a system can effectively detect the moving objects, track their trajectories, and use these to recognise anomalous events. A vertical camera is used to capture continuous video for detection, which involves low level of image processing to detect the objects and stores their properties and positions. The tracking program is used to track the person and form a trajectory from the detector output file. The tracking program produce the file of tracked persons trajectories, all trajectories have different lengths depending upon how much time the person stays in view of camera. Normally a person stays in view of the camera for 10 to 15 seconds. To describe each trajectory with an equal number of attributes Spline fitting algorithm is used, which gives the six control points for each trajectory. The detection of anomalous behaviour is described by different number of parameters like error in spline fit, vector distance from closest mean vector and multivariate Gaussian probability.

# Table of Contents

i.	List of Figure .....	vii
ii.	List of Table .....	ix
<b>Chapter 1 Introduction.....</b>		<b>1</b>
1.1	Motivation.....	1
1.2	System Objective .....	2
1.3	The Forum.....	2
<b>Chapter 2 Background .....</b>		<b>4</b>
<b>Chapter 3 System Overview .....</b>		<b>5</b>
3.1	Detection .....	5
3.2	Tracking .....	8
3.3	Representing Trajectories as Spline .....	8
3.4	Abnormality Detection .....	8
<b>Chapter 4 Tracking.....</b>		<b>9</b>
4.1	specification .....	9
4.1.1	Merging and Splitting .....	12
4.1.2	Disappearing .....	13
4.1.3	People as Separate Blobs.....	13
4.2	Design .....	13
4.2.1	Corresponding Person and merging .....	13
4.2.2	Splitting .....	15
4.2.3	Splitting .....	15
4.2.4	People as Separate Blobs .....	15
4.2.5	Core Algorithm .....	16

4.2.6 Calculations .....	22
4.2.6.1 Velocity .....	22
4.2.6.2 Predict Position .....	22
4.2.6.3 Radius .....	23
4.2.6.4 Probabilities .....	24
4.2.6.4.1 Prediction Error Probability.....	24
4.2.6.4.2 Histograms Probability .....	25
4.2.6.4.3 Angles Probability .....	25
4.3 Removing Bad Trajectories .....	26
4.4 Implementation .....	29
4.5 Output File Description .....	30
4.6 Evaluation .....	30
<b>Chapter 5 Representing Trajectories as Spline .....</b>	<b>35</b>
5.1 Specification .....	35
5.2 Design .....	35
5.2.1 Choosing Control Points .....	35
5.3 Implementation .....	37
5.4 Output File Description .....	37
<b>Chapter 6 Abnormality Detection .....</b>	<b>39</b>
6.1 Overview .....	39
6.2 Building the Model .....	40
6.2.1 Calculations .....	41
5.3 Choosing Parameters .....	44
5.4 Implementation .....	51
5.4 Results .....	51

**Chapter 7 Conclusions and Future Work ..... 53**

**References ..... 54**

### **List of figures**

S.No.	Index No.	Name	Page No.
1	1.1	The view from the camera.	2
2	1.2	Entry and exit points	3
3	3.1	Detected points on 4 <sup>th</sup> January 2010	7
4	4.1	An example of two persons merge	10
5	4.2	An example of two person split	10
6	4.3	An example of a person disappearing and another person appearing at the same time	11
7	4.4	An example of one person being detected as several blobs	11
8	4.5	A simple scenario of people walking together	12
9	4.6	Bhattacharyya distances between histograms of the same and different people.	14
10	4.7	The distribution of prediction errors	25
11	4.8	The marginal area where trajectories have to start and end.	28
12	4.9	Block Diagram of Tracker program	29
13	4.10	Single person trajectory	31
14	4.11	Example of merging and splitting condition	31
15	4.12	Example of disappearing	32
16	4.13	Tracked object for 4 <sup>th</sup> January 2010.	33
17	4.14	Image plot of all the detection points after tracking using all trajectories (65529).	34
18	5.1	Variation of Median fitting error with number of control points	36

19	5.2	Spline curves for 4 <sup>th</sup> January 2010	38
20	6.1	An example training cluster	40
21	6.2	Variation of number with the variation of spline fit	41
22	6.3	Variation of vector distance to closest mean vector.	42
23	6.4	Variation of numbers on each log(probability	43
24	6.5	Variation of false positive and false negative at different values of spline fit error threshold	45
35	6.6	Variation of false positive and false negative at different values of spline fit error threshold	46
36	6.7	variation of false positive and false negative at different values of log(probability) threshold	46
37	6.8	Examples of false negative trajectories	47-48
38	6.9	Examples of false positive trajectory	48-49
39	3.10	Clusters those are modelled (107)	51
40	3.11	Shows four different clusters for different paths	52

## List of Tables

S.No.	Index No.	Name	Page No.
1	4.1	The failure rate of correct identity	32
2	6.1	Shows equal error rate for each parameter	47
3	6.2	Thresholds values at best equal error rate for training.	47
4	6.3	Shows number of trajectories in all 169 clusters present from total 65529 trajectories	49



# Chapter 1

## Introduction

### 1.1 Motivation

Video surveillance has been used for many purposes like crime prevention, traffic monitoring, transport safety and industrial process monitoring. There are up to 4.2million CCTV (Closed Circuit television) cameras in Britain - about one for every 14 people in 2006 [1]. These cameras can be used to gather information by observing the behaviour of objects and for security reasons or for other process. However, the main use of surveillance systems is to provide a safer environment through detection or even prevention of dangerous situations and crimes. These cameras are used at public places to provide help and safety in case of any accident, suicide attempts, assault, thefts or fight. CCTV cameras have other benefits also like reducing the fear of crime and helping police to obtain evidence in criminal cases. After installation of CCTV cameras in Glasgow city centre overall crimes fell by 3156, crime of violence by 230, fire raising by 57, offences of petty assault, breach of peace and drunkenness by 272, and vehicle offences by 318 [2]. According to same source CCTV cameras has different perception by peoples 72% thought CCTV would prevent crime and disorder, 81% thought it would be effective in deterring perpetrators and 79% thought it would make people feel they would be less likely to become victims of crime.

A large number of cameras are needed to ensure the good coverage of area [3]. One operator can watch at most four screens at a time, so good surveillance require a lot of operators to watch multiple screens. And operator needs breaks, so continuing surveillance is impossible with a small number of operators. Also sometime operators suffer from boredom which decreases their level of attention [4]. Manual surveillance systems are more expensive and vulnerable to errors. Therefore it is beneficial to improve visual surveillance by automating some of the tasks performed by operators. In particular, abnormal or suspicious behaviour should be detected and the operator's attention directed to particular video clip [5].

## 1.2 System objectives

The project investigates whether such a system can effectively detect moving objects, track their trajectories, and use these to recognise anomalous events in the Informatics Forum area. A camera pointing vertically downwards towards the ground floor is used to capture continuous video for detection. A tracking program is used to track the person and form a trajectory from the detector output and produce trajectories of persons. The trajectories are used as input to the final model which detect the abnormal behaviour. The system was implemented to ensure the efficiency to cope with change in environmental conditions, also problems with detector. The system was built on two platforms, C++ and Matlab.

## 1.3 The Forum

The Forum is one of the University of Edinburgh buildings located on the Central Campus. The camera is mounted on the ceiling, camera pointing vertically downwards towards the ground floor. An example image obtained from the camera is shown in Figure 1.1. The image covers most of the main hall.



Figure 1.1: The view from the camera.

There are many entry and exit points in Forum, those are shown in image given below figure 1.2. Entry and exit points are given serially: { Auditorium big gate, cafe, stairs, elevator, night exit door, robot lab small door, robot lab big door, vision lab door, servitor box, reception, Main entry (11+12), auditorium small door }. These entry and exit points will be used in chapter 6.

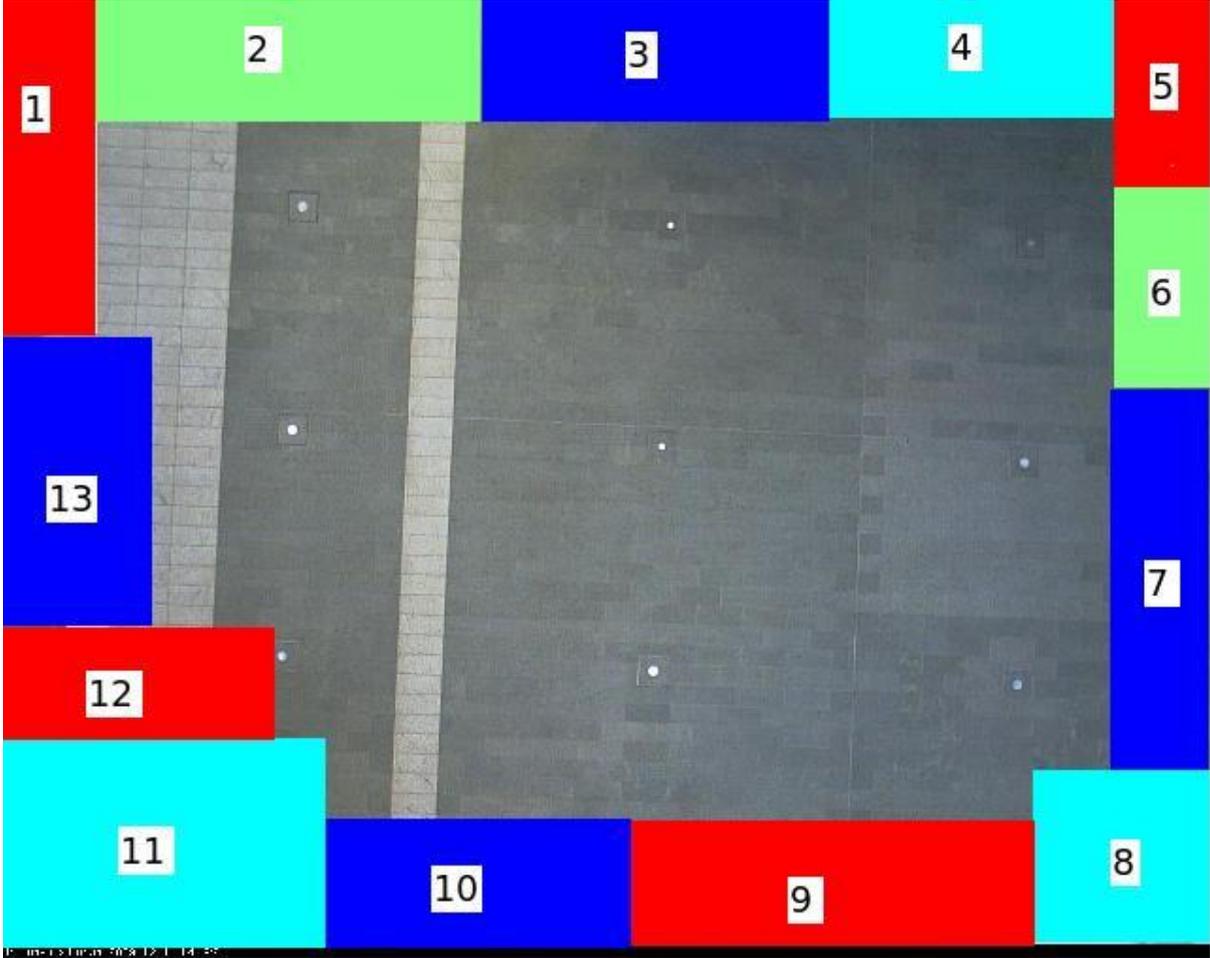


Figure 1.2 Entry and exit points.

# Chapter 2

## Background

This project is improvement to previous work by Barbara Majecka [4] at University of Edinburgh. Automated surveillance has been a very popular research field in recent years. A vast range of techniques have been developed for different parts of surveillance systems. There are many techniques proposed by different people.

Incorporating techniques based on local motion descriptors can provide more precise information about the types of actions performed by a given target (e.g. running or fighting [6]). Techniques based on trajectories are the most appropriate for this project. Among them, the simplest are those that form a geometric representation of the raw trajectory data [7]. In [8], the full trajectories are approximated by cubic spline curves with seven control points. In this way, each trajectory is represented by the same number of – the control points and the duration of the object’s existence in the scene [7].

I start the work from tracking. Collection of detection data was started by Barbara Majecka. There are many changes to the tracker which are explained further in this thesis.

# Chapter 3

## System overview

The project consisted of four subtasks:

1. Detection of moving objects
2. Tracking objects
3. Representing trajectories as spline
4. Abnormality detection

Each of these subtasks was implemented as a separate component of the system.

### 3.1 Detection

The efficiency of the detection application was crucial because it determined the maximum capture rate of video footage that would be used by the whole system. The greater this rate, the more accurate the tracking could be, allowing the detection of anomalous behaviour to be more successful. In order to allow the frequent capture of live images and to minimise the amount of data stored, the detection and tracking processes were carried out separately. The detection focussed only on the extraction of the basic information, including bounding boxes and colour histograms of the detected objects. At each new frame, this information was used to update an appropriate file.

#### Detection overview

The detector component was developed by Barbara Majecka as her MSc project. A brief summary of the detector is described here. The purpose of the detector is to segment each image obtained from the camera into two sets of pixels: foreground and background. The background is not completely constant throughout a day. The main changes to the background are caused by:

- Reflections from the ground (due to sunlight in building)
- Shadows (due to artificial lighting near staircase)

- Changes in the ambient lighting (weather, time of day, and use of artificial light sources and their position)

There were many methods compared to choose the best one to overcome all these problems. Those methods are as following.

- Background subtraction
- Constant background updating
- Background subtraction using chromaticity coordinates
- Background division using chromaticity coordinates
- Principal Component Analysis

The background subtraction technique performed very badly. The constant background updating technique copes with changes in overall lighting conditions and constant objects. Unfortunately, this method has some problems too. The background subtraction using chromaticity coordinates was not able to cope with shadows. The background division using chromaticity coordinates performed similar to the previous technique. The principal component analysis technique gave very good results so it was chosen.

The model of the background image was built by using the Principal Component Analysis. A set of 50 background images was gathered and used to build the mean image by using Principal Component Analysis. This technique required to choose two parameters: the number of eigenvectors used in the model and the threshold. These parameters were chosen by number of experiments.

Greater efficiency of the detection process was achieved by implementing it as a multithreaded C++ application. There were different tasks in detection process: Fetching image, Obtain a binary image, Label image, Append frame information to output file. To implement these tasks different threads were used. More can be read in [4]. The detector produced one output file for each day. The following description is from the project web page summary.

