HUMAN ACTIVITY TRACKING FOR WIDE-AREA SURVEILLANCE

BY

PATRICK D. O'MALLEY

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2002

ACKNOWLEDGMENTS

I would like to thank Dr. Doty and Dr. Arroyo for allowing me to do pretty much whatever I wanted in their lab for most of my undergraduate career. I would also like to thank Dr. Arroyo for allowing me to be involved in the inner-workings of the lab. It was a real learning experience. Of course, I have to thank Dr. Nechyba for paying me to do whatever I wanted and putting up with my attempts to avoid his work at all costs. Teaching that machine intelligence class so I could graduate was really nice, too. Thanks go to Drs. Schwartz, Gugel, Arroyo and Nechyba for making sure I was not fed to the lions. For all the support from my friends here in Gainesville I am deeply grateful. Lastly, I would like to thank my family--parents especially--who did not bother me too much about little things like graduating and who offered me consistently good advice which I promptly ignored.

TABLE OF CONTENTS

page

LIST OF FIGURES

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

HUMAN ACTIVITY TRACKING FOR
WIDE-AREA SURVEILLANCE

By

Patrick D. O'Malley

May 2002

Chairman: Dr. Michael C. Nechyba
Major Department: Electrical and Computer Engineering

We present a method for tracking and identifying moving persons from video images taken by a fixed field-of-view camera. Specifically, we have developed a system to first track moving persons in a given scene and generate color-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 color modeling is accomplished using a mixture-of-Gaussians statistical model. The expectation-maximization algorithm is used to generate the color models over a sequence of frames of data in order to make the entire process near-real-time. Once a color model is developed for a given person, that model will be placed in a database where it will be used for future identification. Unlike other color-based tracking systems, this system will compare an unrecognized person against the database only once and thereafter use the position-based tracker to continue following the person. We predict that the entire system will run at upwards of 15 frames per second on a reasonably fast computer.

CHAPTER 1
INTRODUCTION

1.1 Objectives

Applications for human tracking systems have in recent years exploded as the wide-ranging capabilities of such systems have been imagined. Video surveillance and security are immediate beneficiaries of human tracking technology by alleviating the need for humans to constantly monitor a myriad of security cameras. Combining tracking technology with gesture recognition could enable computers to automatically detect intrusions into secure areas or identify suspicious behavior. Another area of great interest which has already seen computer generated viewer-aids is in sporting events. Being able to track and recognize the players could help give viewers interactive information about their favorite athlete as the play continues.

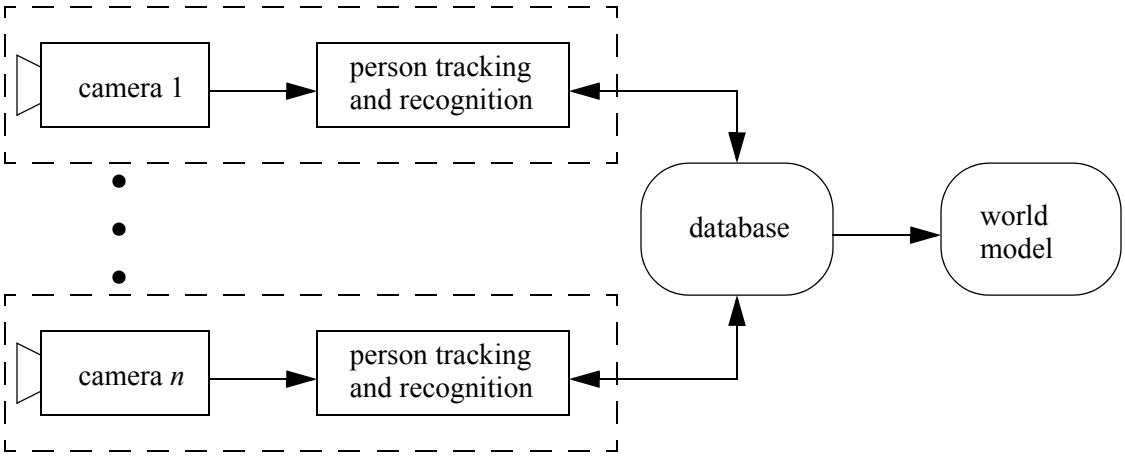


Figure 1-1: System architecture

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 1-1.) 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.

The central database would serve as an effective means of recreating the scenes of each individual camera into one synthetic map of tracked persons over a wide area. A wide-area surveillance network could be utilized on the perimeter of a military installation, for example, or in a location already well-covered with cameras such as metropolitan centers or malls.

1.2 Related Work

1.2.1 One- and Two-step Methodologies

Human detection schemes can be classified as one-step and two-step processes. The one-step methods such as that proposed by Rigoll et al. [1] detect the human form from the background by means of statistical or other methods. Typical two-step methods (the most common) first find movement regions by comparison of the background to an incoming frame (the "foreground") and looking for differences between the two which would indicate movement. The second step in the process is to apply classifiers to the segmented regions in order to detect the human form. Typical two-step schemes include those by Khan et al. [2], Cai et al. [3] and Lipton et al. [4].

