A Dissertation on

Multiple camera based object tracking-Obstacle detection

Submitted in partial fulfillment of the requirement

for the award of the degree of

MASTER of ENGINEERING

(Electronics & Communication Engineering)

Submitted by

Bhavin V Kakani

College Roll No: 06/E&C/2K9

University Roll No: 8515

Under the supervision and guidance of:

Prof. Rajiv Kapoor

Dept. of Electronics & Communication

Delhi College of Engineering, Delhi

DEPARTMENT OF ELECTRONICS & COMMUNICATION

DELHI COLLEGE OF ENGINEERING

UNIVERSITY OF DELHI

2009-2011

Certificate

This is to certify that the work contained in this major project entitled "**Multiple camera based object tracking-obstacle detection"** submitted by **Bhavin V Kakani (R. No.-8515)** of Delhi College of Engineering in partial fulfillment of the requirement for the degree of Master of Engineering in Electronics & Communication is a bonafide work carried out under my guidance and supervision in the academic year 2009-11.

The work embodied in this dissertation has not been submitted for the award of any other degree to the best of my knowledge.

Prof. Rajiv Kapoor
HOD
ECE Department (DTU)

Guided by
Prof. Rajiv Kapoor
ECE Department
DTU

Acknowledgement

It is a great pleasure to have the opportunity to extend my heartiest felt gratitude to everybody who helped me throughout the course of this project.

It is distinct pleasure to express my deep sense of gratitude and indebtedness to my learned supervisor **Dr. Rajiv Kapoor** for their invaluable guidance, encouragement and patient reviews. He kept on boosting me with time, to put an extra ounce of effort to realize this work. With their continuous inspiration only, it becomes possible to complete this dissertation.

I would also like to take this opportunity to present my sincere regards to all the faculty members of the Department for their support and encouragement.

I am grateful to my parents for their moral support all the time; they have been always around to cheer me up, in the odd times of this work. I am also thankful to my classmates for their unconditional support and motivation during this work. It would be injustice if I do not mention the names of **Ashish Chittora** and **Amit Kumar** –my two classmates. Both of these helped me a lot to complete this project.

Bhavin V Kakani

M.E. (Electronics & Communication Engineering)

College Roll No. 06/E&C/09

University Roll No.: 8515

Department of Electronics & Communication Engineering

Delhi College of Engineering, Delhi-42

TABLE OF CONTENTS

CHAPTER 3

CHAPTER 4

CHAPTER 5

CHAPTER 6

CHAPTER 7

CHAPTER 8

List of figures

Abstract

In this thesis, novel methods for background modelling, tracking and occlusion handling via multi-camera configurations are presented. Specifically, we have developed a system to first track moving persons in a given scene and generate colour-based models of those persons to accomplish identification at a later time. The tracking is non-invasive meaning that it does not require persons to wear any particular electronics or clothing to be able to track them. Tracking is accomplished using a position-based data association algorithm while the colour modelling is accomplished using a mixture-of-Gaussians statistical model. The expectation-maximization algorithm is used to generate the colour models over a sequence of frames of data; but to track people successfully in multiple perspective imagery; one needs to establish correspondence between objects captured in multiple cameras. We present a system for tracking people in multiple uncalibrated cameras. The system is able to discover spatial relationships between the camera fields of view (FOV) and use this information to correspond between different perspective views of the same person. We employ the novel approach of finding the limits of field of view (FOV) of a camera as visible in the other cameras. Using this information, when a person is seen in one camera, we are able to predict all the other cameras in which this person will be visible.

Chapter 1

1.1 Introduction

The field of machine (computer) vision is concerned with problems that involve interfacing computers with their surrounding environment through visual means. Such problems like surveillance, activity monitoring and gait analysis has an objective to monitor a given environment and report the information about the observed activity that is of significant interest. In this respect, video surveillance usually utilizes electro-optical sensors (video cameras) to collect information from the environment[28 27]. In a typical surveillance system, these video cameras are mounted on fixed positions or on pan-tilt devices and transmit video streams to a certain location, called *monitoring room*. Then, the received video streams are monitored on displays and traced by human operators. However, the human operators might face many issues, while they are monitoring these sensors. One problem is due to the fact that the operator must navigate through the cameras, as the suspicious object moves between the limited field of view of cameras and should not miss any other object while tracking it. Thus, monitoring becomes more and more challenging, as the number of sensors in such a surveillance network increases[26]. Therefore, surveillance systems must be automated to improve the performance and eliminate such operator errors. Ideally, an automated surveillance system should only require the objectives of application, in which real time interpretation and robustness is needed. Then, the challenge is to provide robust and real-time performing surveillance systems in an affordable price. However, machine vision algorithms (especially for single camera) are still severely affected by many shortcomings, like occlusions, shadows, weather conditions, etc[20]. As these costs decrease almost in daily basis, multi-camera networks that utilize 3D information are

becoming more available. Although, the use of multiple cameras leads to better handling of these problems, compared to a single camera.

There are still challenging problems within the surveillance algorithms, such as background modelling, feature extraction, tracking, occlusion handling and event recognition. Moreover, machine vision algorithms are still not robust enough to handle fully automated systems and many research studies on such improvements are still being done.

1.2 Overview of related work

The set of challenges outlined above span several domains of research and the majority of relevant work will be reviewed in the upcoming chapters. In this section, only the representative video surveillance *systems* are discussed for better understanding of the fundamental concept.

Haritaoglu et al. [1] propose a real time visual surveillance system, *W4*, for detecting and tracking multiple people and monitoring their activities in an outdoor environment. The system can identify and segment the objects that are carried by people and can track both objects and people separately. Moreover, it can recognize events between people and objects, such as depositing an object, exchanging bags, or removing an object. Mittal and Davis [2] present a system, *M2 tracker*, that is capable of segmenting, detecting and tracking multiple people in a cluttered scene by using multiple synchronized surveillance cameras located far from each other. The system is fully automatic, and takes decisions about object detection and tracking by the help of evidence collected from many pairs of cameras[23].

Beymer et al. [3] developed a system to measure the traffic parameters. The proposed video traffic surveillance system extracts the 3D positions and velocities of vehicles and processes

the track data to compute local traffic parameters, including vehicle counts per lane, average speeds, lane change frequencies, etc. These parameters, together with track information (time stamp, vehicle type, colour, shape, position), are transferred to the transportation management centers at regular intervals.

Carnegie Mellon University (CMU) and the Sarnoff Corporation (Sarnoff) constructed a video surveillance program, denoted as *'Video Surveillance and Monitoring (VSAM)'* [4][25 24]. The objective of the program is to develop a cooperative multi-sensor video surveillance system that provides continuous information over given environment. The system provides the capability to detect moving objects, classify them as human or vehicle, keep track of people, vehicles and their interactions, as well as classify these activities.

1.3 Scope of the Thesis

This thesis is devoted to the problem of defining and developing the fundamental building blocks of automated video surveillance systems via multi-camera configurations. The fundamental building blocks can be decomposed into two main parts:

The first part consists of the blocks that can be used in single camera configurations as well as in multi-camera configurations. For the first part, initial problem, which is the extraction of the objects or features that are subject to interest in surveillance application, is discussed and background subtraction methods are compared.

The second part is based on the methods which use the advantages of multi-camera systems and utilization of 3D information obtained from overlapping field-of-view(FOV) of different camera network. A major problem for tracking applications, which is occlusion, is

discussed and a novel algorithm to handle occlusions by using two cameras is proposed. Finally, multi-camera tracking and event recognition by using multi-view data are explained.

1.4 Outline of the Thesis

- In Chapter 2, building blocks of single camera surveillance, which are moving object detection and tracking, are discussed. Moving object detection algorithms include frame differencing, eigenbackground subtraction, Gaussian mixture modelling(GMM) are used and simulation results are presented while tracking algorithm include position based data association algorithm.
- In Chapter 3, we chose to use colour as the feature which we will use to distinguish one person from another. The colour modelling is accomplished using a mixture-of-Gaussian statistical model. The expectation-maximization algorithm is used to generate the colour models over a sequence of frames of data.
- In Chapter 4, an algorithm is presented to handle occlusions by using two cameras. We formalize the handoff problem and describe how the relationship between the FOV of different cameras can be used to solve the handoff problem. We describe how this relationship can be automatically discovered by observing motion of people in the environment.
- In Chapter 5, we present the result of the experiment on different test videos.
- In Chapter 6,Conclusion
- The future plans is suggested in Chapter 7.