**Description of output file [7]:** Each file contains one or more header lines: BEGIN TTT where TTT is the "number of seconds since (00:00:00 UTC, January 1, 1970)". Then, for each frame thereafter in which a target is detected there is a line F M T where T is the time since the start of this file in 0.1 second units and M is the number of the downloaded frames since the start of the program. Due to occasional detector program crashes, there may be more than one BEGIN statements and even the occasional reset without the BEGIN, which can be seen by the M and T values restarting. For each frame, there are one or more detected blobs. Each blob is encoded on one line in the file in the

form: *[blob id]: [number of pixels] [x\_center] [y\_center] [x\_top\_left] [y\_top\_left] [width] [height] HISTOGRAM*. The blob ids are notional and the same target in the next frame may have a different blob number. The number of pixels is a count of the pixels that are detected as being foreground inside the bounding box. The (x\_center,y\_center) is the center of mass of the foreground pixels. The bounding box is defined from the pixel (x\_top\_left, y\_top\_left) at the top left with the given width and height. The color histogram bin order is rgb : 000, 001, 002, 003, 010, 011,012,013,020,...,033,100,...,133,200,...233, 300...333, where indices 0,1,2,3 cover the ranges given above. So bin 032 means red range 0, green range 3 and blue range 2.

Figure 3.1 shows the detected objects on 4<sup>th</sup> January 2010 which will be used later in this thesis.

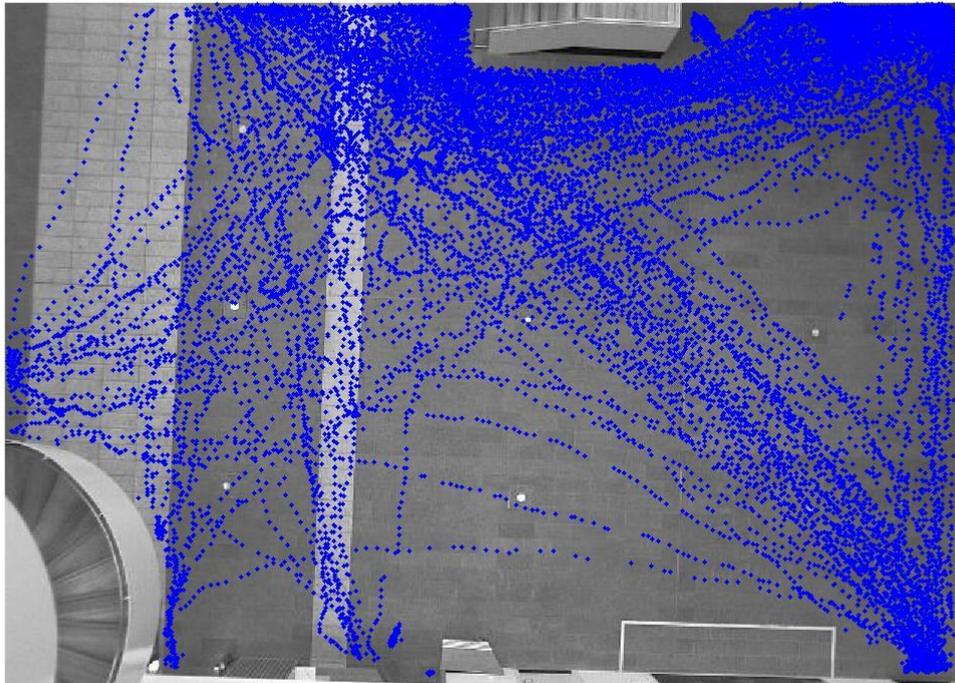


Figure 3.1 shows raw detected objects on 4<sup>th</sup> January 2010.

### 3.2 Tracking

The files written out by the detection process are used as input for the tracker to infer trajectories of each object. The tracker needs to deal with different scenarios, including merging silhouettes of people. Also, it has to cope with imperfect detections. The trajectories constructed contain sequences of position coordinates of a particular object together with times at which the coordinates were sampled:

$$\{(x_1, y_1, \text{time}_1), (x_2, y_2, \text{time}_2), \dots, (x_N, y_N, \text{time}_N)\}$$

The full sequences have different lengths  $N$  due to the different lifetimes of objects within the scene. The tracking process is described in chapter 4.

### 3.3 Representing Trajectories as Spline

The trajectories are approximated by cubic spline curves and represented by the vectors of their control points of same length. The average error of the spline fit to the tracked trajectories was calculated to decide the number of control points. When the behaviour of a person is very much abnormal, and then based on average error of spline fit to tracked trajectory, we can say it is abnormal behaviour. But this approach cannot detect all abnormal behaviours, So we need more parameters to check the abnormality of trajectory, those are described in next component. The fitting spline process is described in chapter 5.

### 3.4 Abnormality Detection

This process describes the modelling of detection of abnormal trajectories. Each trajectory is represented by 6 control points. And clusters were chosen from entry and exit points. For each cluster a mean vector and covariance matrix was determined to calculate the probability for each trajectory belonging to the cluster. Vector distance was calculated for each trajectory to the closest mean vector or mean vector of the belonging cluster. Thresholds were chosen experimentally for vector distance, logarithm of probability and spline fit error. Four conditions were chosen to detect the abnormal trajectory. Based on these conditions trajectories were flagged as anomalous.

# Chapter 4

## TRACKER

### 4.1 Specification

The detector output file is fed to the tracker process to get trajectories from detected objects. Each trajectory is represented by a sequence of centre positions with time when the object was detected at this position. To detection of abnormal behaviour time is very important. Different trajectories can have different lengths even if they start at same entry point and ends at same exit point due to different lifetime of trajectory.

The tracker component makes a distinction between the notions of a person and a blob:

**Person** is a tracked real object that appeared in the scene.

**Blob** is an area of 8 point connectivity labelled as foreground by the detector.

Each person has a unique identity based on the size and colour histogram. Deciding which person corresponds to each blob in a given frame is not a trivial task. There are several problems that the tracker has to overcome:

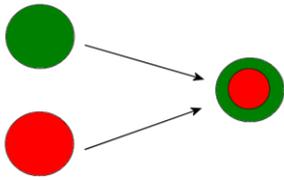
1. Several people could merge together when walking side by side, and therefore they could be represented by a single blob (Figure 4.1).
2. A group of people, represented in one frame as a single blob could split in the next frame producing multiple blobs (Figure 4.2).

In each figure

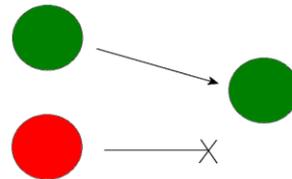
Circles represent blobs (which are inputs to tracker component).

Colours represent people's identities.

Arrows represent the movement of person from one frame to another

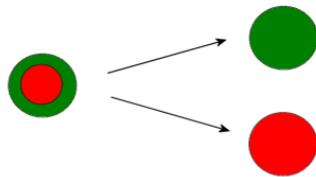


(a) Actual scenario: two people merge.

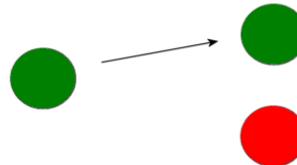


(b) Failed tracking: the red person is judged to have disappeared.

Figure 4.1: An example of two persons merging (figure copied from [4]).



(a) Actual scenario: Two people split.



(b) Failed tracking: Red blob is judged as new person but person was there in last frame, here tracker should copy trajectory of green to red and assign red to a new person .

Figure 4.2: An example case of two persons split (figure copied from [2]).

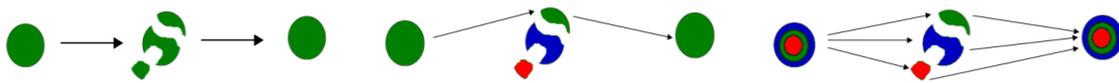
3. For some people the detection process could fail completely causing the person to disappear for a few frames (Figure 4.3).
4. Some parts of a person's body may not be detected resulting in the person being represented by several disjoint blobs (Figure 4.4).



(a) Actual scenario. Green person not detected in second frame.

(b) Failed tracking: The blob that appears in the second frame is chosen as green but that should be the new person. In the third frame green is assigned as new person. So tracker should allow a person to disappear for a few frames.

Figure 4.3: An example of a person disappearing and another person appearing at the same time (figure copied from [4]).



(a) Actual scenario: a single person is detected as separate blobs in the second frame.

(b) Failed tracking: the second frame is assumed to contain three people, two of who appeared only in that frame?

(c) Failed tracking: all Three frames are assumed to contain three people. In the first and the third frames the people are merged together.

Figure 4.4: An example of one person being detected as several blobs (figure copied from [4]).

### 4.1.1 Merging and splitting

Occlusions possibility is very less due to perpendicular positioning of camera but not zero there is very less occlusions is still present. These situations are the most common on the edges of the scene. Therefore, it is possible that people could enter into the scene as merged (two or many people represented by a single blob) and then split into several blobs when they come closer to the centre of the scene and again merged together in a single and once again split into many at the end. Such a situation is depicted in Figure 4.5. Figure 4.2b shows an undesirable tracking behaviour: the merged people are recognised as a single person, and a new blob that appears after the splitting is assigned a new identity with a trajectory starting in the middle of the floor. The tracker should not allow this; it should try to reason about where the new blob came from and create new person with same trajectory, so you have two trajectories of these two person and last points are added those are different.

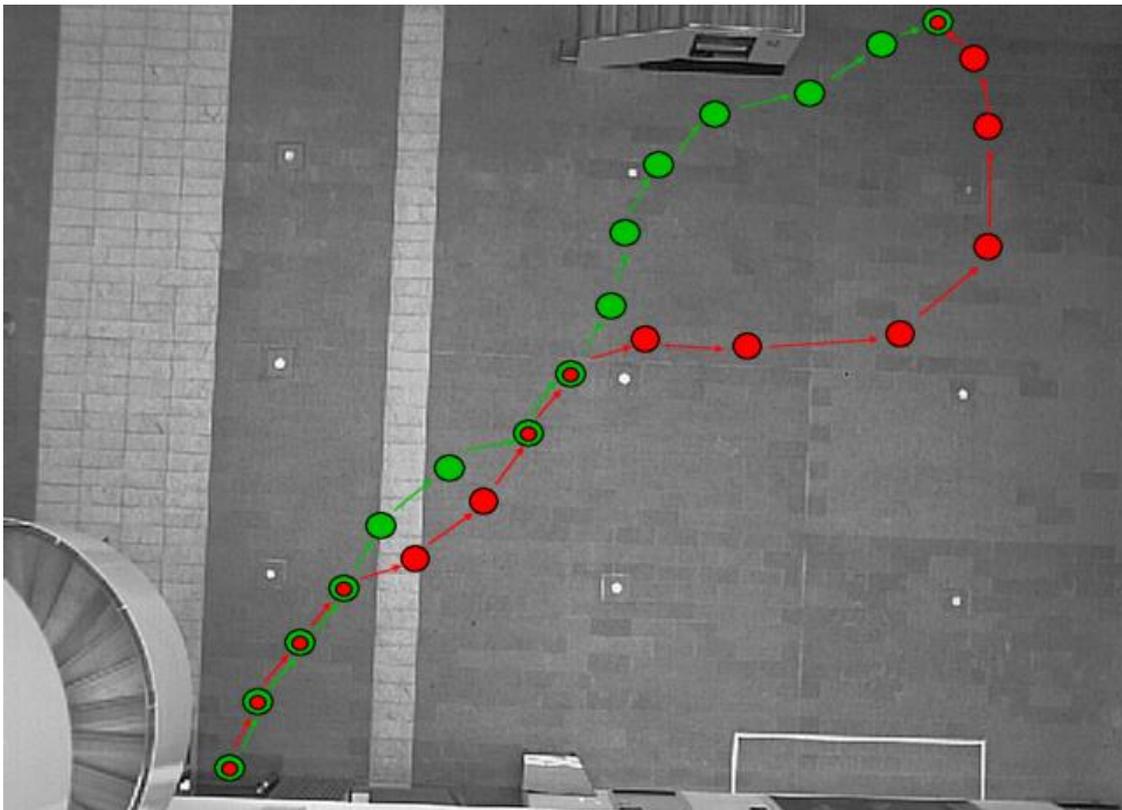


Figure 4.5: A simple scenario of people walking together. Two people enter the scene merged together. They split before middle, merge again, walk together, split again and walk away from each other, and then merge again before leaving the scene[Figure copied from 4].

### 4.1.2 Disappearing

It is showed in evaluation of the detector [in [4]] component sometimes a person is not detected for a few frames. The tracker should be able to cope with this problem. It should allow a person to disappear for few frames (4 or 5). It should not terminate the trajectory of person as if it disappears for very few frames. Also tracker should identify the correct person when that person appears after disappearing for a few frames. It should terminate a trajectory if it decides that there is a high probability that the person has left the scene or if the person has been disappeared for a very large number of frames.

### 4.1.3 People as separate blobs

12% of people are represented by the detector as disjoint blobs [4]. This causes several problems. If these blobs are recognised as separate people then the merging and splitting rules will be applied. This way, many redundant trajectories could be produced. There would be more than one trajectory instead of one trajectory which will cause detecting the abnormal behaviour and also increase the size of output file. Only one among these trajectories should be preserved.

## 4.2 Design

The tracker keeps a list of all the people currently being tracked. It updates their trajectories one frame at a time by processing the file provided by the detector. Because one person can be represented by multiple blobs and one blob can represent multiple people, an M-to-N relationship has to be preserved in the program. Therefore, at every frame, a list of detected blobs is kept as current frame data, and for each blob is stored a list of the people represented by it. A copy of all current blobs is kept as in current frame data copy variables to use in merging condition.

### 4.2.1 Corresponding Person and Merging

Check the every blob for people currently being tracked finds the corresponding blob for that from the current frame data for every frame. For each new frame data person perform the following tasks.

- A. Get Velocity
- B. Predict Next Position
- C. Predict Radius

After calculating velocity, expected position and radius, we check all the blobs one by one. For each blob from current frame, calculate the Bhattacharyya distance between the person's histogram and the blob's histogram. Then check if the blob is in the area of person the being tracked by using expected position and predicted radius. Also check if Bhattacharyya distance is less than a threshold. The threshold is chosen from a number of experiments as in figure 4.6. I have chosen the threshold 0.48 to allow the variation in Bhattacharyya

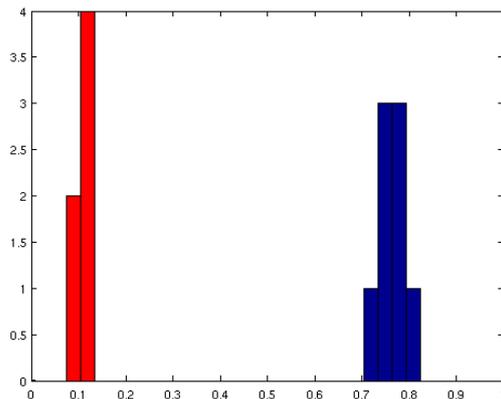


Figure 4.6- Comparison of Bhattacharyya distances between histograms of the same and different people. The former is shown in red and the latter in blue colour (figure copied from [4]).

distance due to every time update in histogram of person being tracked. The most probable person is chosen by calculating the probability for each blob that is are in area of the person and for which the Bhattacharyya distance is less than the threshold.