Chapter 2

2.1 Methodology for tracking & identification

We envision a system of many cameras networked to provide large scale surveillance over a wide area or over a small area with many different points of view. (See Figure 2-1.)

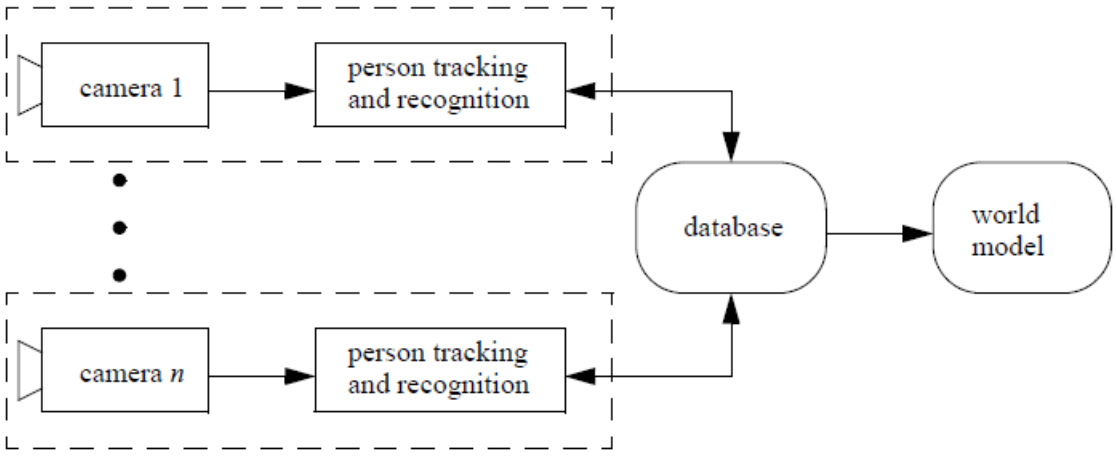


Figure 1-1: System architecture

Each camera would be connected to a system such as that described herein in order to track and identify each person in that camera's field of view. The data from each camera would send critical information about each person to a central database. That same database would provide human identification information to the individual cameras by acting as a query server whereby a camera (and its attached computing) can request identification of an unknown person. If the database cannot make a positive identification, the camera will continue to track the unknown person while it captures the identifying features. When a full identification feature set is developed, the camera hands it off to the database for future use.

2.2 Design basics

We chose to use background subtraction for finding movement regions in the field of view of a fixed camera. Instead of using a simple static background image as Zapata [5] did, we chose to use a statistical model of the background built over time in order to get a better "subtraction" of background from foreground. This system is used by Khan et al. [6] and others. The use of a statistical model rather than an image of the background not only improves the "subtraction" process but also provides a path whereby we can update the background slowly over time simply by updating the statistics using common methods. Updating a static background image is usually a more binary problem: a pixel in the background must be updated in one time step rather than slowly over time unless you wish to generate some "weighting" scheme whereby it updates incrementally.

The tracking system we use is a position tracker best described in radar literature [7]. This basic tracking system--which only uses position to follow a person around the scene is given as the foundational infrastructure on which better tracking can be accomplished. Position tracking is a very low overhead operation which is scalable to many targets. By using tracking in combination with other classification or identification methods, we can reduce the complexity of certain problems.

2.3 Image Processing

2.3.1 Background subtraction to find motion regions

The main purpose of the image processing is to find moving objects in the scene. Because we use stationary, fixed field-of-view cameras, we can use a background image with no moving objects in it as a reference to later determine if anything has changed by subtracting the current frame from a background image which does not have any moving (or changing)

objects in it. We initially used mathematical subtraction of colours in RGB (red-green-blue) colour space. However as noted by Zapata [5], we found that the HSV (hue-saturation-value) colour space was more conducive to reducing the effects of the shadow.

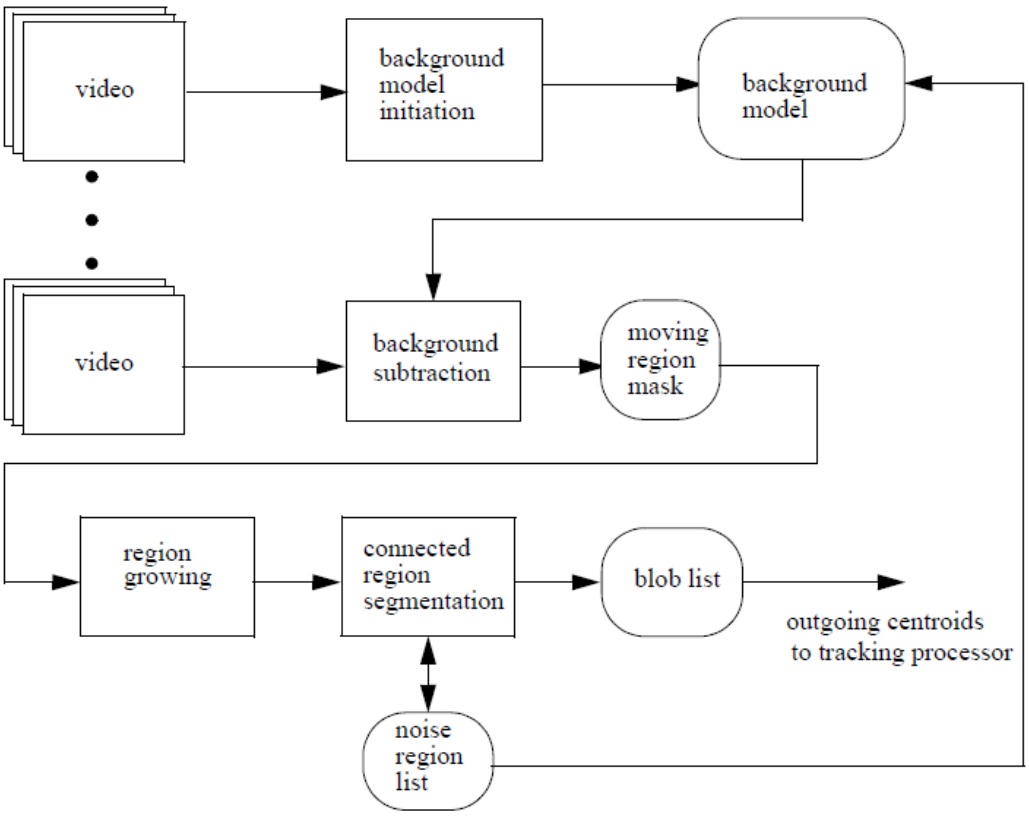


Figure 2-2: Background subtraction and segmentation overview

Mathematical subtraction of colours in HSV space proved a better solution but still caused problems. The biggest problem was caused by how a pixel in the current frame is determined to be foreground (part of a moving object) or background. We made the determination using

$$r = [(H_b - H_f) > t_H \ [(S_b - S_f) > t_S \ [(V_b - V_f) > t_V \ ] \quad (2.1)$$

Where H_b *is* the hue component of the background pixel is, H_f is the hue component of the foreground pixel (pixel from the current frame) and t_H is a threshold. The same pattern holds for the saturation and value variables. If Equation (2-1) evaluates logic true, then the

pixel was considered foreground otherwise background. The problem was that the thresholds (t_H, t_S, t_V) were very difficult to determine experimentally.

To eliminate the need for three thresholds, we chose to do as Khan, et al. [6] and model the background as an array of normal distributions with one distribution per pixel. The first *n* frames of the video sequence are used to initialize the normal distributions. We use a three dimensional vector x_{ij}^k given in Equation (2-2) which represents the colour in HSV colour space at pixel (i, j) for frame *k*. Equations (2-3) and (2-4) compute the mean and covariance of the distribution over *n* frames of data per pixel.

$$x_{ij}^k = \begin{bmatrix} h \\ s \\ v \end{bmatrix} \tag{2-2}$$

$$\mu_{ij} = \frac{1}{n}\sum_{k=1}^{n} x_{ij}^k \tag{2-3}$$

$$\Sigma_{ij} = \frac{1}{n-1}\sum_{k=1}^{n} (x_{ij}^k - \mu_{ij})(x_{ij}^k - \mu_{ij})^T \tag{2-4}$$

When a new frame is to be processed, it is necessary to classify each pixel in that frame as either background or foreground. Using the background model for a given pixel, we can compute the "distance" of that pixel from the mean of the distribution using either the Euclidean or Mahalanobis distance.

$$d_{Euclidean} = \sqrt{\bar{x} - \bar{\mu}} \tag{2-5}$$

$$d_{Mahalanobis}^2 = (\bar{x} - \bar{\mu})^T \Sigma^{-1} (\bar{x} - \bar{\mu}) \tag{2-6}$$

The Mahalanobis distance was chosen because it considers the “spread” of the normal distribution (given by the covariance,) and is therefore more likely to give useful results. Using this method, we are able to get rid of the three thresholds and replace them with one, the maximum distance from the mean of the background model (given by the Mahalanobis distance) that a pixel can be before it is considered foreground. It is easy to find this distance experimentally.

The background “subtraction” process produces a binary mask as shown in Figure 2-2. The white areas are “foreground” or movement whereas the black areas are considered background or unchanged.

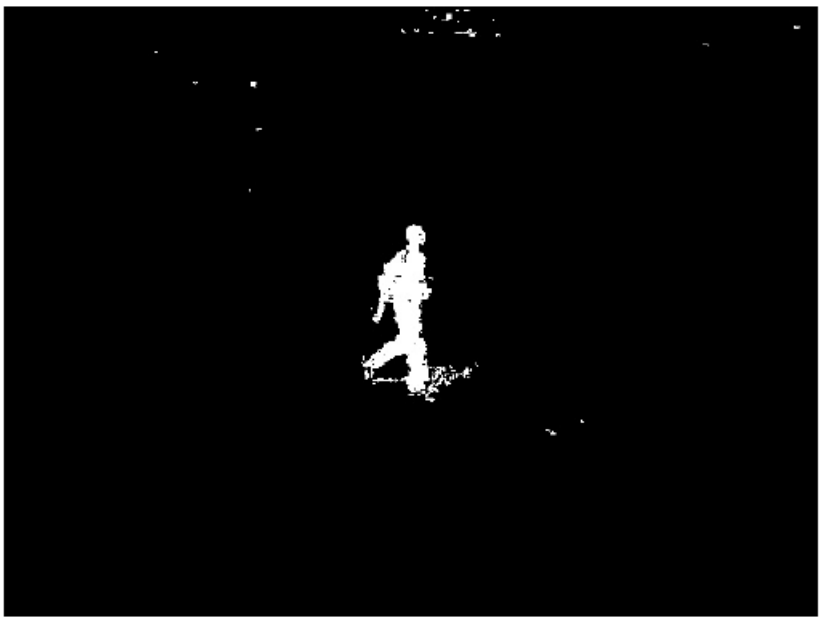


Figure 2-3: Moving region mask

2.3.2 Segmentation

2.3.2.1 Connected region segmentation

The next step is to use the moving region mask as shown in Figure 2-2 to find connected regions of movement. We will define a “blob” as being one connected region of pixels in the

mask image. For example, in Figure 2-2 the mask of the person is one blob which we would like to track. The noise near the top of the frame would also be considered a blob.

We send the binary image mask through a blob detection algorithm. The algorithm does a recursive search through the binary mask finding connected regions of mask pixels. To later recover the pixels in the original image contained in a given blob, a second mask is created. The second mask, known as the "blob identifier mask," begins as a copy of the binary moving region mask. As the blob detection algorithm searches through a connected region of pixels in that mask, it classifies each pixel as belonging to a unique blob and replaces each pixel in the mask with that blob's unique identifier number label.

The blob detection algorithm also returns a list of statistics about each blob:

- size in pixels: n
- extents: $x_{max}, x_{min}, y_{max}, y_{min}$
- centroid: x_{cent}, y_{cent}
- unique identifier label

We will send the centroid information to the track processor. The blob size (in pixels) is used to eliminate objects which obviously cannot be people because they are too small. The centroids of these small blobs are discarded.

2.3.2.2 Improving Image Segmentation

We found that the above image processing algorithm works sufficiently well for most images but was not foolproof. The most common problem we encountered was a person wearing clothing which resembled the background in colour. Figure 2-3 shows how something as simple as a belt blending into the background can cause the moving region

mask for a person to split into two blobs. (The head also disconnects because the subject's neck looked like the background.) When the track processor then encounters two blobs instead of one, it will fail by dropping the track up to that frame.

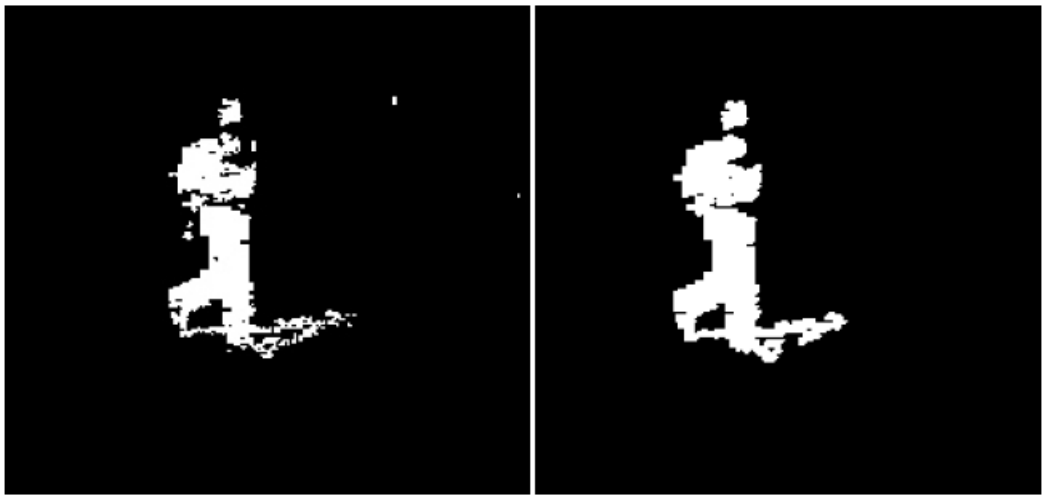


Figure 2-4: Disconnected moving region mask(left) and grown mask(right)

We chose to use a region growing algorithm to re-join separated blobs. The algorithm is very simple: Using the blob identifier mask, each blob is "grown" by searching each pixel in the blob and, if it its neighboring pixels are considered background, it makes those neighbors members of the given blob. Eventually, the entire blob will grow by one pixel-width. The algorithm can be repeated for greater growth but usually one pass works. When the growing is complete, the new blob identifier mask is passed through the blob detection algorithm to find newly connected regions and generate a new blob identifier mask. The image on the right in Figure 2-4 shows a reconnected moving region mask after growing it by one pixel. Note that while the head is still disconnected, the body is now one connected region.

The are two significant drawbacks to using a region growing algorithm. The first is that it will indiscriminately grow unconnected regions -- two separate people into one. The second is

that by growing the blobs without regards to background, there is a distinct possibility of growing background pixels or shadows into the blob.

2.4 Tracking moving objects

This chapter presents a generic framework for accomplishing moving target tracking using only position information of the target. The goal of object tracking is to be able to do three things: initiate a track, continue an initiated track and drop a track at the appropriate time. These three steps will serve as the basis for tracking any moving object and combine to form a robust mechanism known as the 'track processor' on which object recognition can be built.

Figure 3-1 shows the general flow of data through the track processor. The incoming data to be tracked, as stated earlier, is merely the point centroid of a bigger object as determined by the image processing and segmentation described in Chapter 2.