When a person merges with any other person in that frame we will have less blobs. In this condition we check all the blobs again for the person who has merged. Person who has merged with other person will seem to have disappeared, So we will check all the blobs only if the trajectory of the person has 6 or more points to ensure that the person is being tracked from the last 6 or more frames because the person who has disappeared could be appeared just in or two frames. Sometimes one person can produce two or three blobs as in problem number 4 (figure 4.4) as in that condition we can't say the person has merged.

After choosing the corresponding blob we need add the blob to trajectory and update the person's properties and histogram. Properties include [Average size, Average width, Average height ] and the colour histogram is averaged to write out at the end, and another histogram is updated to use finding the corresponding person and Bhattacharyya distance.

## 4.2.2 Splitting

Two people, merged in the previous frame and split into separate blobs in the next frame is the splitting condition. Problem number 2 describes it very well with figure 4.2. So every time when adding a new person from the remaining blobs that are not chosen by any person, check all remaining blobs if they had split from other persons. If, yes then copy the trajectory of the corresponding person to the blob which is added as new person as the trajectory of the corresponding person is copied at the end when writing the trajectory of the new person. When a person splits then we have two persons with the same trajectory and then the last points are added to the trajectories of these two Persons, so we have two Persons with exactly the same trajectory but the last points of these trajectories differ.

## 4.2.3 Disappearing

To overcome disappearing, I allow the person to disappear for 4 frames and a maximum of 1300 milliseconds and also check if the predicted position is out of boundaries of image. In this case if the person has disappeared, then we will write its trajectory to a final variable after checking if trajectory is right.

## 4.2.4 People as separate blobs

A single person can be detected as multiple blobs, which results in a new person added to the list of current persons being tracked. But in the next frame person or after two frames the person is again detected as a single blob. This will lead us to a merging condition. As described earlier if the trajectory of a person has 6 or more points than only the merging condition is applied and all blobs are checked again. This mean the new person will not find any new point to add to the trajectory, so automatically these trajectories will not pass the right trajectory test which means a trajectory should have a minimum of 15 points to write it out.

## 4.5 The Core Algorithm

The high level algorithm is illustrated by Pseudo code 1.

Where:

Trajectories of the current persons being tracked are stored in `curr_Data`.

Histograms of current persons being tracked are stored `curr_Histogram`.

Properties of current persons being tracked are stored `curr_Properties`.

Current frame blob's [CentreX, CentreY, Frametime] as `curr_frame_Data`.

Current frame blob's histogram as `curr_frame_Histogram`.

Current frame blob's properties as `curr_fram_Properties`.

➤ `curr` – stands for current.

Track is final variable where we store all good trajectories. This is written to a file along with them final Properties and their final averaged Histogram

❖ Code of tracking can be downloaded from [9]  
<http://homepages.inf.ed.ac.uk/rbf/FORUMTRACKING/>.

---

**Pseudo code 1: The tracking algorithm (high level)**


---

```
%%%%%%%%%% Program Starts %%%%%%%%%%
```

```
Initialization of variables
```

```
FOR EACH frame in file
```

```
curr_frame_Data = getframedata();
```

```
    %%%%%%%%% ---- Step 1---%%%%%%%%
```

```
    FOR EACH person in list
```

```
        %%%-- Step 1.1 --%% see next page for more details
```

```
        chosen = Findcorrespondingblob( curr_Data, curr_frame_Data );
```

```
        If chosen
```

```
            Addcorrespondingblob(curr_Data, curr_frame_Data(chosen));
```

```
        End
```

```
        %%%-- Step 1.2 --%%
```

```
        ElseIf IsItTime_to_Write_Trajectory
```

```
            Save to Track (curr_Data)
```

```
        END
```

```
        %%%-- Step 1.3 --%%
```

```
        ElseIf IsItMerged
```

```
            Find_and_add_blob(curr_Data, curr_frame_Data_copy);
```

```
        END
```

```
    END
```

```
%%%%%%%%---- Step 2 ---%%%%%%%%
```

```
FOR EACH remaining blob // those are not chosen by any person
```

```
    FindCorrespondingPerson(Remaining_curr_frame_Data, curr_Data );
```

```
    MakeCopyOfCommonPoints();
```

```
END
```

```
%%%%%%%%---- Step 3 ---%%%%%%%%
```

```
FOR EACH person who has disappeared
```

```
    Remove(curr_Data)
```

```
END
```

```
END
```

```
%%%%%%%%---- Step 4 ---%%%%%%%%
```

```
WriteOutFile(OutFilename,Track, Properties, Histogram)
```

```
%%%%%%%% Program Ends %%%%%%%%%%
```

---

---

**Pseudo code 2 Step 1.1: Choose corresponding blob**


---

```
%%%%%%%% Step 1%%%%%%%%
```

```
FOR EACH person in list
```

```
    Velocity = getvelocity(curr_Data)
```

```
    PredictedPosition = predictPos(curr_Data,velocity,frametime)
```

```
    Radius = PredictRadius(curr_Data Velocity,frametime,MinRadius)
```

```
    FOR EACH blob in current_frame_data
```

```
        HistDistance=Bhattacharyya(curr_Histogram,curr_frame_Histogram);
```

```
        IF Isinarea(PredictedPos,Radius, curr_frame_Data) && HistDistance<threshold
```

```
            Probability=getProbability(curr_Data,PredictedPos.....);
```

```
        END
```

```
    END
```

```
    Chosen= getMostProbableblob(probability);
```

```
    %%%% adding chosen blob to person's trajectory %%%%
```

```
    IF chosen
```

```
        curr_Data=Addcorrespondingblob(curr_Data, curr_frame_Data(chosen));
```

```
        curr_Histogram=UpdateHistogram(curr_Histogram, curr_frame_Histogram);
```

```
        curr_Properties=UpdateProperties(curr_Properties, curr_frame_Properties);
```

```
    END
```

```
%%%%%%%%/ Step 1 continued %%%%%%%%%
```

---

---

**Pseudo code 3 Step 1.2: Saving trajectories**


---

```

%%%%%%%%%%/ Step 1 continued %%%%%%%%%%%
Time = frametime-Person_last_point_time;

ELSEIF ( PersonHasDisappeared && Time >Maxtime && IsOutOfbound(curr_Data))

    IF personSplitted

        Curr_Data=add( CommonPointsBeforeSplitting, PointsAfterSplitting)
            %% common points are from variable these were copied in splitting part.
        END

    IF(IsTrajectoryRight(curr_Data))

        Track{h,1} = curr_Data

        Properties(h,:)=curr_Properties;

        Histogram(h,:)=averaged_curr_frame_Histogram;
        h=h+1;
    END
    %% Removing the blob from list, So it will not be chosen by other persons
    %% and increase the speed of program.
    Removeblobs(curr_frame_Data(chosen))
END

%%%%%%%%%%/ Step 1 continued %%%%%%%%%%%

```

---

---

**Pseudo code 4 Step 1.3: Two persons merged**


---

%%%%%%%% Step 1 continued %%%%%%%%%

ELSEIF (TrajectoryLength>5 && Time>Maxtime )

FOR EACH blob in current\_frame\_Data\_copy

HistDistance=Bhattacharyya(curr\_Histogram,Curr\_frame\_Histogram\_Copy);

IF IsInArea(PredictedPos, Radius,curr\_frame\_Data\_copy  
&& HistDistance<threshold)

Probability=getProbability(curr\_Data, PredictedPos.....)

END

chosen= getMostProbableblob(Probability);

END

%%%%%%%% adding chosen blob to person's trajectory %%%%%%%%%

IF chosen

curr\_Data=Addcorrespondingblob(curr\_Data,  
curr\_frame\_Data\_copy(chosen));

curr\_Histogram= UpdateHistogram(curr\_Histogram,  
curr\_frame\_Histogram\_copy);

curr\_Properties=UpdatePproperties(curr\_Properties,  
curr\_frame\_PrOpperties\_copy);

END

%%%%%%%% Step 1 Ends %%%%%%%%%

END

---

---

**Pseudo code 5 Step 2: Two person splits and adding new person**

---

```
FOR EACH remaining blob    // those are not chosen by any person
    FOR EACH person in list
        HistDistance=Bhattacharyya(curr_frame_Histogram, curr_Histogram);
        IF IsInArea(LastPointOfTrajectory,20,current_frame_Data_copy
                    && HistDistance<threshold2)
            Probability=getProbability(curr_Data, PredictedPos.....)
        END
        chosen= getMostProbableblob(Probability);
        MakeCopy(chosen);
    END
    AddNewPersons(curr_frame_Data)
END
```

---

## 4.2.6 Calculations

There are many calculations in the process of tracking. All are described here.

**4.2.6.1 Velocity.** Every time a new point is added to a trajectory, the person's instantaneous velocity  $V$  is calculated at that point. Velocity is calculated as a weighted average of short velocity and long velocity.

$$\text{Short velocity: } V_s = \frac{\text{Pos}(n) - \text{Pos}(n-1)}{\text{time}(n) - \text{time}(n-1)}$$

$$\text{Long velocity: } V_l = \frac{\text{Pos}(n) - \text{Pos}(n-3)}{\text{time}(n) - \text{time}(n-3)}$$

$$\text{Velocity: } V = \left| \frac{V_s + V_l * 3}{4} \right|$$

In this way, I minimise the error caused by the “jumping” centroid point. This error is especially noticeable when different parts of a person's body are detected at each frame causing their position, as described by the centre of mass of all their pixels, to change significantly. At the same time, by choosing a small number of previous points, I still allow for rapid changes in the trajectory. This is important for detection of behaviour abnormalities. When a person walking normal both short and velocity are same, which is more often observed condition, but when stops and after spotting some time starts then short velocity comes plays good role because this will be used to predict the next position.

When a trajectory has only one point the velocity assigned 0, and when a trajectory has less than four points then the starting point is used in long rather velocity than pos(n-3).

**4.2.6.2 Predict position.** Having the instantaneous velocity of the last point in the trajectory, the expected position is estimated as follows:

$$\text{Pos}(n) = \text{Pos}(n-1) + V(n-1) \times (\text{time}(n) - \text{time}(n-1))$$

The smaller the time difference, more accurate the predictions. After a longer time, the predictions could be inaccurate, and I therefore allow a person to disappear only for four frames. And most of the time prediction is very good. If trajectory has only one point then Pos(n) remain same as first position as velocity is 0.

### 4.2.6.3 Radius.

The position prediction is not always accurate, therefore the prediction area should be large enough to allow for those errors, but at the same time small enough to avoid mistakes with other blobs. I chose the radius on the basis of the previous instantaneous velocity of the considered person. If the velocity of a person is known, the radius is chosen to be FACTOR = 1.5 times longer than the distance from the last position to the next predicted position.

---

#### **Pseudo code 6** Choosing the radius.

---

```

Rafius= getRadius(Minraduis,minraduis,trajectory)
{
IF (trajectory.hasOnlyOnePoint (trajectory))
    Radius =30;
ELSE
    Radius = velocity * time * FACTOR
    IF (velocity< 1.5 Radius < 7)
        Radius = 7;
    ELSE IF ( velocity<2.65 & Radius<12)
        Radius = 14;
    ELSE IF (Radius<22)
        Radius= minRaduis1;
    ELSE IF (Radius<36)
        Radius = MinRaduis;
    END
END
}

```

---

I experimentally determined the maximum speed at which a person can walk: MAX\_SPEED = 26 pixels per 100ms [ 4 ]. Before the actual velocity of a person is known (i.e. when there is only one point in trajectory) the maximum velocity is assumed, so I assigned the radius 30 in very starting. When person walks fast, in few cases centroid point jump it could lead the point to shift nearly more than 30 pixels, so I used the minimum radius (MinRadius) 36. Third conditions is used to avoid mistakes when person walks at normal speed about 10 to 20, so I used chose the minimum radius 25 (MinRaduis1) in this condition. First two conditions are very useful to avoid stationary objects to be the trajectory of any person.

#### 4.2.6.4 Probabilities:

The Bhattacharyya distances  $H_{new}$  from the figure 4.6 which show that there is a clear distinction between the people based purely on their colour histograms. However, the detection quality of some people varies from frame to frame, e.g. some lighter parts of their clothing might not be detected in several frames. This causes their colour histograms to differ, which:

1. Increases the Bhattacharyya distance between a person and the same person in the previous frame, and
2. Raises the likelihood of confusing the person with another.

Also, some people might simply wear very similar clothes, which additionally increases the difficulty of distinguishing them. Therefore, I have avoided relying purely on the colour histograms  $H_{new}$ , I have also used the error of position prediction  $Err_{new}$  and angle between expected position and new point's position from person's third from last position  $Ang_{new}$  to estimate the probability of a blob representing a particular person  $P_i$ . Using Bayes' theorem:

$$p(P_i | H_{new}, Err_{new}, Ang_{new}) = \frac{p(H_{new}, Err_{new}, Ang_{new} | P_i) p(P_i)}{p(H_{new}, Err_{new}, Ang_{new})}$$

I assume that

$$\forall_{i,j} p(P_i) = p(P_j)$$

$$\forall_{i,j} p(H_i, Err_i, Ang_i) = p(H_j, Err_j, Ang_j)$$

$$p(P_i | H_{new}, Err_{new}, Ang_{new}) = p(H_{new} | P_i) p(Err_{new} | P_i) p(Ang_{new} | P_i)$$

**4.2.6.4.1 Prediction Error Probability.** I define the error in position prediction as the distance between the predicted and the actual positions. I computed a histogram of error values for the position predictions of 25 people, which totalled 4059 samples. I chose the bins to have a width of one pixel [ 4 ]. The numbers in each bin were divided by the number of samples; using this maximum likelihood approach I obtained a likelihood measure for each bin. I then tried to fit different continuous distributions to this discrete distribution. The results are shown in Figures 4.6a–4.6b. Since the error values are positive, the Gaussian distribution was fitted to the error distribution plus its reflection about zero. Then the values of the Gaussian were multiplied by two and shown only for positive error values. The multiplication was necessary to ensure that the distribution would still integrate to one after removing half of it. I computed the negative log likelihood for each fitted distribution. The exponential distribution proved to be the best fit.

$$p(\text{Err}_{new} | P_i) = \lambda e^{-\lambda \times \text{Err}_{new}}$$

$$\lambda = 0.18$$

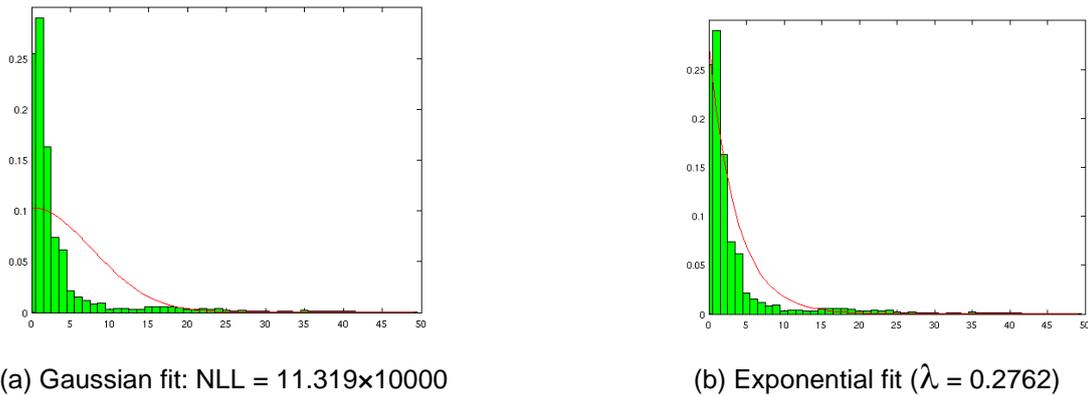


Figure 4.7: The distribution of prediction errors [figure copied from 4].