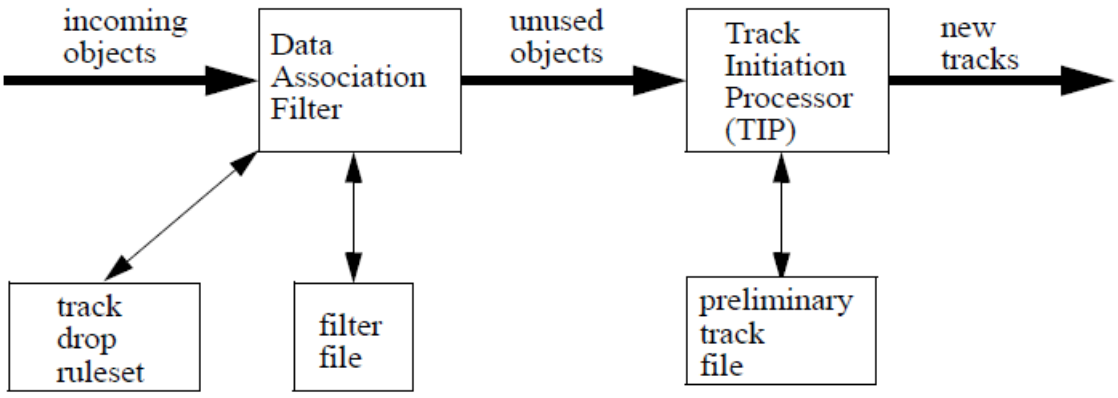


Figure 2-5: Basic operation of the tracking processor

2.4.1 Track initiation

The first step in tracking is determining what to track. The main problem to solve is that each frame of data from the image segmentation described in Chapter 2 has noise scattered

in with the legitimate point object(s) (our "signal") we wish to track. For any single frame, there is no way to distinguish between noise and signal from the given position data alone. However, using a series of frames we can find our signal if we make three assumptions: Firstly, that the objects we wish to track move with constant velocity over a small number of frames. Secondly, that the noise found with our object is random. Thirdly, that our objects move with an upper- and lower- bounded constant velocity. The assumptions are all valid for the average person walking or even running as it is unlikely they will change velocity or direction very often.

Using these three assumptions we can use a standard algorithm from radar known as "retrospective processing" [7] with some changes for greater speed. The idea of this processor (known here as the Track Initiation Processor or TIP) is to match objects in a frame X with those in frames X+1, X+2,...,X+n following our three assumptions. The objects from frame X are all initially assumed to be our signal, that is, there is no noise. We will revisit this data later to eliminate the noise retrospectively. Given our third assumption, that all objects we wish to track move with a bounded upper velocity, we know that they can only move a certain distance in the constant time between two frames of video. Knowing this, we can draw circles of distance around the objects in this initial frame. These circles are the maximum distance each of these objects could possibly travel by the next frame if in fact they are signal objects and not noise. Figure 3-2a shows first frame data (blue dots) from a simulation and the maximum distance circles. Figure 3-2b shows (as red dots) the data from frame X+1. Note the red dots that fall inside the green maximum distance circles. All of these objects are potential signals because they fall within that maximum distance from an object from frame X. All other objects in frame X+1 are eliminated as potential signal objects.

Figure 3-2c shows the predicted locations (magenta circles) of objects in frame X+2 if those potential objects from frames X and X+1 really are signals and not noise. Finally, Figure 3-2d shows (as black dots) the objects from frame X+2. Note that only one of them falls within a circle predicted by the assumptions given the data from frames X and X+1. This means that only the three points making up that line follow our three assumptions.

It is useful to note that this processor eliminated five noise objects out of six in each frame. If we keep our three known-good points as a track, we can go back to frames X, X+1 and X+2 and eliminate all of the noise. In practice this wouldn't matter and we would be content with knowing a good track.

There is always the possibility that one of those three points is actually noise. To gain more confidence in the sequence of points before declaring it a good track and passing it along to the next stage, we can do the exact same processing described above over a series of N frames of data. This is easy to accomplish as the frames come in by analyzing frames [X,X+1,X+2] and finding any preliminary tracks from those. (The preliminary tracks -- those tracks which are incomplete are held in the preliminary track file as shown in Figure 3-1.) Then, when frame X+3 arrives, we can analyze frames [X+1,X+2,X+3] and make sure that the preliminary tracks found previously still follow with new data. When frame X+4 arrives, we can again analyze frames [X+2,X+3,X+4] and again make sure that any preliminary tracks found earlier still exist in these frames. By overlapping the frames we analyze it is assured that any objects we track over all five frames (in this example) follow our assumptions. The number of frames to track an object over before declaring it a legitimate signal track can be as little as three or as many as needed. There is a trade-off between a number too high and too low. Using the minimum number (three) makes false positives more likely. However, using many frames means that any noise in the measurement of the centroid of

the objects could cause the processing to fail and it would need to start all over wasting time.

One way to prevent noise in the measurement of centroids from effecting the TIP is to make the matching of frame X+2 data with the predictions of frames X and X+1 very "lenient." The magenta circles in Figure 3-2d give the maximum distance from the predictions that a black point (a frame X+2 objects) could be and still be considered "matched" with the prediction. By making it large you risk noise being mistaken for signal. Making it too small and noise in the measurement of the centroid of an object will cause the match and ultimately the tracking to fail.

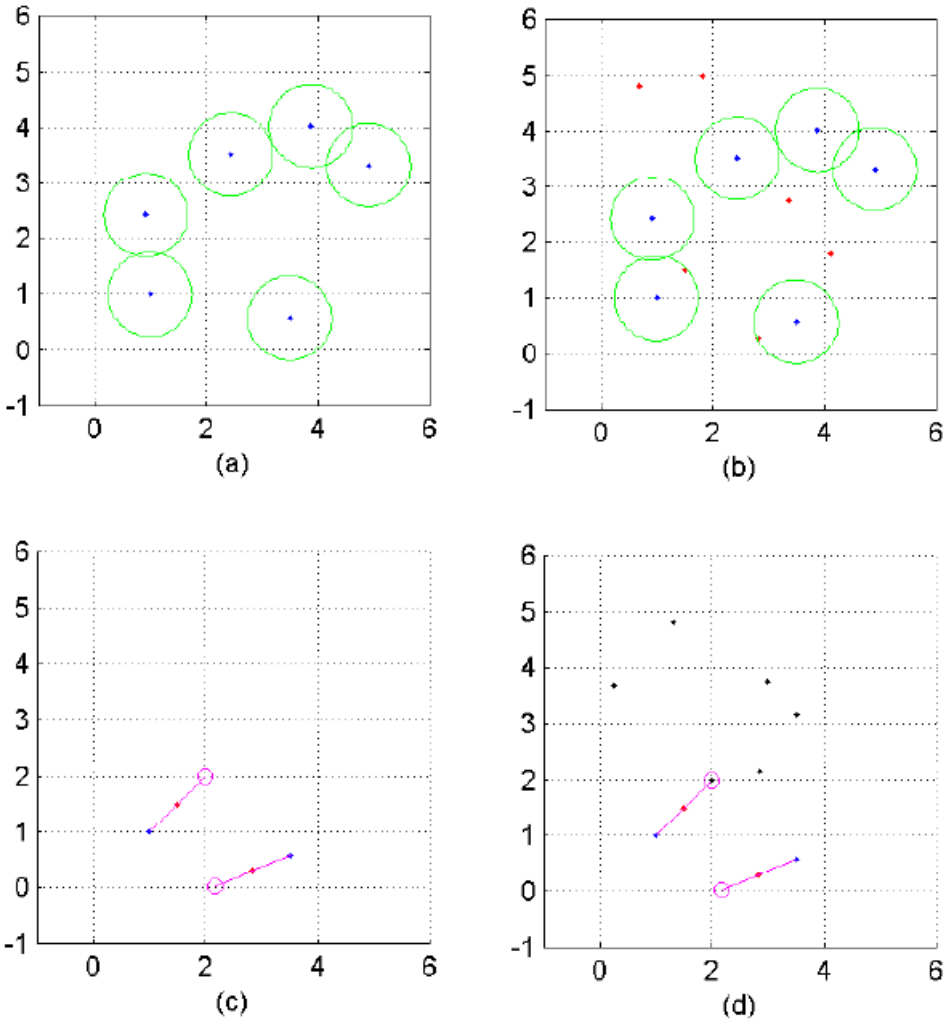


Figure 2-6: Track Initiation Processor (TIP) simulation

2.4.2 Data association

Once a track has been initiated, we need to continue following that object as it moves through the scene. We also want to prevent this object from being mistaken for yet another trackable object by the TIP. Therefore, as shown in Figure 3-1, we look for objects we are already tracking in the data association filter before sending the unused objects on to the TIP for analysis.

2.4.3 Kalman filtering

The data association filter works by matching incoming objects with predictions of where pre-existing tracks indicate that the object should be. We use a simple linear-model Kalman filter with the following parameters:

$$\hat{x}_k = \begin{bmatrix} x \\ y \\ \frac{dx}{dt} \\ \frac{dy}{dt} \end{bmatrix} \quad A = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{2-6}$$

As can be seen above, the state of the system $(\hat{x}_k)$ is defined by the location (x, y) of the object and its velocity $(\frac{dx}{dt}, \frac{dy}{dt})$. The state update matrix (A) contains the frame update rate of the camera (dt). We use the standard formulation for the linear Kalman filter:

$$\hat{x}_k^- = A\hat{x}_{k-1}$$

$$P_k^- = AP_{k-1}A^T + Q$$

$$K_k = P^-_k H^T (HP_k^- H^T + R)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

$$P_k = (I - K_k H)P_k^-$$

For each trackable object returned by the TIP, a new Kalman filter is added to the "filter file". The Kalman filter is converged (or "seeded") with known good data by using the entire track found by the TIP. In this way the filter will be prevented from returning bad data before it has a chance to update its state.

The operation of the data association filter is very simple. As objects (noise and signal) enter from the image processing and segmentation code, the data association filter determines which objects are being tracked by which filter. It does this by making a prediction of the state of the system (the location of the object) in frame X using data from frames 0..X-1. When frame X enters the system, it matches those predictions with the objects.

Figure 3-3 depicts a common problem for the data association filter. Given the five initial points of an object which the TIP says is a signal (shown as black points), the data association filter must classify the incoming data (white points) as data or noise. In this example, there is an object that is clearly the one associated with the initial track. That point will be used to update the state of the Kalman filter for that track. The other three points will be ignored.

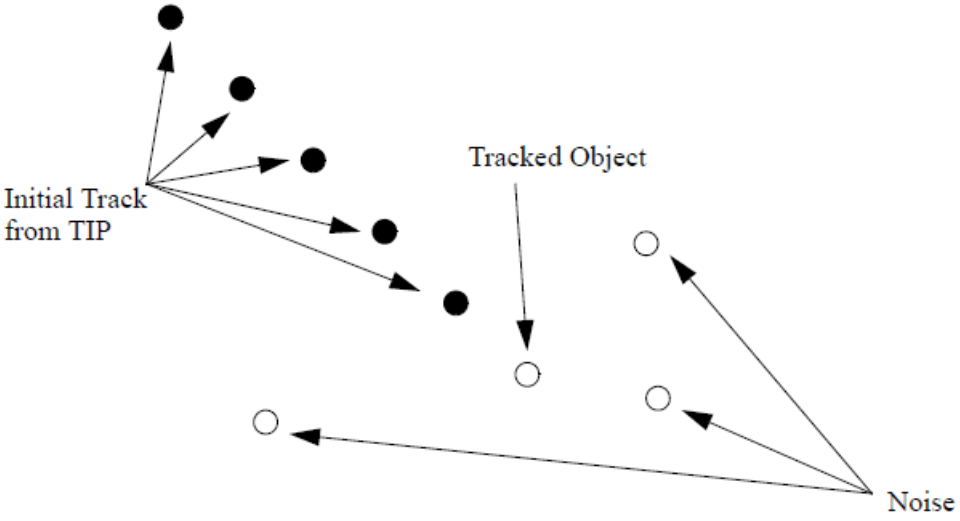


Figure 2-7: The data association problem

2.4.4 Nearest neighbour data association

The question arises of how to associate the predictions given by the Kalman filter with the incoming objects. If a noise object is mistakenly associated with a filter tracking an object, the track will probably be lost so it is critical that the data association be as robust as possible.

There are many methods for choosing which data is associated with which prediction. The most common method is nearest neighbour association and this is the method we chose. The Kalman filter, in addition to predicting the next state of the system, gives a error measure for that state, P_k. Using that error measure, we determine the Mahalanobis distance from the prediction to each of the incoming objects. The object nearest the prediction is associated with the filter that made the prediction.

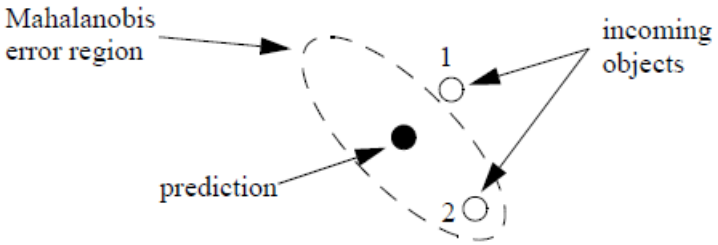


Figure 2-8: Mahalanobis distance versus Euclidean distance

We chose to use the Mahalanobis distance because the Euclidean distance can be incorrect depending on the error in the measurement of the centroids of each object. Using the prediction error measure, , a situation as in Figure 3-4 can arise where the Euclidean distance is smaller than the Mahalanobis distance (the ellipse). However, based on the error surface returned by the Kalman filter, object number two is more likely to be associated with the filter which gave the prediction. Therefore, we use the Mahalanobis distance to determine the nearest neighbour. One problem with a simple nearest neighbour approach is that if a new object appears (noise or a new signal) or an object disappears (noise is removed or an object leaves the scene) we encounter a situation where the number of filters is unequal with the number of incoming objects to associate with. In the case where there are too few objects, we simply assume that the filter which doesn't get data associated with it will be dropped. (See the next section on track drop.) In the situation where there are more objects than filters, we need to "gate" the association to prevent a filter from becoming associated with any object on the screen.

Gating is a simple way of saying that a filter will not be associated with data if that data is too far from the prediction of that filter. This prevents a common problem where the object a filter is tracking disappears (usually because it leaves the scene) so the filter searches for any available object (including noise) to be associated with. By limiting the search to a region near where the filter thinks its tracked object should be, we prevent noise from interfering. It will also help when it comes time to drop a track.

2.4.5 Track Drop Rules

Identifying the situations in which a track should be dropped -- that is, when it should be

realized that the object it is tracking no longer exists in the video sequence -- is a difficult problem. It must be solved with respect to the overall mission of the tracking software.

We use one simple track drop rule which works well for our application. The rule is that is a prediction from a given tracking filter goes unmatched to an object for *N* frames, that track with the associated filter will be dropped. *N* must be chosen based a number of factors: Firstly, if *N* is too low, a track may be dropped because of noise in the measurement of the centroids of the object it is tracking is too low and the filter won't have time to try to recover the object if it is lost because of noise. If *N* is too high, the filter may become associated with noise because the filter will essentially be waiting for an object (including noise) to classify as the object it has been tracking.

Through the dynamics of the track initiation processor, data association and the track drop rules, we are able to follow moving objects through a scene with great robustness even in the presence of noise. By carefully choosing how we implemented the track initiation, we made sure that only valid moving objects were tracked. Also choosing how quickly the tracks are dropped made sure that noise was not unintentionally classified as a signal.

Chapter 3

3.1 Colour modelling

3.1.1 Introduction

We chose to use colour as the feature which we will use to distinguish one person from another. There are many other features we could have used such as face recognition but none that are as obvious or as simple as colour. In a real-world system where positive identification is necessary, face recognition would be a much better identification scheme. The goal of our work is to develop a fast, robust method of statistically modelling the colours contained in the pixels of the person as segmented by the algorithms described in Chapter 2. The colour model has to be able to be generated (or 'trained') quickly but must also provide a way to compare the colour of incoming pixels to an already-generated colour model for recognition purposes. The colour model must also be practically usable in that each model generated will have to be stored in a database.