I also checked wheter this distribution worked in other different condition. And found that value of lambda 0.18 works very good in most of the conditions, so I changed the value of lambda to 0.18.

**4.2.6.4.2 Histograms Probability.** I approximate the probability distribution of the colour histograms using Bhattacharyya distance. Bhattacharyya distance it varies between 0 to 1. But it varies very much linearly which was not giving satisfying results, so varied it exponentially as given below.

$$p(H_{new} | P_i) = e^{-5 \times H_{new}}$$

Where -  $H_{new}$  is Bhattacharyya distance.

**4.2.6.4.3 Angles Probability.** I approximate the angle probability using a normal distribution. I calculated the angle between expected position and current blob position to choose from third position from last of trajectory. I gathered the 96 normal trajectories which gave 5457 samples of angles in radians and calculated the mean and variance. Values of mean and variance came 0.008 and 0.1023 respectively, so I chosen the mean 0. But results were not satisfactory with this variance value, and then I tried the trial and error method to find the correct value of the variance. I found 0.70 was very much satisfactory. I tried some big values more than 1 also but didn't work out, because with high values value of angle probability was negligible in front of other probabilities.

In our case mean is zero. So:  $p(\text{Ang}_{new} | P_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\text{Ang}^2}{2\sigma^2}}$

Where:  $\text{Ang}$  is angle.  
 $\sigma$  is variance(0.7).

Angle probability is very useful when two people walk parallel, so there is chance of confusion in error in position prediction probability and histogram probability because it does not vary much so we need another factor in these types of cases.

### 4.3 Removing bad trajectories

The detector and tracking components are not 100% accurate, so before writing the trajectory to file, we need to remove all those bad trajectories. Some of these may be recognisably incorrect, such as those falling into the following classes:

1. Tracker can produce a bad trajectory as the result of a stationary object. Those have more than 100 points, but all are concentrated in one small region. Normally successive points has very low distance, so I made another trajectory from original trajectory whose successive points have distance more than minimum distance ( 7 ). I repeated this process one more time with minimum distance (14). So we have two new trajectories different lengths (len2 and len3). I checked the change in length of the two trajectories and calculated the change in percentage as (per2). Also I calculated the change in percentage from original trajectory length (len1) and len2 as (per2). To remove these bad trajectory if  $\text{per1} > 55$  than minimum percentage is chosen 40 else minimum percentage is chosen 75. Per2 should be more than minimum percentage. These parameters are chosen by number of trails and experiments. It is described in Pseudo code 7.
2. Trajectories shorter than 15 points or  $\text{len1} < 15$ . These could represent spurious detections.
3. Trajectories which start or end outside the marginal area of the scene. The marginal area is shown in Figure 4.7. These trajectories could be produced if the detector did not detect a person in the initial or final frames. Also, the tracker could fail to notice that two people split and classify one of the resulting blobs as a new person in the scene. If they split outside the marginal area, the new person would appear out of nowhere. So these trajectories should be removed.

---

Pseudo code 7 Remove Bad Trajectories

---

```
len1= length(trajjectory);
trajjectory1=removeClosePoints(trajjectory,7);
len2=length(trajjectory1);
trajjectory2= removeClosePoints(trajjectory1,14);
len3=length(trajjectory2);
per1=100*(len1-len2)/len1;
per2=100*(len2-len3)/len1;
IF per1>55
    MinPer=40;
ELSE
    Minper=75;
END
IF(len1<15 || per2< MinPer || len3<10)
    %%% bad trajjectory
    return false
ELSE
    Return true
END
```

---

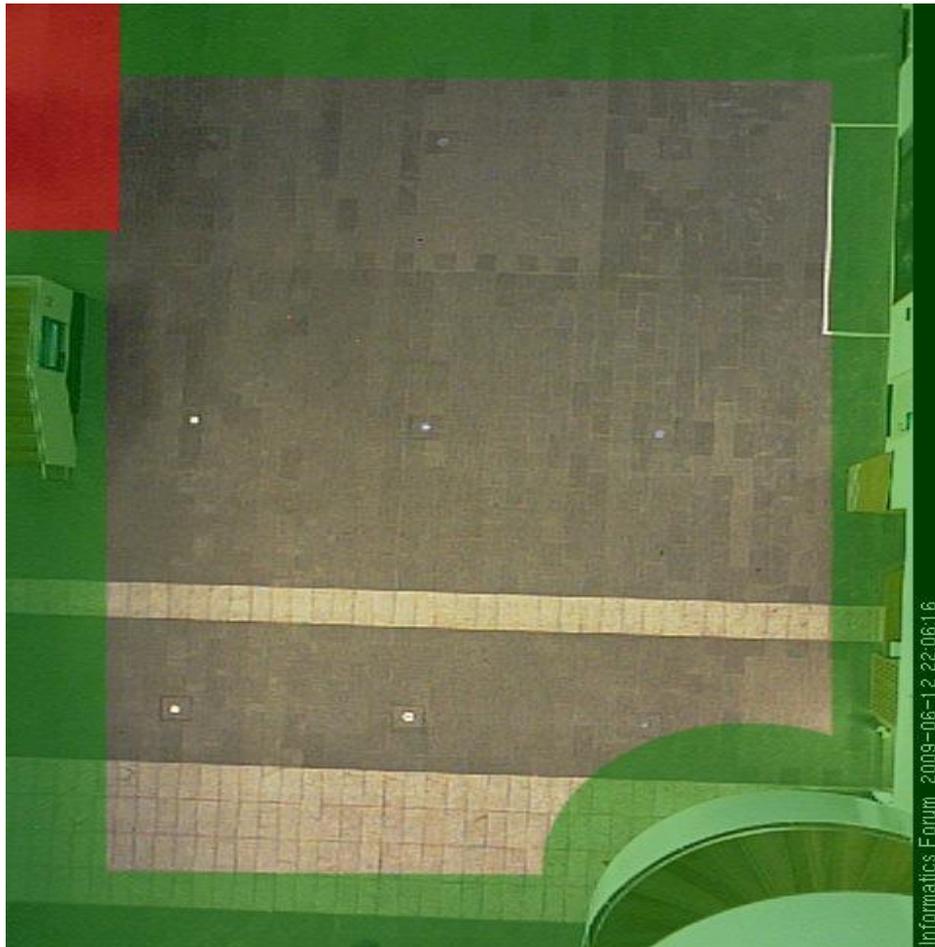


Figure 4.8: The marginal area (green) shows the region where trajectories have to start and end. The red area shows the region next to the lifts where the trajectories were removed [ figure copied from [2]].

## 4.4 Implementation.

The tracker was implemented in Matlab. Main variables like current data storing the trajectory of person being tracked and final variable storing all the trajectories are cell array in Matlab. The main structure of the program is illustrated by functions in diagram shown in Figure 4.8.

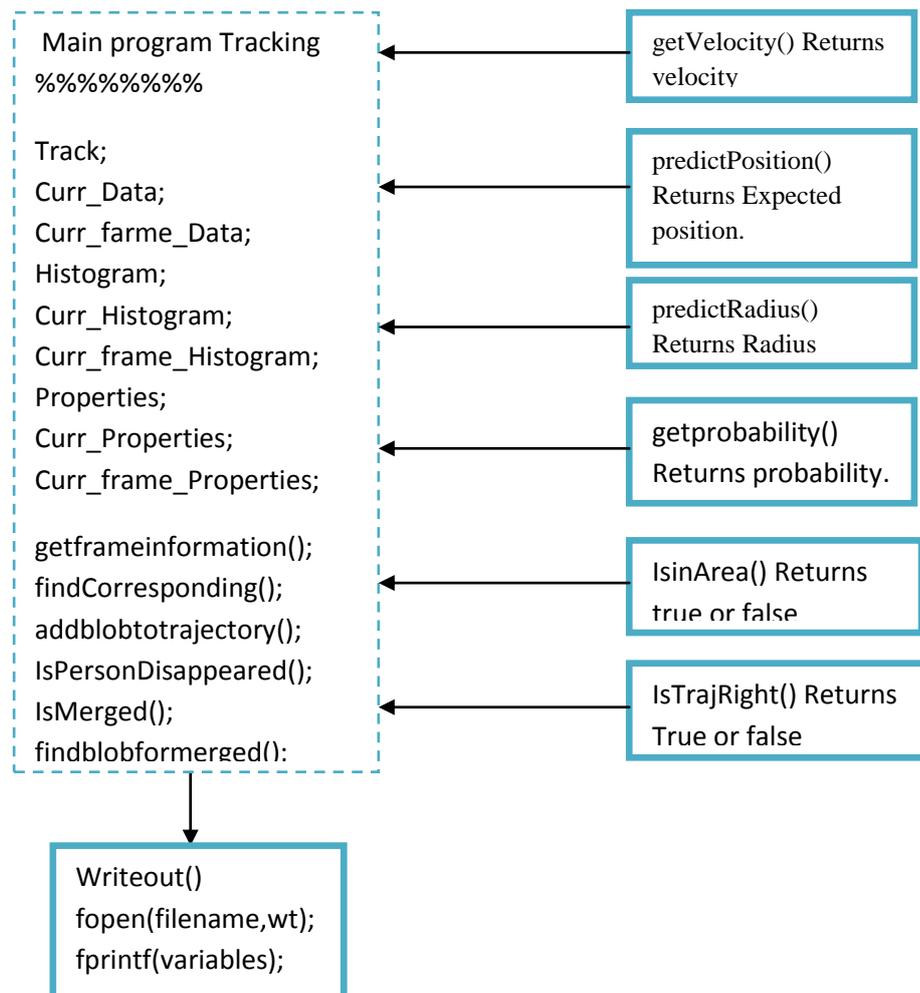


Figure 4.9 Block Diagram of Tracker program

## 4.5 Output File Description.

These files contain sets of detections that have been tracked together into a single target's trajectory. Tracker files start with "% Total number of trajectories in file are [Number]", where Number defines the number of trajectories. Files contain the information in the form of a Matlab structure. The trajectory points and the properties are in two different variables with same identifier. Each trajectory has a different identifier like "R1" for trajectory number 1 and "R2" for trajectory number 2 and so on. The first variable is *Properties.{Identifier} = [ Number\_of\_Points\_in\_trajectory, Start\_time, End\_Time, Average\_Size\_of\_Target, Average\_Width, Average\_height, Average\_Histogram ]*; The histogram has the same format as in the detection file. The second variable contains the full trajectory as *TRACK.{Identifier} = [[ centre\_X(1) CentreY(1) Time(1)] ; [ centre\_X(2) CentreY(2) Time(2)] ..... and so on ..... until ..... [ centre\_X(end) CentreY(end) Time(end) ]]*; The size of tracked files is about 1MB each. These files can be downloaded from [7].

## 4.6 Evaluation.

The evaluation of the tracker was carried out on 5 categories with 60 trajectories from day. Results were very satisfactory. Results are shown given below category wise.

1. When there is only person in frame:  
 Total number of persons in scene: 41  
 Number trajectories preserved: 41

In case of single person in scene tracker performed very well. An example of single trajectory is shown in figure 4.9.

2. When there were people walking closer together.  
 Number of persons: 3  
 Trajectories preserved: 4

This is the scenario of merging and splitting, so one trajectory is produced redundantly, because sometime these persons were represented by separate blobs. An example of this is shown in figure 4.10. The tracker performed very well in when there were only two people in the scene.