3.1.2 Gaussian mixture modelling

Figure 3-1(a) shows a typical person segmented from a video sequence by the image processing described in Chapter 2. Figure 3-1(b) shows a graph of each pixel in that image mapped in RGB (red-green-blue) colour-space. We need to be able to model those pixels in RGB colour-space efficiently and quickly over a series of video frames.

The model we chose is a gaussian mixture model. In this model, we assume that our data (the colours in colour-space) is statistically spread as *k* gaussians. Figure3-1(c) shows the data from the segmented image modelled as three gaussians (red, green and blue ellipses.) As can be seen, two of the three gaussians (red and green) overlap significantly because most of the data falls in one region of colour-space. The rest of the data is modelled by the remaining ellipse (blue) which has a greater spread because the data it models is much more spread than the data modelled by the green and red ellipses.

3.1.3 Expectation Maximization(EM) algorithm

In order to generate the mixture-of-gaussians models, we use the expectation maximization (EM) algorithm. The EM algorithm is a method of producing maximum-likelihood paramter estimates for mixtures of exponential distributions. In our application, we will use a mixture of *k* Gaussian distributions over *n* vectors of data. For this model, the individual component densities are given in Equations (3-1) and (3-2),

$$p(x|\phi_i) = p(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu_i)^T\Sigma^{-1}(x-\mu_i)\right] \quad (3\text{-}1)$$

$$\phi_i = \{\mu_i, \Sigma_i\} \quad (3\text{-}2)$$

where μ_i and $\sum_i$ give the mean and covariance for the *i*th component density, respectively.

The EM algorithm applied to the mixture-of-Gaussians problem produces three parameter update equations found in [8],

$$\overline{P(\omega_i)} = \frac{1}{n}\sum_{j=1}^{n} P(\omega_i|x_j, \Theta), i \in \{1, 2, ..., k\} \tag{3-3}$$

$$\overline{\mu_i} = \frac{\sum_{j=1}^{n} P(\omega_i|x_j, \Theta)x_j}{\sum_{j=1}^{n} P(\omega_i|x_j, \Theta)}, i \in \{1, 2, ..., k\} \tag{3-4}$$

$$\overline{\Sigma_i} = \frac{\sum_{j=1}^{n} P(\omega_i|x_j, \Theta)(x_j - \overline{\mu_i})(x_j - \overline{\mu_i})^T}{\sum_{j=1}^{n} P(\omega_i|x_j, \Theta)}, i \in \{1, 2, ..., k\} \tag{3-5}$$

Where,

$$\Theta_i = \{\phi_i, P(\omega_i)\} \tag{3-6}$$

are the maximum likelihood parameters we wish to estimate and where the *d*-dimensional data $X = \{x_j\},\ j \in \{1,2, \dots, k\}$ is assumed to be taken from the probability density function,

$$p(x|\Theta) = \sum_{i=1}^{k} p(x|\phi_i)P(\omega_i) \tag{3-7}$$

Where $P(\omega_i)$ gives the probability of the *i*th component density, $(x|\varphi_i)$.

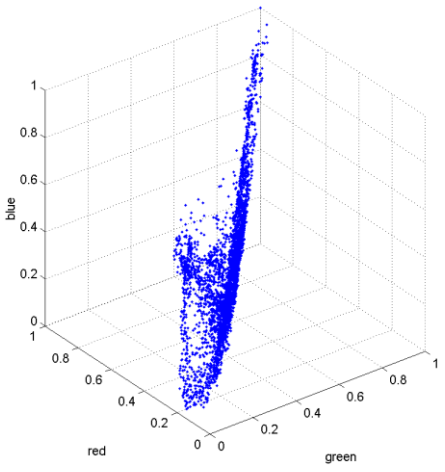
In order to develop an algorithm for computing the mixture model, we need to compute $P(\omega_i|x_j, \Theta)$which is expressed by Bayes' rule expansion,

$$P(\omega_i|x_j, \Theta) = \frac{p(x_j|\phi_i)P(\omega_i)}{p(x_j|\Theta)} \tag{3-8}$$

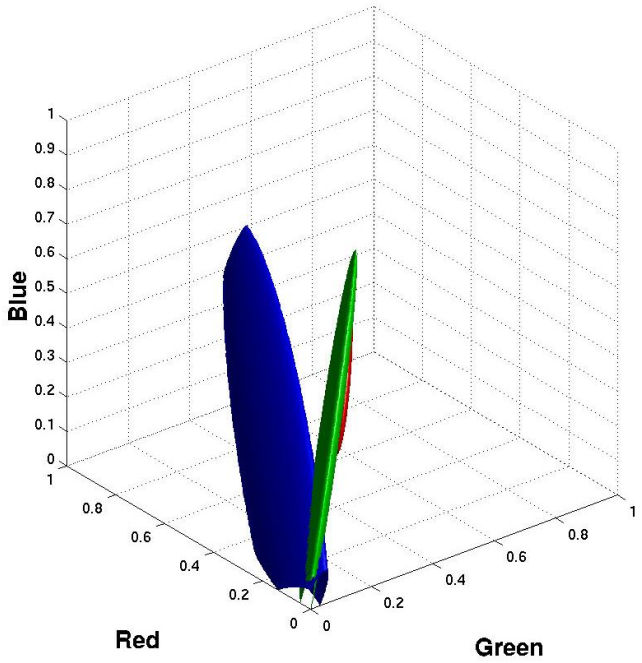
3.1.4 Training the mixture model

When a person is unrecognized by pre-existing mixture models, a new model needs to be trained. If we make the replacement suggested by Equation (3-8) into Equations (3-3), (3-4) and (3-5) we will have three EM update equations capable of being implemented in software. However, notice that although all three update equations share the expression $(\omega_i|x_j,\Theta)$, all three require at least one sum evaluation for one Gaussian. Two factors can therefore be seen to effect the execution time: number of Gaussians in the mixture model and the number of datapoints over which the model will be fit.

(a)



(b)



(c)

Figure 3-1: Example of gaussian mixture modelling. a) Segmented person image from image processing. b) Colour-space diagram for pixels in person image. c) Mixture of three gaussians model for the colour data.

From experimental data, a person image such as that in Figure 3-1 occupies approximately 1000-1600 pixels. Using a mixture model of three Gaussians over that many pixels would be very computationally expensive. For that reason, we chose to subsample the pixel data to a reasonable, representative number. The EM updates are still computationally expensive, however, when it is considered that to converge the model, one needs to iterate those equations many times.

3.1.5 Comparing Colour Data to the Database Model

To determine if person who has just been acquired by the track processor is someone we have seen before, we need to compare that person's colour data to the colour models in the database. This is done simply by finding the sum log-likelihood of the data over the model. If

that likelihood is above a certain threshold (determined experimentally) then we match that person with the model. If none of the models return log likelihood above that threshold, we create a new model for that person as described above.

Chapter 4

4.1 Multiple camera tracking

4.1.1 Overlapping field of view

To cover an area of interest, it is reasonable to use cameras with overlapping FOVs. Overlapping FOVs are typically used in computer vision for the purpose of extracting 3D information[33]. The use of overlapping FOVs, however, creates an ambiguity in monitoring people. A single person present in the region of overlap will be seen in multiple camera views. There is need to identify the multiple projections of this person as the same 3D object, and to label them consistently across cameras for security or monitoring applications.

The two test videos which we have taken for tracking purpose have the following overlapping field-of-view (FOV)[31]:

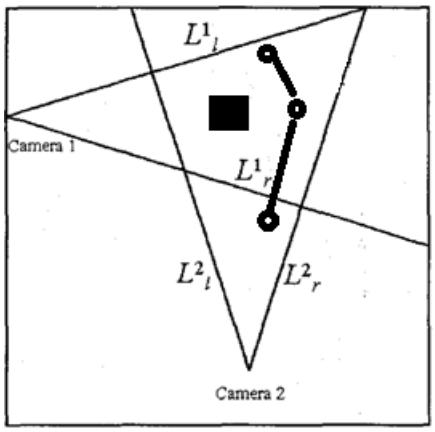


Figure 4-1(a): Field Of View of Test video: 1

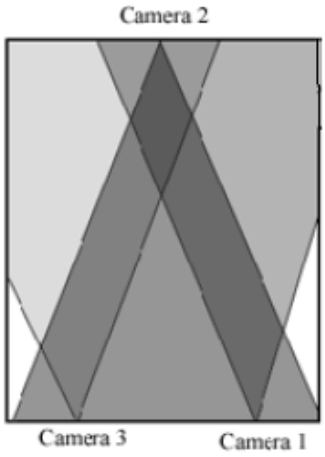


Figure 4-1(b): Field Of View of Test video: 2

4.1.2 Edge of field-of-view (FOV) lines

The handoff problem occurs when a person enters the FOV of a camera[30]. At that instant we want to determine if this person is visible in the FOV of any other camera, and if so, assign the same label to the new view. If the person is not visible in any other camera, then we want to assign a new label to this person. Note here that we could have matched colour features of the persons visible in one view to the new view in other view to find the most likely match; but sometimes the condition is harsh to identify the same person from different views as because different cameras can have different intrinsic parameters as well

as photometric properties (like contrast, colour-balance etc.). Lighting variations also contribute to the same object being seen with different colours in different cameras.

For shallow mounted cameras each FOV's footprint can be described by two lines on the floor-plane, the left and the right limit of FOV. Let L_l^i and L_{rl}^i be the left and right limits of FOV of the i^{th} camera (C_i) on the ground plane (Figure 4.1). Let the projection of $L_x^i, x \in \{l, r\}$ in camera *j* can be noted by L_x^{ij}. Note that L_x^{ij} denotes the left and the right sides of the image in C_i. As far as the camera pair(*i, j)* is concerned, the only locations of interest in the two images for handoff $L_x^{ij}\ and\ L_x^{ji}$. These are up to four lines, possibly two in each camera. Let us currently assume that a person already visible in one of the cameras is entering the FOV of another camera. In this case, all that needs to be done is to look at the associated line in the *other* camera and see when person is crossing that line. The figure below describes more clearly. A person is entering the FOV of *C*2. There isn't any person visible in *C*1 at this instant. This person is being tracked and we have a bounding box around him. By looking at the bottom part of the bounding box, we can determine quite easily when this person has entered the FOV of *C*1. The line that helped us determine this is L_l^{21} i.e. the left FOV of *C*2 as seen in *C*1. The new person in *C*2 is therefore assigned the same label as the one it was assigned in *C*1. Note that we are considering only the left and right edges of FOV in this formulation, which is sufficient for cameras mounted at a low angle of depression.

4.1.3 Occlusion handling

This framework implicitly handles occlusion well. As described in chapter 2 when a new person enters the scene the system models its feature with colour modelling and make a

database for that particular person. Now if the person goes outside the field-Of-View (FOV) or is being occluded with some obstacle then the system tries to find out the view from different camera where he is visible[29]. If there isn't any track able object for *n (found experimentally)* consecutive frames then view automatically switch to the second camera. Now, if the person comes out of the occlusion then again with the help of previously stored database we will identify that person. In case, if the person is trackable in both the scene then we can opt for anyone of the views as best view.

4.1.4 Detection of new person

A person entering from the door (in this case he might just "appear" in the middle of the image) or he might be entering the FOV from a point that is not visible in any other camera. If the camera setup is such that the environment is completely covered, then the latter case will never happen. However, to keep the formulation general, the second case has to be considered too. In the previous case, we looked at the FOV lines of the current camera as seen in other cameras. To find whether a person is visible in other cameras or not, we look at the FOV lines of other cameras as seen in the current camera. Consider the scenario when a person is entering the FOV of C_i. Whether this person is visible in any other camera (C_j, $j \neq i$) or not can be determined by looking at all the FOV lines that are of the form L_x^{ij}, i.e. edge of FOV lines of other cameras as visible in this camera (C_i). These lines partition the image C_i into (possibly over lapping) regions, marking the areas of image C_i that correspond to FOV of other cameras. Thus all the cameras in which current person is visible can be determined by acquiring the region of the person's feet. Thus with each line L_x^{ji}, an additional variable δ_x^{ij} is stored. The value of δ_x^{ij} can either be +1 or –1, depending

upon which side of the line falls inside the FOV of C_j. Then, given an arbitrary point (x', y') in C_i, the point's visibility in C_j can be determined by just determining if this point is on the correct side of both L_l^{ji} and L_r^{ji}. If L_l^{ji} is represented by A x' + B y' + C. The point (x', y') is visible in C_j if and only if

$$sgn\left(L_l^{ji}(x',y')\right) = \delta_l^{ij} \; and \; sgn\left(L_r^{ji}(x',y')\right) = \delta_r^{ij}$$

In the case when only one of the left or right lines of C_j is visible in C_i, the condition in above Equation is simplified to only one of the anded terms.

Chapter 5

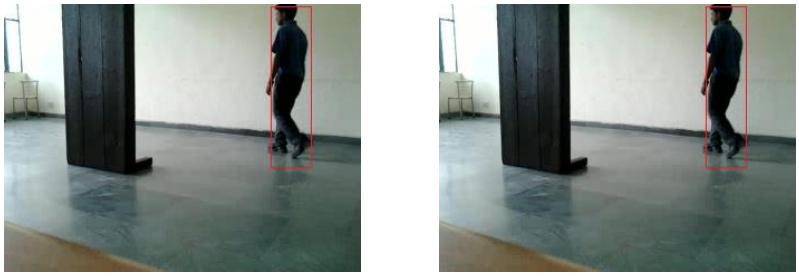
5.1 Experimental results

We have taken view from two cameras and the tracked object is chosen from the two views:

Test video # 1

CAMERA 1 **CAMERA 2** **TRACKED VIEW**











Test video # 2

CAMERA 1 **CAMERA 2** **CAMERA 3** **TRACKED VIEW**




Fra

me = 52




Frame = 57




Frame = 62




Frame = 67




Frame = 352




Frames = 362




Frames 367

Chapter 6

6.1 Conclusion

In this paper, we have combined the concept of mathematics, computer science, image processing along with mathematical geometry to model a method for efficiently tracking a person in multi-view environment with obstacle handling. Suitable changes could make the whole process near realtime.Initially, the statistical distributed frame differencing is used for background subtraction and postion-based data association algorithm is used for tracking of the object in single-view environment. This type of background subtraction and tracking is robust as it intends to remove the effects of shadow and light intensity variations in the background. Now for the identification of the object we have used colour based modelling using Gaussian mixture model (GMM). This is again a weak parameter for identification as the person blob size is variable while in motion and can identify as different person if it tracks from backward; Face recognition technique can be used in place of it but its require high memory allocation along with high picture quality camera. For tracking in multidimensional environment we have used overlapping Field-Of-View(FOV) lines techniques as mentioned by Mubarak S.[9]. In case of occlusion, automatic switching of views have been made for continual tracking of object.

Chapter 7

7.1 Future Aspects

7.1.1 Using Non-Stationary Cameras

In development of the human tracking system described herein, it was realized that moving cameras such as those on pan-tilt units could be used successfully as the image source. The basic problem with using a moving camera and a background "subtraction" scheme as described in Chapter 2 is that the background will change so significantly that the algorithm will see only movement where there really is none. The solution to the problem is twofold. First, we need to qualify the use of moving cameras. A camera continually moving will not work with this algorithm without major modification. However, a camera that moves to specific discrete locations and stays at each for a significant period of time (at least a few seconds) would work.

The reason we can move the camera and still accomplish tracking is because we can make the initial assumption that the first frame we get after moving the camera is the background. This is usually not a valid assumption, however, as there could be people in the scene. Assuming we did use that first frame as our background (or at least as part of our background model) any moving people in that scene would cause two movement regions to appear when the background subtraction was done. The first region would be the person. The second would be where the person was a stationary region caused by the person moving and thus revealing the real background.

7.1.2 Efficient region growing and shrinking

we use a simple region growing algorithm to reconnect objects that became disconnected by a bad background subtraction. The algorithm is currently very inefficient -- having to search the entire moving region mask. A better algorithm could snake around the outside edge of a connected region and only look at those limited number of pixels for pixels to grow.