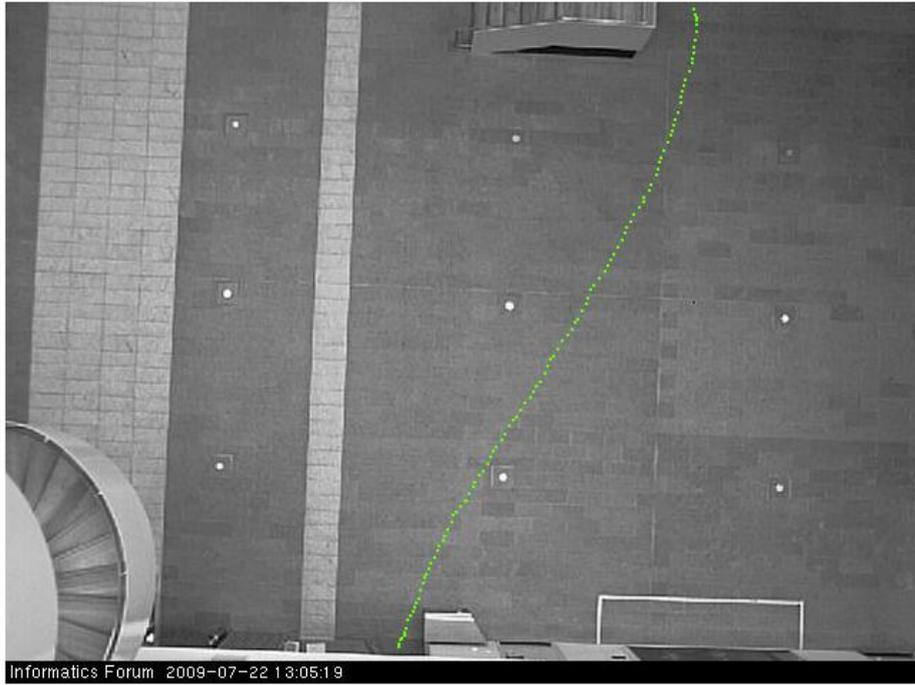


Figure 4.10: Single person trajectory

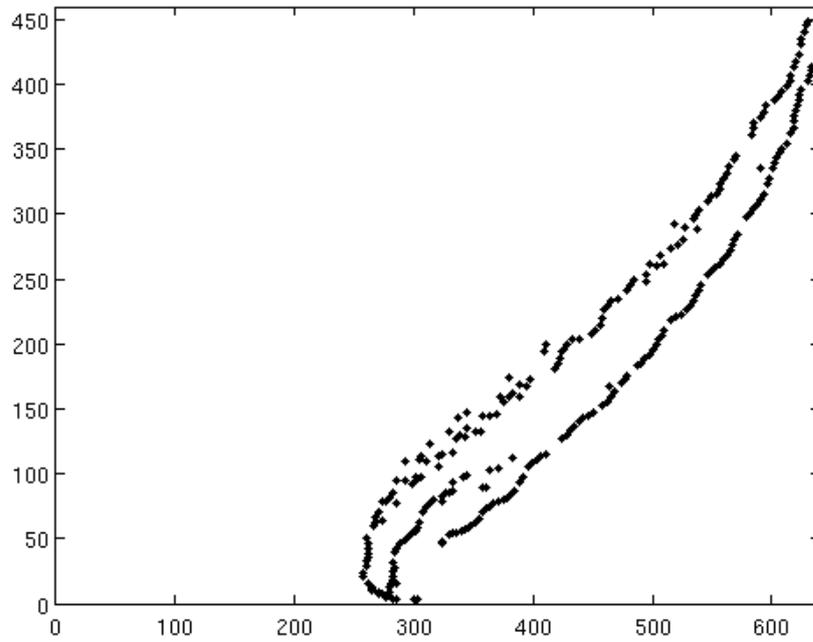


Figure 4.11: Example of merging and splitting condition.

3. When a person disappears for a few frames.  
 Number of people disappear more than 2 frames: 11  
 Number of trajectories preserved: 9.

One person disappeared more than 5 frames, so the trajectory terminated.  
 An example figure of disappearing is shown in figure 4.11.

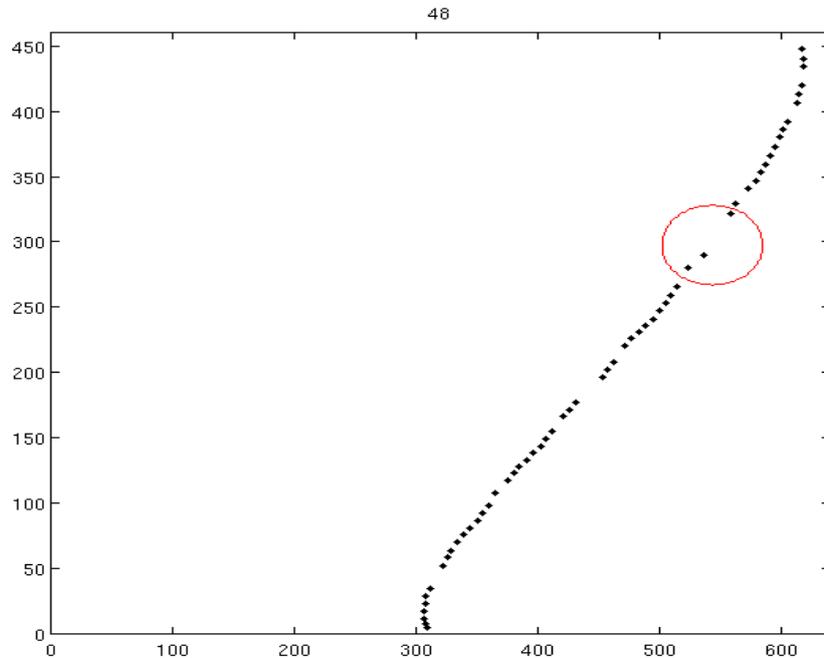


Figure 4.11: Example of disappearing. Red circle shows where a person disappeared for 3 or more frames.

#### 4. Failure rate

Tracker performed very well, but still it fails in some situations like there is merging and splitting. Sometime it fails to identify correct blob after merging and splitting.

#people	#merging and splitting	# correct identity reassignment	% failure rate
2	2	2	0%
3	4	3	25%
total	6	5	16.6%

Table 4.1. The table shows the failure rate of correct identity reassignment.

Overall performance of tracker was very satisfactory. All trajectories of Jan 04 2010 are shown in figure 4.13.

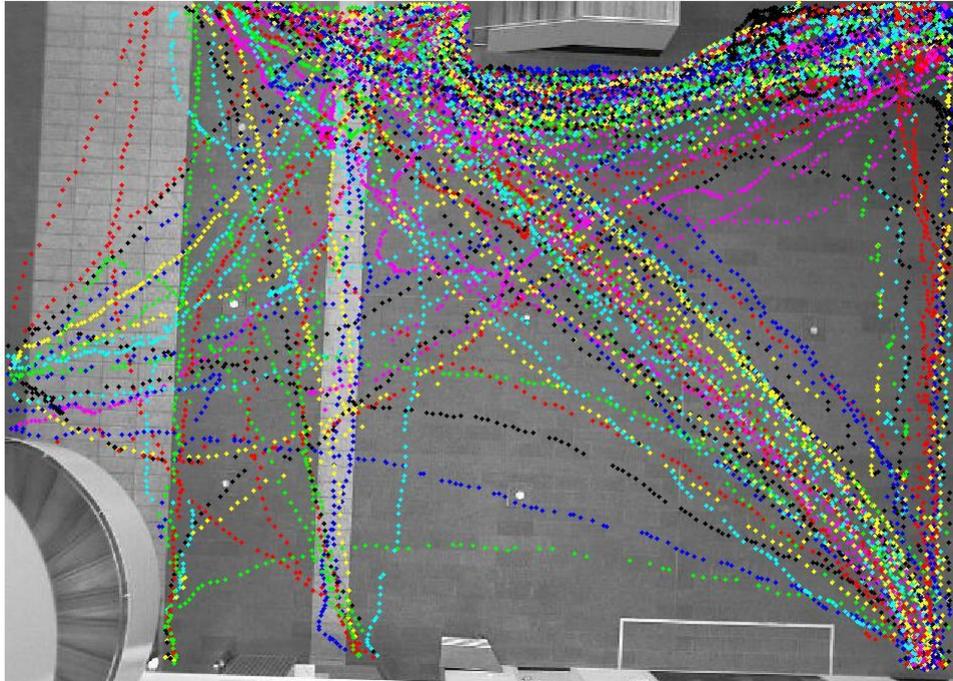


Figure 4.13 Tracked object for 4<sup>th</sup> Jan 2010. These are same day trajectories corresponding to detected objects shown in figure 3.1.

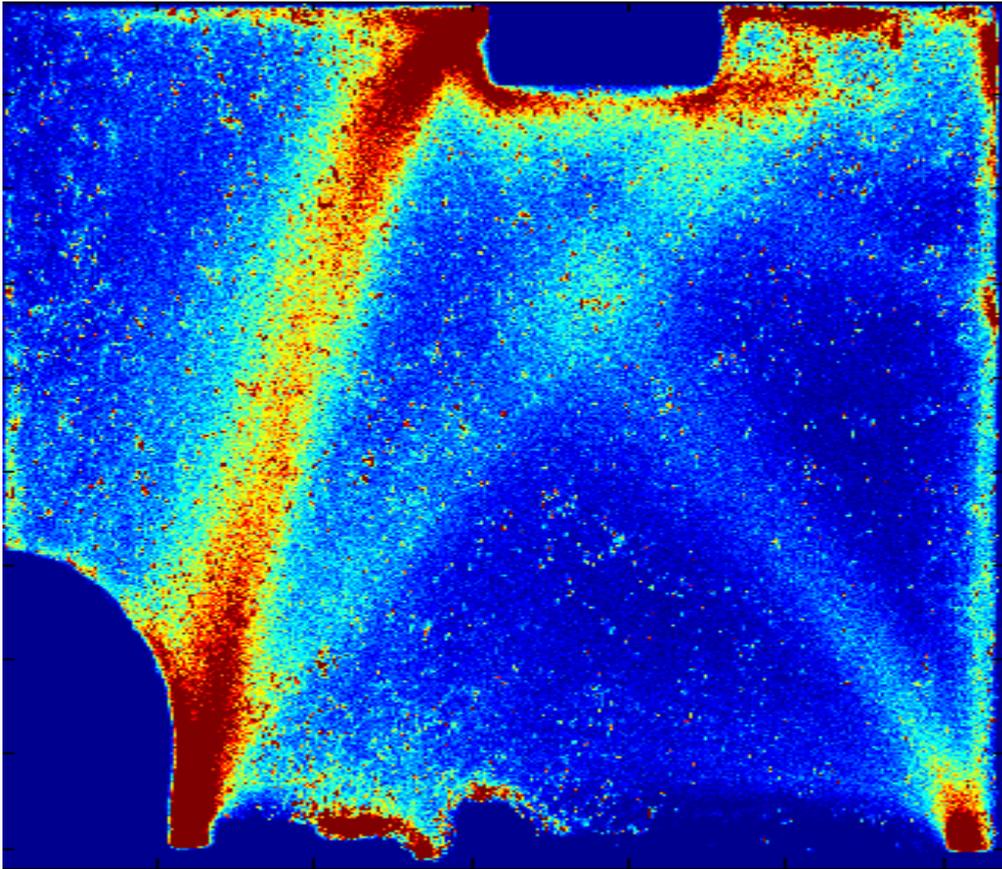


Figure 4.14: This figure is image plot of all the detection points after tracking using all trajectories (65529). The colour varies from dark blue to dark red through light blue, green and yellow with the variation of number of detected point at that point. It tells us where people go most of the time. Colour represents the density of visiting at a particular point. As we can see from the figure most of the people go from the main entry to the stairs and vice versa. Also density near the elevator is high, because people waiting for the elevator. Density is high near all main entry and exit points like vision lab door, robot lab door, night exit door, and near reception.

# Chapter 5

## Representing Trajectories as Spline

### 5.1 Specification

Trajectories have different numbers of points which makes it difficult to compare them. In order to represent them using the same number of attributes, each trajectory was approximated by a cubic spline curve with 6 control points. The fitting algorithm and its implementation were provided by Rowland R. Sillito [6].

### 5.2 Design

First Spline read all the trajectories from tracker file. The trajectories produced by the tracker have point X and Y positions. So all these trajectories are transformed in 0 to 1 range both X and Y. After pre-processing according to number of control points a spline fit to each trajectory. The average error of the spline fit to the tracked trajectories is calculated assigned as deviation to each trajectory. With abnormal behaviour deviation also increases.

#### 5.2.1 Choose Control Points

I chosen the number of control points based on the average error of the spline fit to tracked trajectories I used 2343 trajectories, and took the median of deviation of all the trajectories for different number of control points and plotted it as shown in figure 5.1.

Number of control points was chosen 6, after 6 there was very small change in deviation. And also when number of control points is 6 deviations is about 1% of image width. Trajectories used (2343) to calculate meadian also include the bad trajectories.

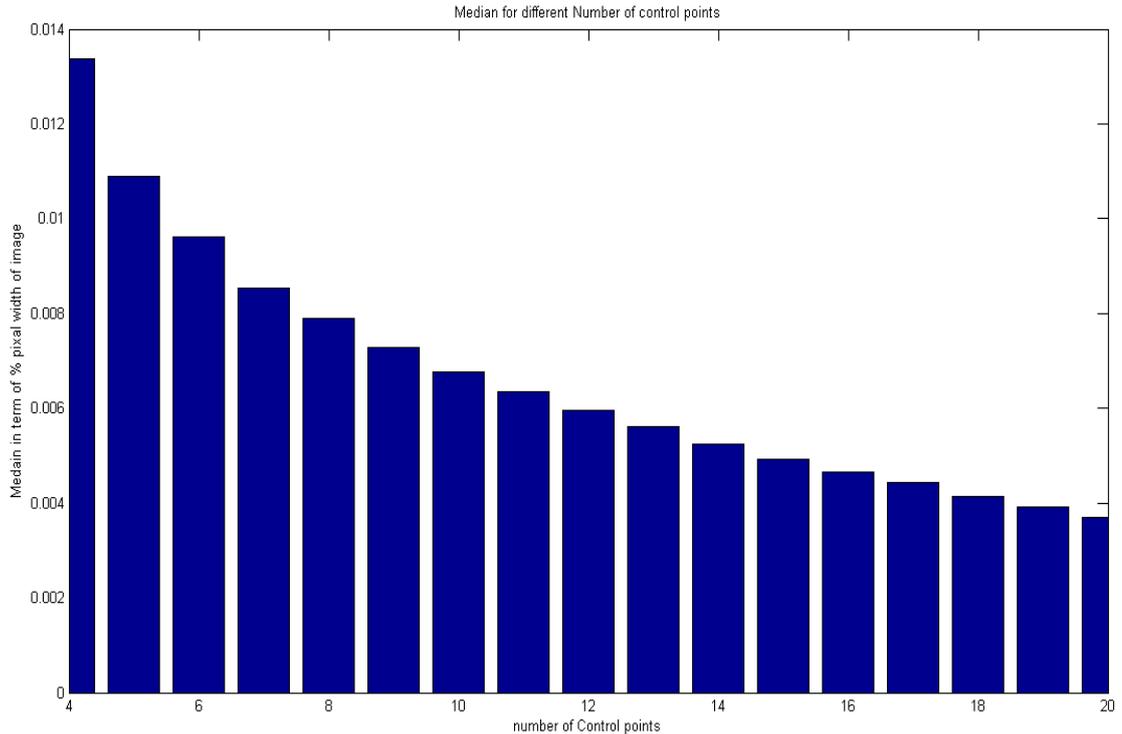


Figure 5.1 Variation of Median fitting error with number of control points.

The high level algorithm is illustrated by Pseudo code 7.

---

#### Pseudo code 7 Fitting Spline

---

```
Trajectories=readTrackerFile();
```

```
FOR EACH Trajectory
```

```
    [x y]= transformTrajectory(trajecotry);
```

```
    ContolPoints= splinefit(x,y,numberOfControlPoints);
```

```
    [Xr Yr]= constructSpline(x,y, controlPoints,LengthOfTrajectory);
```

```
    Deviation =getdeviation(x,y,Xr,Yr);
```

```
    writeOutput File(Deviation,ControlPoints);
```

```
END
```

---

## 5.2 Implementation

Spline is implemented in Matlab. Splinefit was developed by Rowland R. Sillito. Other experiments related to spline were also done on Matlab like choose number of points.

## 5.3 Output File Description

These files contain sets of 6 point spline descriptions of the tracked trajectories. The spline file contains the average error of the spline fit to the tracked trajectories, and the control points. This is for each trajectory produced by tracker with same identifier as tracker. The first line of spline file is "% Total number of trajectories in file are [Number]", where Number defines the number of trajectories. "X and Y are normalized by dividing 640 and 460 respectively" and "Image size is 640\*460". Normalization is done because the spline fit works for variables in the range [0,1], so we transformed the values of the trajectory points to fall into [0,1]. The file contains the information in the form of a Matlab structure. Identifiers of each spline are the same as given in the tracker file for the corresponding trajectory. Deviation and Control points are stored as Deviation.{Identifier}= [ Standard deviation ];. This is the average distance between the tracked point and the closest point on the spline. The control points are stored as: Controlpoints.{Identifier}= [[Controlpoint\_x1 Controlpoint\_y1]; [Controlpoint\_x2 Controlpoint\_y2]..... and so on until six points ]];

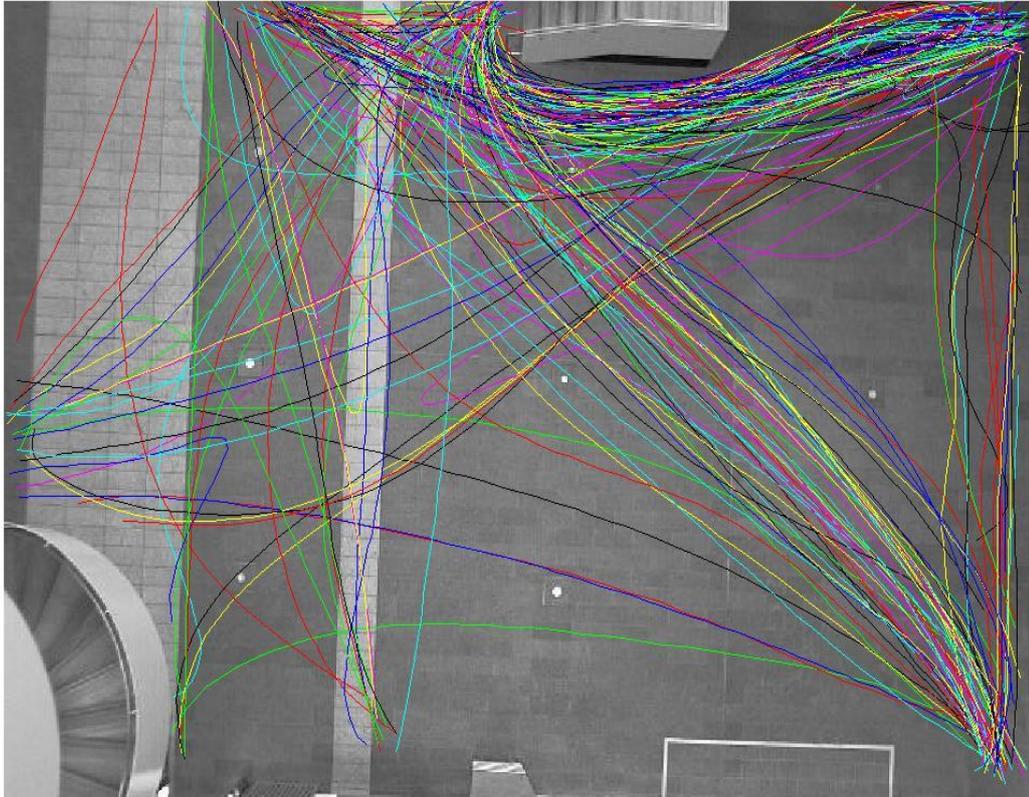


Figure 5.2: This figure shows spline curves of all the trajectories shown in figure 4.13 on 4<sup>th</sup> January 2010.

# Chapter 6

## Abnormality Detection

### 6.1 Overview

The purpose of this model is to detect the abnormal behaviour from the given trajectory based on the multiple parameters. Spline is used to define all trajectories with same number of attributes (Control points). There are many step involved in the detection of abnormal trajectories. Trajectories provided by tracker component are fed to this component. There are three parts involved in abnormal behaviour detection

1. **Training dataset** was used for training the model (section 6.2)
2. **Training dataset for choosing threshold of parameters** was used to choose the appropriate parameter for process of detection of anomalous trajectories (sub section 6.3).
3. **Test dataset** was used to evaluate the robustness of component (section 6.5).

According to figure 1.2 there are 13 entry and exit points, which means there are 13 possible paths from one entry point to other exit points (including entry point as exit point as a path), so there are total 169 possible paths (which make 169 clusters). I gathered the samples for each cluster to calculate the mean vector and covariance matrix for each cluster.

Choosing the classifier parameters was a difficult task, which required a lot of calculation and manual understanding. We were looking for multiple parameters for decision making, so it was difficult to choose parameters by observing manually. I used several possible combinations of parameter and set the threshold for each, which is described further in subsection 6.2.1. A trajectory is flagged abnormal if it met the criterion.

## 6.2 Building the Model

I gathered the 60 samples for each possible path, 20 normal behaviour tracks were selected for each path for building the model of normal trajectories. Paths where entry and exit points were same and paths where sufficient number of tracks for modelling (less than 20 normal tracks) was not available were not modelled, because 6 control points are represented by a 12 point vector, so a minimum 13 samples needed to calculate covariance matrix. There were 107 clusters modelled, because only 107 out of 156 (excluding those clusters where start and end points were same) had 20 more tracks. If a trajectory starts and ends on same point or if it belongs to un-modelled path it is flagged as abnormal. From 20 training trajectories of a cluster, the mean vector of control points and covariance matrix were calculated. As shown in figure 6.1 all trajectories are very much similar except few are little different (e.g. Right most trajectory and one it's near) which allow little variation in normal behaviour.

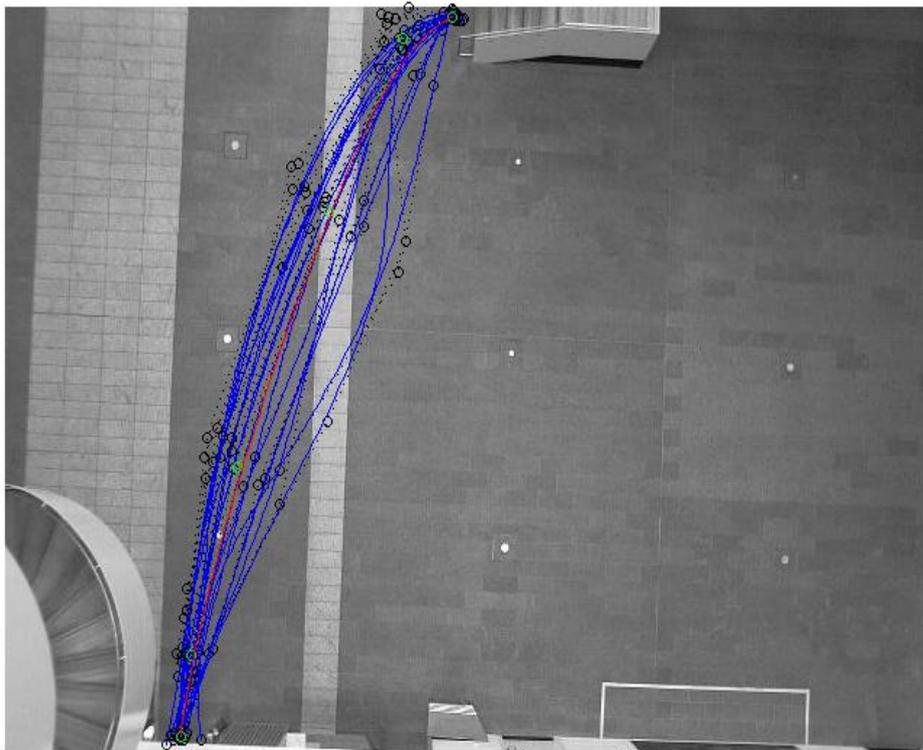


Figure 6.1 An example training cluster. Where training spline are in blue colour with their control points in black and red colour spline represent mean vector with green mean control points

## 6.2.1 Calculations

There are three main calculations to compute the parameters.

### 6.2.1.1 Spline Fit Error :

Spline fit error is the deviation of spline curve to original trajectory curve. It was computed as given in equation 6.1.

$$SFE = \sqrt{\sum_{i=1}^{length} (Traj_i - Spline_i)^2}$$

Where: Traj is original trajectory point  
 Spline is spline fit curve closest point  
 length is number of points in trajectory

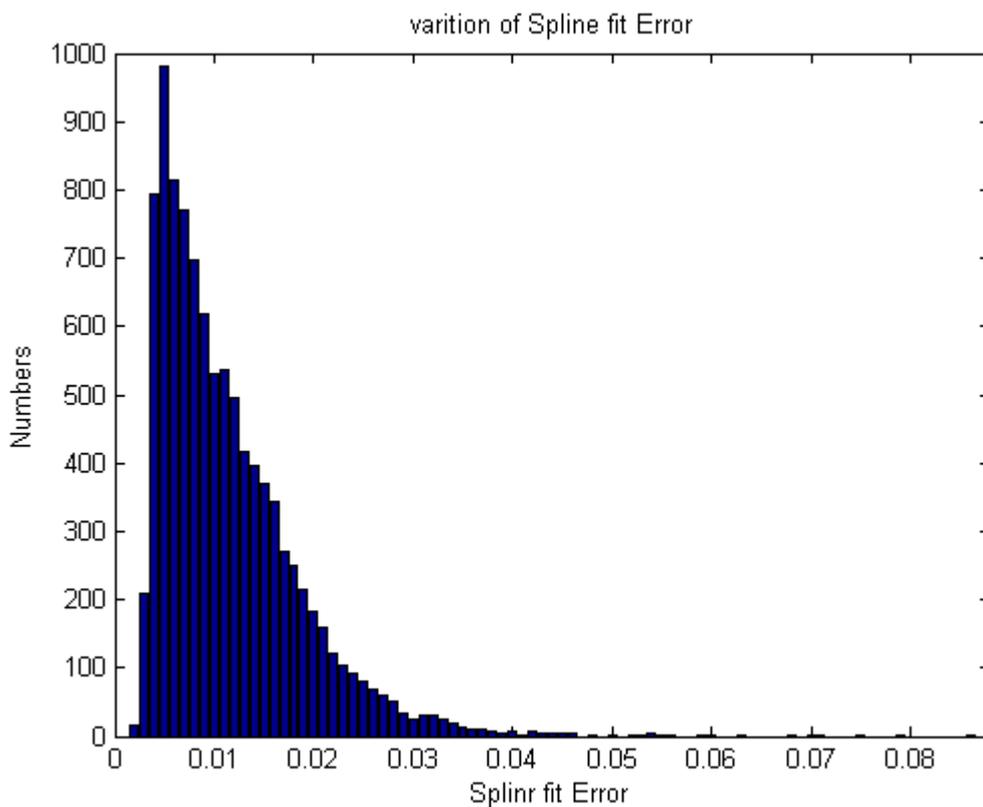


Figure 6.2: figure shows the variation of number with the variation of spline fit error most of the trajectories having a spline fit error of less than 0.01.

### 6.2.1.2 Vector Distance:

With the samples of 107 clusters I computed 107 mean vector and 107 covariance matrixes. Each mean vector contain 6 control points and length of vector is 12 as 6 X and 6 Y in each mean vector. Vector distance was computed between present trajectory's control points vector and closest mean vector. Matlab's norm function was used to compute vector distance.

$$VD = \text{norm}(X - \text{Mean})$$

Where: X is present vector  
Mean is mean vector

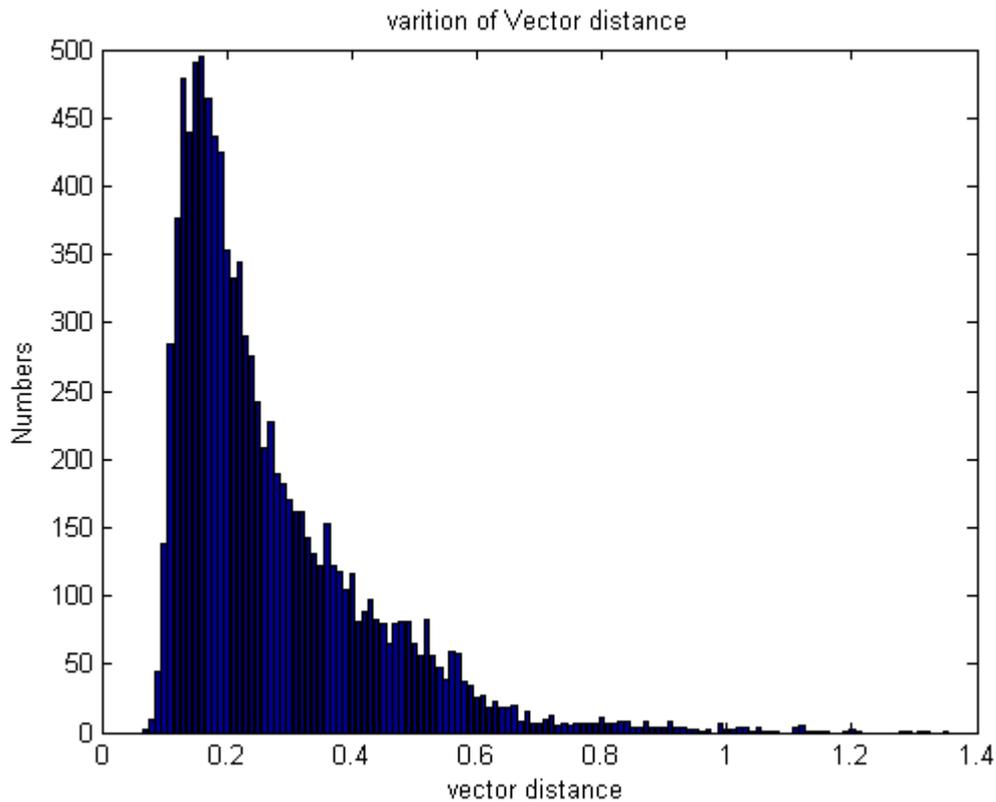


Figure 6.3 shows the variation of numbers with variation of vector distance to closest mean vector. Most of the trajectories have vector distance about 0.2.