Another good algorithm to implement would be an efficient region shrinking algorithm to shrink regions after having grown them. This would not, however, cause regions which became connected by the growing to disconnect. It would only shrink pixels that are on the outside edge of a connected region.

One problem with growing one region into another is that the two regions could be unconnected in reality. (Two people walking near one another, for example.) In order to prevent them from becoming connected, before growing one region into another, the algorithm could check the colour models of the two regions. If they are significantly similar, they could be the same region and should be connected. If not, the growing algorithm should be prevented from connecting them.

7.1.3 Camera model for real world coordinates

In order to accomplish tracking over a distance greater than that covered by one camera, a global coordinate system would be necessary to fuse the multiple tracks of the same person onto a map, for example. The algorithm presented in this thesis does not deal with the problem of mapping coordinates in an image to coordinates in a real-world space

or into a synthetic space shared among multiple cameras. The problem is to produce a model of what the camera sees -- a “camera model” -- and using that to map locations in the image to locations in a different coordinate frame. Work has been done using these camera models [12,20] to generate three-dimensional coordinates of an object from two cameras. A similar method would be used to recover two-dimensional coordinates from one or more cameras.

7.1.4 Speed improvements

One simple method of improving how long it takes to analyze a single video frame is to use a significantly sub-sampled image to find motion regions of interest before looking to the whole frame for details. For example, instead of using a whole 320x240 image, we could subsample that to 160x120. We could do an initial background differencing on that smaller image to find regions we will then difference in the bigger image. This method could save a significant amount of time by only differencing those regions where movement is likely[32].

Chapter 8

7.1 References

[1] I. Haritaoglu, D. Harwood, and L.S. Davis, "W4: Real-Time Surveillance of People and Their Activities", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 8, pp. 809-830, 2000.

[2] A. Mittal and L. S. Davis, “M2Tracker: A Multi-view Approach to Segmenting and Tracking People in a Cluttered Scene”, *International Journal of Computer Vision*, Vol. 51, No. 3, 2003.

[3] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik, "A Real-Time Computer Vision System for Measuring Traffic Parameters", *Proceedings of IEEE Computer Vision and Pattern Recognition*, pp. 495-501, 1997.

[4] T. Kanade, R. Collins, A. Lipton, P. Burt, and L. Wixson, "Advances in Cooperative Multi-Sensor Video Surveillance", *DARPA Image Understanding Workshop*, pp. 3-24, November 1998.

[5] I. Zapata, “Detecting Humans in Video Sequences Using Statistical Color and Shape Models,” Master’s Thesis, Department of Electrical and Computer Engineering, University of Florida, 2001.

[6] S. Khan and M. Shah, “Tracking People in Presence of Occlusion,” Asian Conference on Computer Vision, Taipei, Taiwan, January 2000.

[7] E. Brookner, *Tracking and Kalman Filtering Made Easy*, Wiley, New York, 1998, pp. 111-116.

[8] M. Nechyba, “Maximum-Likelihood Estimation for Mixture Models: the EM algorithm,”

EEL6935 Fall 2001 Class Notes, University of Florida, 2001.

[9] S. Khan et al., “Camera hand off: tracking in multiple uncalibrated stationary cameras”, IEEE,0-7695-0939-8,University of central Florida, 2000.

[10] Q. Cai, A. Mitiche, and J. K. Aggarwal. “Tracking Human Motion in an Indoor Environment,” Second Intl. Conf. on Image Processing, pp. 215-218, Washington D.C., October 1995.

[11] A. J. Lipton, H. Fujiyoshi, and R. S. Patil, “Moving Target Classification and Tracking from Real-Time Video,” *Proc. IEEE Image Understanding Workshop*, pp. 129-136, 1998.

[12] M.H. Yang, N. Ahuja, "Gaussian Mixture Model for Human Skin Color and Its Application in Image and Video Databases," *Proc. of the SPIE*, vol. 3656: Conf. on Storage and Retrieval for Image and Video Databases (SPIE 99), pp. 458-466, San Jose, Jan., 1999.

[13] C. Beumier, M.P. Acheroy, “Automatic Face Authentication from 3D Surface,” *Proceedings of the British Machine Vision Conference BMVC 98*, University of Southampton UK, 14-17 Sep, 1998, pp. 449-458.

[14] R. Brunelli and D. Falavigna. "Person Identification Using Multiple Cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 10, pp. 955-966, October (1995).

[15] J. Sobottka and I. Pittas, “Segmentation and Tracking of Faces in Color Images,” *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pp. 236-241, 1996.

[16] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. “Human Face Detection in Visual Scenes,” Technical Report CMU-CS-95-158R, School of Computer Science, Carnegie Mellon University, 1995.

[17] M.H. Yang, N. Ahuja, "Gaussian Mixture Model for Human Skin Color and Its Application in Image and Video Databases," *Proc. of the SPIE*, vol. 3656: Conf. on Storage and Retrieval for Image and Video Databases (SPIE 99), pp. 458-466, San Jose, Jan., 1999.

[18] G. Welch and G. Bishop, “An Introduction to the Kalman Filter,” TR 95-041, Department of Computer Science, University of North Carolina at Chapel Hill, 2001.

[19] M. Nechyba, “Maximum-Likelihood Estimation for Mixture Models: the EM algorithm,” EEL6935 Fall 2001 Class Notes, University of Florida, 2001.

[20] W. Hu, T. Tan, L. Wang, and S. Maybank, "A Survey on Visual Surveillance of Object Motion and Behaviors", *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, Vol. 34, No. 3, pp. 334-352, 2004.

[21] V. Kettnaker and R. Zabih, “Bayesian Multi-camera Surveillance”, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 253-259, 1999.

[22] S. L. Dockstader and A. M. Tekalp, “Multiple-camera Tracking of Interacting and Occluded Human Motion”, *Proceeings of IEEE*, Vol. 89, No. 10, pp. 1441-1455, Oct. 2001.

[23] R. T. Collins, A. J. Lipton, H. Fujiyoshi, and T. Kanade, “Algorithms for Cooperative Multi-sensor Surveillance”, *Proceedings of IEEE*, Vol. 89, pp. 1456-1477, Oct. 2001.

[24] J. Black, T. Ellis, and P. Rosin, “Multi-view Image Surveillance and Tracking”, *Proceedings of the Workshop on Motion and Video Computing*, 2002.

[25] T. H. Chang and S. S. Gong, “Tracking Multiple People with a Multicamera System”, *IEEE Workshop on Multi-Object Tracking*, pp. 19-28, July 2001.

[26] T. H. Chang, S. S. Gong, and E. J. Ong, “Tracking Multiple People Under Occlusion Using Multiple Cameras”, *British Machine Vision Conference*, September 2000.

[27] G. T. Kogut and M. Trivedi, “Maintaining the Identity of Multiple Vehicles as They Travel Through a Video Network”, *IEEE Workshop on Multi-Object Tracking*, pp. 29-34, July 2001.

[28] O. Javed, Z. Rasheed, A. Alatas, and M. Shah “KNIGHT: A Real Time Surveillance System for Multiple Overlapping and Non-Overlapping Cameras”, *International Conference on Multimedia and Expo*, 2003.

[29] O. Javed, Z. Rasheed, K. Shafique, and M. Shah “Tracking Across Multiple Cameras With Disjoint Views”, *IEEE International Conference on Computer Vision*, pp. 952-957, 2003.

[30] S. Khan, O. Javed, Z. Rasheed, and M. Shah, “Human Tracking in Multiple Cameras”, *IEEE International Conference on Computer Vision*, pp. 331-337, July 2001.

[31] S. Khan, O. Javed, and M. Shah, “Tracking in Uncalibrated Cameras with Overlapping Field of View”, *2nd International Workshop on Performance Evaluation of Tracking and Surveillance*, December 2001.

[32] C. Stauffer and W. E. L. Grimson, "Learning Patterns of Activity Using Real-Time Tracking", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 8, pp. 747-757, August 2000.

[33] K. K. Lee, M. Yu, and Y. Xu, “Modeling of Human Walking Trajectories for Surveillance”, *Intelligent Robots and Systems*, Vol. 2, pp.1554-1559, Oct. 2003.