### 6.2.1.3 Probability

Each trajectory is represented by control points, which gives us a 12 point vector. This vector was used to calculate multivariate Gaussian probability. 12 dimensional Gaussian probability is computed for every trajectory as given below.

$$P(x|\mu,\Sigma) = \frac{1}{(2\pi)^{k/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)}$$

Where  $\mu$  is mean vector  
 $\Sigma$  is covariance matrix

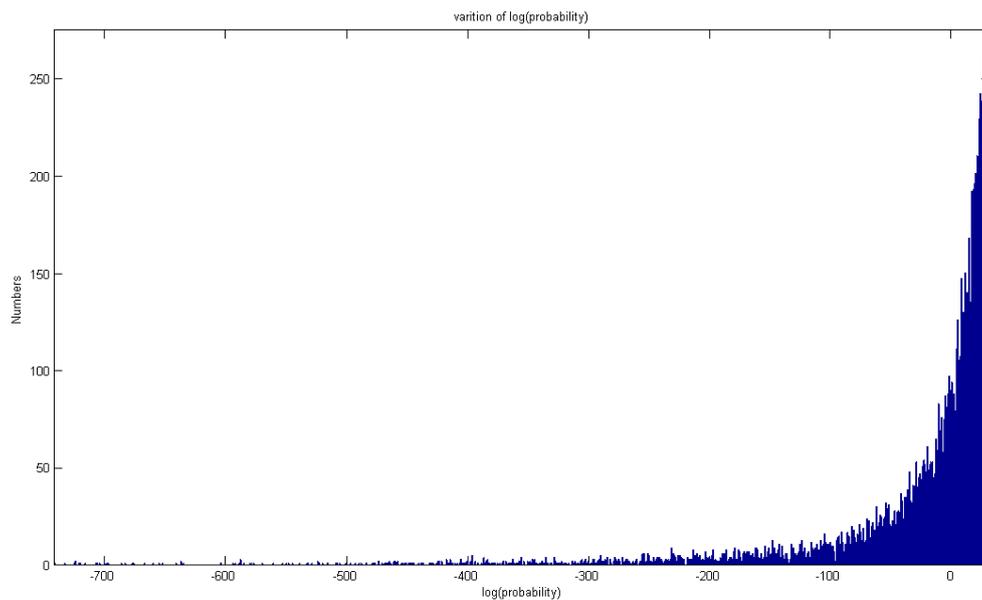


Figure 6.4: figure shows the variation of numbers on each log(probability). I took the log of probability because the span of probability was very wide, which was not suitable for plotting. There are some trajectories whose log of probability minus infinity, because when log of probability is less than -744, then Matlab round it to minus infinity.

### 6.3. Choosing Parameters Threshold

I gathered another training dataset of 535 trajectories (5 from each cluster those were not used in building the model) for choosing the appropriate threshold of parameters. First I looked on all trajectories and labelled the bad trajectories. I found there were total 80 bad trajectories. Three criteria for detection of bad trajectories are given below.

1. If spline fit error greater than threshold of spline fit error (Thres1\_Sfe) .
2. If vector distance greater than threshold of vector distance (Thres2\_Vd).
3. It is combination of all three parameters a). Log (probability) (Thres3\_P)  
b) Vector distance (Thres4\_Vd) c) spline fit error (Thres5\_Sfe).

Threshold values are given in table 6.2.

In logical form 3<sup>rd</sup> condition is described as given below.

```
IF (Thres3_P && Thres4_Vd && Thres5_sfe)
    % It is bad trajectory;
    Plot (bad)
END
```

Combination of all 3 is expressed as given below:

```
IF (Thres1_Sfe || Thres2_Vd || (Thres3_P && Thres4_Vd && Thres5_sfe))
    % It is bad trajectory;
    Plot (bad)
END
```

To find the appropriate threshold for each I used the concept of false positive and false negative and tried all the possible combinations for all five threshold hold.

False positive: bad trajectory accepted as good.

False negative: good trajectory rejected as bad.

$$FP = \frac{A \cap B}{N} \quad FN = \frac{R \cap G}{N}$$

Where: FP - False positive.

FN - False negative.

- A - Accepted trajectories.
- R - Rejected trajectories.
- B - Bad (80 out of 535) (pre-labelled).
- G - Good (465 out of 535) (pre-labelled).
- N – G + B (535).

To figure out the equal false positive and false negative rate for all three variables (spline fit error, vector distance and log (probability)) I plotted three graph (FN(%) Vs FP(%)) for each variable. Plots are shown in figures 6.5, 6.6 and 6.7. These plots gave a good rough idea for choosing the thresholds.

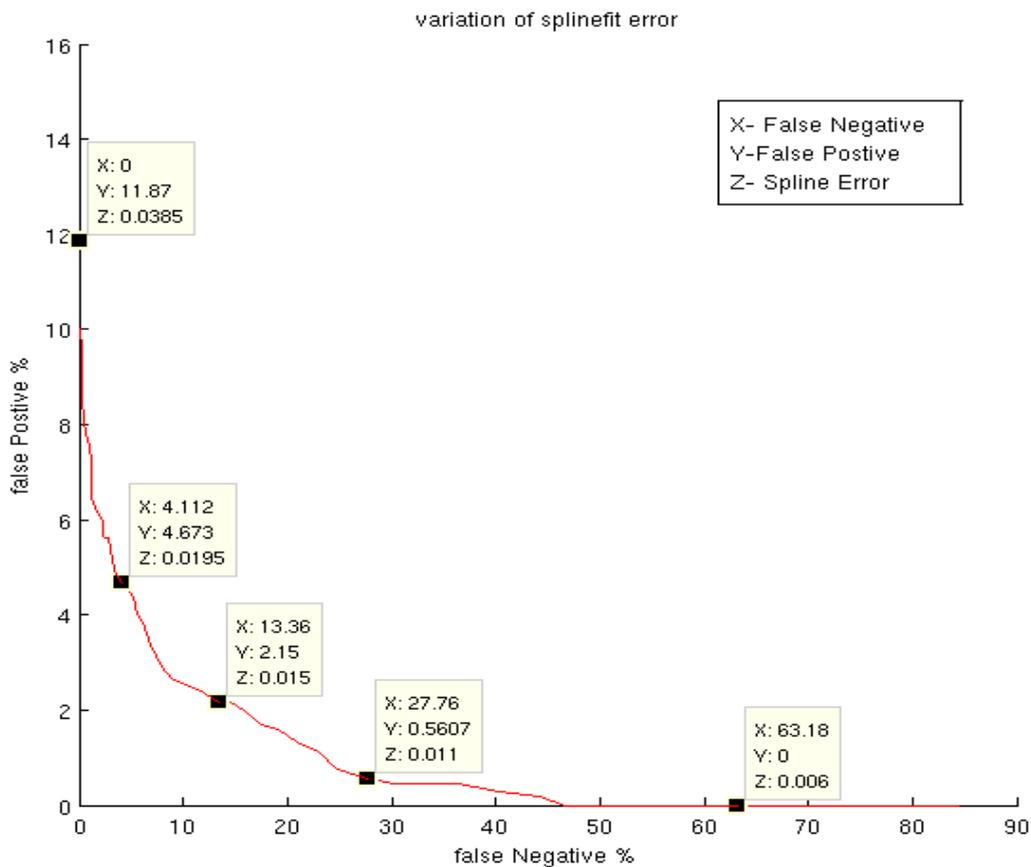


Figure 6.5: This figure shows the variation of false positive (Y) and false negative (X) at different values of spline fit error threshold (Z). This plot show the effect of only spline fit error, only spline fit error was used to detect the bad trajectories (only 1<sup>st</sup> condition).

It has equal error rate (about 4.3%) at spline fit error threshold is near 0.0195.

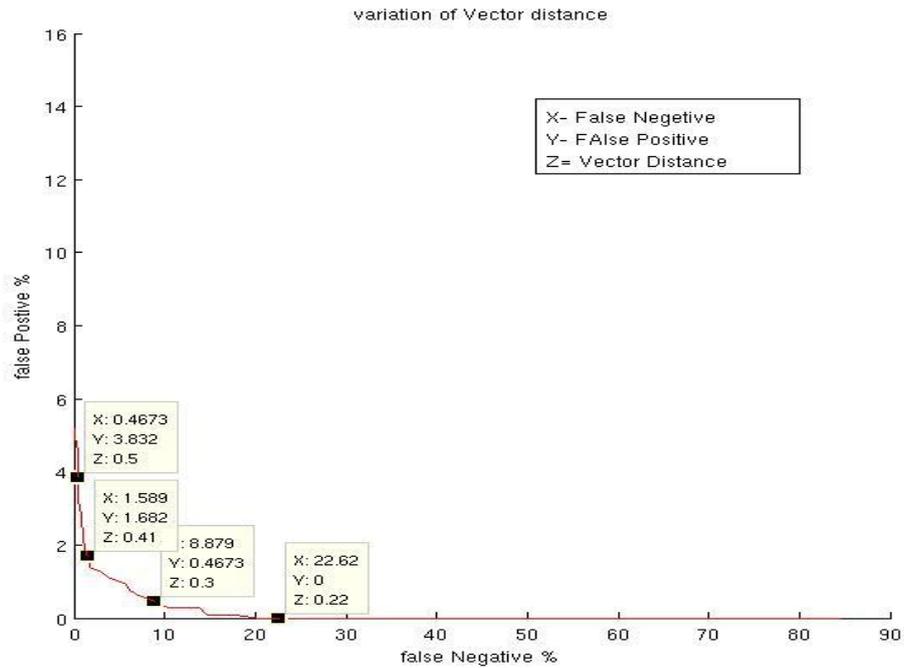


Figure 6.6: This figure shows the variation of false positive (X) and false negative (Y) at different values of vector distance threshold (Z). This plot show the effect of only spline vector distance.

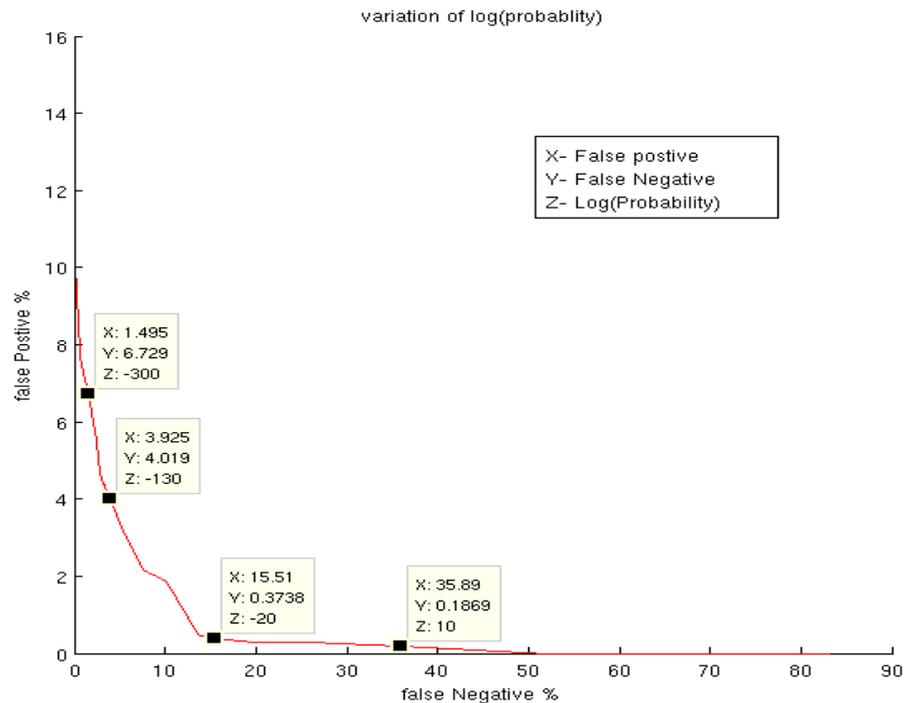


Figure 6.7: This figure shows the variation of false positive (Y) and false negative (X) at different values of log(probability) threshold (Z). This plot show the effect of only log(probability).

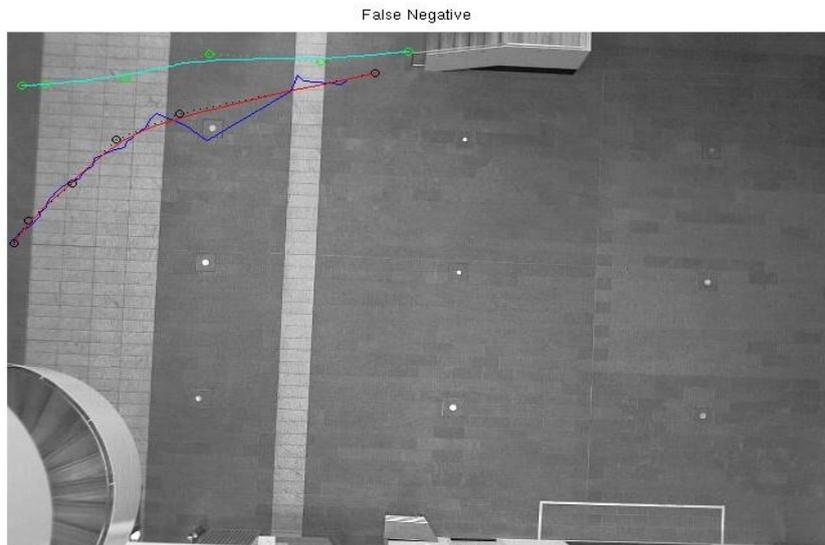
Parameter	Spline fit error	Vector distance	Log(probability)
Equal error rate	4.3 %	1.6%	4.0%
Threshold	0.0195	0.41	-130

Table 6.1: shows equal error rate for each parameter with threshold at that EER.

From table 6.1, we can say that vector distance has lowest error rate, so it most effective parameter for detecting the bad trajectory. I tried all the possible combination of five thresholds to find minimum equal error rate on 535 training trajectories. I found minimum equal error rate equal to be 1.2245% at following values of thresholds. These threshold values are used to recognise anomalous behaviour. Equal error rate for false negative and false positive is about 1.2%. Some examples of false negative and false positive trajectories are shown below. Here mean vector (control points) are in green colour, its spline in cyan, original trajectory in blue, control points in black and spline in red.

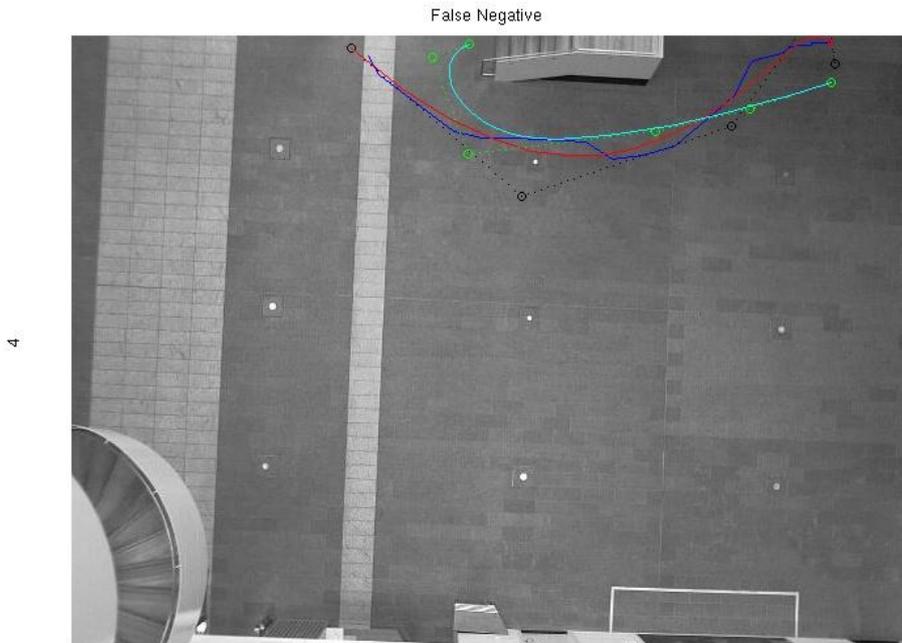
Equal error rate	1.2245
Thres1_Sfe	0.038
Thres2_Vd	0.44
Thres3_P	-20
Thres4_Vd	0.006
Thres5_sfe	0.37

Table 6.2: shows thresholds values at best equal error rate for training.



3

Figure 6.8 (a) An example of false negative trajectory. The vector distance is large because the mean vector is very far from the trajectory control points, but this is because the area 1 is very big so it is difficult to find a mean vector to suit every trajectory.



3

Figure 6.8 b) An example of false negative. Trajectory is normal, but it starts and ends at very corners of start area (3) and end area (4), however mean vector is from middle of start and end areas.



3

Figure 6.9 a) An example of false positive trajectory. Path distance is very small, so the positioning of a bad trajectory's control points does not differ very much so it passes away. Also, small path has small trajectory (fewer number of points), so spline fit error cannot be very high



13

Figure 6.9 b) An example of false positive trajectory. It is same situation as above (small path).

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	332	248	83	28	20	1	4	20	4	61	212	213	245
2	222	252	363	492	291	325	19	783	15	438	1344	630	677
3	152	613	313	1766	1904	247	41	1500	35	609	7348	2321	496
4	16	286	870	622	641	25	8	96	7	209	1353	203	76
5	20	244	1327	535	660	72	95	616	44	328	667	81	78
6	4	232	310	19	88	44	92	508	18	11	98	14	12
7	5	28	35	29	126	81	147	195	48	15	15	8	9
8	22	817	1431	150	807	464	136	124	67	354	632	127	94
9	3	10	18	21	18	4	19	63	111	29	10	9	3
10	50	376	335	271	283	14	8	307	26	247	922	140	135
11	121	1182	6580	3100	549	119	9	605	22	893	287	236	339
12	116	507	2167	526	115	24	8	167	11	154	227	86	139
13	257	727	469	146	66	8	4	104	10	175	270	172	138

Table 6.3: table shows number of trajectories in all 169 clusters present from total 65529 trajectories.

Main structure of abnormal behaviour detection program is described by Pseudo code 8.

---

Pseudo code 8 abnormal behaviour detection

---

```

Load trajectories
Load Mean_Vectors           % in cell array form
Load Covariance_Matrixes    % in cell array form

FOR EACH Trajectory

    Start_Area=AreaNumber(trjectoryi(start point));
    End_Area=AreaNumber(trjectoryi(End point));
    Mu= Mean_Vectors{ Start_Area , End_Area } ;
    Sigma= Covariance_Matrixes { Start_Area , End_Area } ;

    IF( path is modelled?)

        [x y]= transformTrajectory(trajecory);

        ContolPoints= splinefit(x,y,numberOfControlPoints);

        [Xr Yr]= constructSpline(x,y, controlPoints,LengthOfTrajectory);

        Deviation =getdeviation(x,y,Xr,Yr);
        Vector_dist =norm(ContolPoints-Mu);
        Probability = log(getprobability(ControlPoints,Mu,Sigma));

        IF (Thres1_Sfe || Thres2_Vd || (Thres3_P && Thres4_Vd && Thres5_sfe))
            % It is bad trajectory;
            Plot (bad);
            Title('BAD')
        END
    ELSE
        % It is a un-modelled path, so it is a bad trajectory;
        Plot (bad);
        Title('um-modelled');
    END

END

END

```

---

### 6.3 Implementation

Abnormality detection is implemented in Matlab. Covariance matrixes and mean vectors for each cluster were calculated from the training dataset in Matlab and saved as mat file format. And covariance matrices and mean vectors load to main program to calculate 12 dimensional Gaussian probability (which is inbuilt function in Matlab). Also other experiments related to Abnormality detection were done on Matlab.

### 6.4 Results

To evaluate the performance of the abnormality detection process, another 535 trajectories were gathered, 5 from each cluster to ensure that chosen parameters satisfy all the paths (107). False negative and false positive rate were computed for the new dataset. Results were found very satisfactory, with the error rate for false positive and false negative was equal at 1.2069 %, while for training dataset equal error rate (EER) was 1.2245 %.

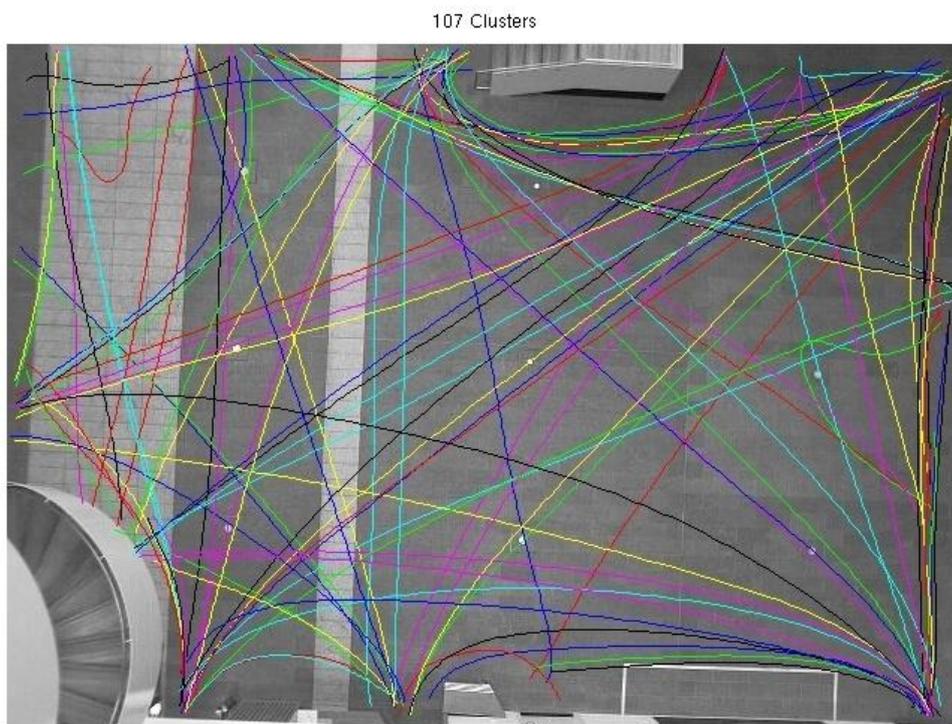
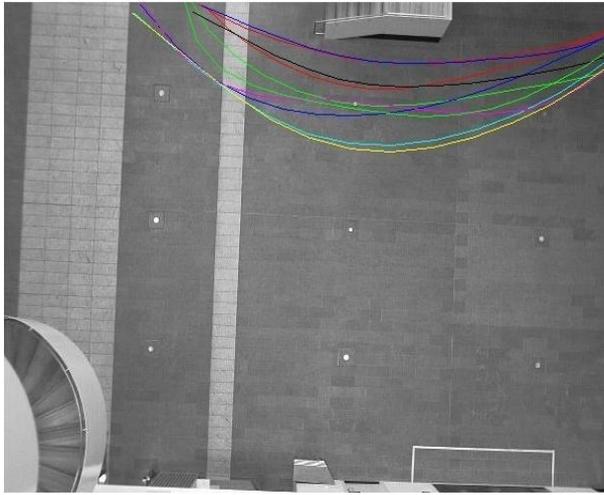
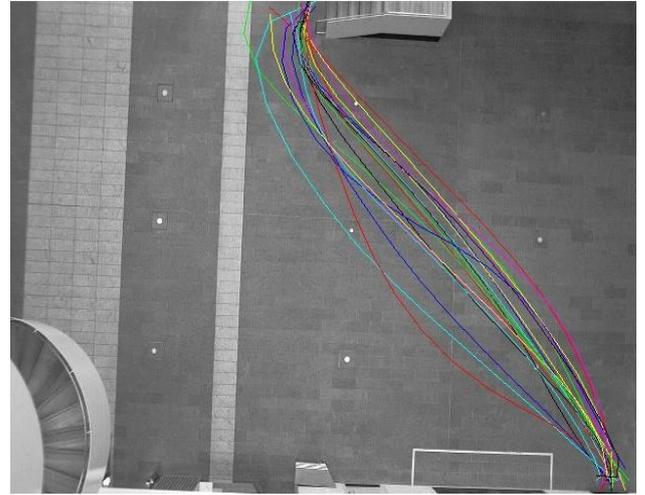


Figure 6.10: figure shows all the clusters those are modelled (107). There two mean vector seems to be abnormal (one at left most corner in red colour and one at right side in green colour) because around 40 trajectories were there for each but normal were very few, so abnormal trajectories were used in modelling.



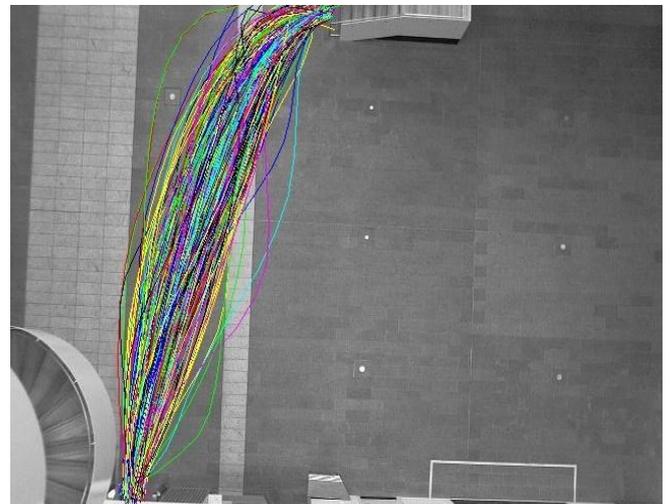
a) Cluster 2 to 5



b) Cluster 3 to 8



c) Cluster 11 to 2



d) Cluster 3 to 11

Figure 6.11: figure shows four different clusters for different paths given above.

# Chapter 7

## Conclusions and future work

Tracking of persons is a difficult task. The tracker has to overcome with the problems of the detector (ambiguities in the data and problems of occlusion and lost detections). The tracking algorithm is used to track the person and form a trajectory from the detector output file. The tracking program produces a file of tracked persons' trajectories. All trajectories have different lengths depending upon how much time the person stays in view of the camera. Normally a person stays in view of the camera for 10 to 15 seconds. Which produces trajectories with 50 150 detections. Overall the tracking process was very satisfactory but some modifications can be done when more than two people merge together to increase the performance of tracker component. To describe each trajectory with an equal number of attributes a Spline fitting algorithm is used. The Spline fitting algorithm gives six control points. The control points are represented by a 12 point vector for each trajectory. The Spline fit algorithm was temporal, sp stationary people can produce unbalanced splines. Some other algorithm can be used for spline fitting to increase the performance of the abnormal behaviour detection component such as basing spline fitting on a spatial description.

The abnormality detection process gave very good results. The model was built by gathering 20 normal trajectories for each clusters. Covariance matrices and mean vectors were determined to calculate a multivariate Gaussian probability for the 12 point vector of each trajectory. Choosing the classifier parameters was a difficult task, which required a lot of calculation and manual understanding. We were looking for multiple parameters for decision making, so it was difficult to choose parameters by observing manually. I used several possible combinations of parameter and set the threshold for each and calculated the error rate of false positive and false negative. The Equal Error Rate (EER) algorithm was used to find the appropriate values of the classifier parameters. Three criteria were chosen for detection of bad trajectories. A trajectory is flagged abnormal if it failed any of the criteria. Results show that it has 1.2% equal error rates. This can be decreased by some more experiments, like changing the number of control point and choosing perfect boundary for each entry and exit points.

## References:

- [1] Kirstie Ball, David Lyon, David Murakami Wood, Clive Norris, Charles Raab, A Report on the Surveillance Society September 2006
- [2] <http://www.scotcrim.u-net.com/researchc2.htm>, Crime and Criminal Justice Research Findings no 30, the Scottish office central research unit, last Accessed May 10, 2010.
- [3] H. Dee and S. Valestin. How close are we to solving the problem of Automated visual surveillance? *Machine Vision and Applications*, 2007.
- [4] Barbara Majecka, Statistical models of pedestrian behaviour in the Forum, Msc thesis, School of informatics, University of Edinburgh 2009.
- [5] B. Brown, Police Research Group. *CCTV in Town Centres : Three Case Studies*, 1995.
- [6] Liberty CCTV, 2005. <http://www.liberty-human-rights.org.uk/issues/3-privacy/32-cctv/index.html>, last accessed May 13, 2010.
- [7] R.R. Sillito and R.B. Fisher. Semi-supervised learning for anomalous trajectory detection. In *BMVC08*, 2008.
- [8] A. Datta, M. Shah, and N. Da Vitoria Lobo. Person-on-person violence detection in video data. In *16th International Conference on Pattern recognition*, volume 1, pages 433–438, 2002.
- [9] Robert Fisher, Barbara Majecka, Gurkirt Singh , <http://homepages.inf.ed.ac.uk/rbf/FORUMTRACKING>  
© 2009 Robert Fisher, last accessed May 12, 2010

## Bio data:

Name : Gurkirt Singh

Father Name: Rakha Singh

Degree: Bachelors' of Technology  
in Electronics and instrumentation.

Area of Interest: Computer Vision  
and Robotics.

Email: gurkirt\_singh@yahoo.in  
and guru094@ gmail.com.

Ph. +91-9790369110 and  
+91-9464740436

Permanent Address:

Vill. Padarth Khera ,  
P.O. Dabhi tak Singh,  
Teh. Narwana, Disst Jind,  
Haryana (India). 126116