There are two approaches to motion detection for the first step in the two-step method. The first and most common relies on a background which does not change significantly from one frame to the next. In this method, the foreground frame is compared to the background model with any discrepancies between them assumed to be caused by an object moving in front of the mostly static

background. This method is used by Khan et al. [2] and Cai et al. [3]. The second method is to use consecutive frame differencing wherein each new incoming frame is subtracted from the previous frame. This differencing method is much harder to use because it sometimes produces misshapen or disconnected movement regions. However, it does eliminate the need for a background model of any sort. Lipton et al. [4] apply this method in their work.

1.2.2 Tracking Systems

Tracking, in general, consists of being able to identify a moving object as it changes position over time. There are several methods that can be used to accomplish tracking. The classical method from radar applications is to use position-based tracking [5]. In this method, the only way to identify a moving object as being the same object now as in the future is its current and future positions. Position is therefore the classifier. Another method proposed by Khan et al. [2] and Nechyba [6] wherein the color of each moving object is the classifier. In these systems, identification over a series of video frames is accomplished by comparing the moving objects to *a priori* determined color models. The last method of tracking is to use form-based classification as proposed by Lipton et al. [4], Haritaoglu et al. [7], Haritaoglu et al. [8] and Ju et al. [9]. In this method, physical templates of the human form are compared to the motion regions detected in the first step. The templates are temporal in that they change with time to depict a person walking, for example. Each tracked person is followed and distinguished from other people based on the current configuration of their template.

1.2.3 Individual Person Identification

The problem of tracking a person can in certain instances be used to also solve the problem of identifying individual persons. However, the identification problem is different from the tracking problem in that identification must be possible even if the tracked person leaves then re-enters the field of view of the camera. That is, the identification problem exists over an indefinite time

period whereas the tracking problem can usually be restricted to only the period in which the person is being viewed by the camera.

Methods for classifying people based on skin color as done by Yang et al. [10] and Yang et al. [11] have the capability at some level of acting as identifiers. Full color can also be used [12]. However, most person identification schemes as implemented by Beumier et al. [13] and Brunelli et al. [14] work with close-in facial images. Some work has been done by Sobottka et al. [15] in segmenting and tracking useful faces from images. A face detection system such as that of Rowley et al. [16] in conjunction with person tracking could aid in identification.

1.3 Approach

1.3.1 Introduction: Design Basics

Our approach to person detection, tracking and identification is fundamentally split into three steps as opposed to one or two as is more common. We do not explicitly solve the person detection problem either but rather assume for the moment that we will be tracking people. (It is not a limitation however as the system described herein will track any object!) We chose to use a position-based tracker to track moving objects. Identification is accomplished by a simple color-modeling algorithm that again assumes we are tracking a person. (And again, it is not limited to identifying persons but rather will identify anything it is trained to.)

We chose to use background subtraction for finding movement regions in the field of view of a fixed camera. (We may also be able to use a movable camera as described in Section 6.1.) Instead of using a simple static background image as Zapata [12] 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. [2] 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 wereby it updates incrementally.

The tracking system we use is a position tracker best described in radar literature [5]. 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.

The classification method we chose for identifying individuals entering and leaving a scene is color modeling. When a person (or other object) enters a scene, it is compared against the color model database and either classified by a pre-existing model or a new model is started. The new model will be generated over a series of frames while the person is being tracked by the position-based tracker. When the color model is sufficiently trained, it will be sent to the central color database where it will be available if that person is seen again by the system. The person is only compared against the database once because the position-based tracking system is assumed to keep a “lock” on that person and that person alone.

We chose to use a mixture-of-Gaussians color model for the person identification similar to that done by Yang et al. [17] for skin color modeling and Khan et al. [2] for person tracking. To train the mixture model, we use the expectation-maximization (EM) algorithm. As the position-based tracker follows one person, the EM algorithm will be training a new mixture model for that person if one is not found in the database that matches the color of this person. To allow the EM algorithm enough time to train properly as it is an iterative algorithm but at the same time continue tracking possibly in real-time, we use a low number of iterations over a long sequence of frames. In order to assure we correctly model all of the colors of the person, we also change the data it

trains on each frame. The training ends when the log-likelihood of independent (that is, non-training) data plateaus.

1.3.2 Problems With Other Tracking Methods

There are significant reasons we chose positon-based tracking with color-based identification as the foundational system. In systems described by Khan et al. [2], color is used to track the people through a scene. The drawback to this method is that each object (person) in a scene must be compared to all known color models each frame because the assumption is that each object "seen" at a given time could be a new object. Fundamentally, no history is used to prune the search tree when trying to match the moving person to a particular color so each person in a scene must be tested against each color model. For small numbers of persons, this would not be a problem. However, in our application, there is a potential for tens if not hundreds of models in the database. Therefore separating the tracking problem from the identification problem gives us a much more scalable architecture. By comparing a person to the color model database only once, we reduce the overhead on the computer systems.

Again, a problem with the color tracking is in occlusion resolution. Khan et al. [2] and previous work done at the University of Florida by Nechyba [6] use pixel-wise classification of regions to separate occluded objects (persons by Khan et al. and cars by Nechyba) into their respective categorizations so as to be able to recover some position information. The problem with this apprach is that in an unconstrained environment -- one without limitation on the number of objects in the scene -- it is impossible to first identify that occlusion occurred and then to determine which objects out of the color database occluded. It can be shown that this method leads to an intractable problem when the number of color models -- and thus trackable objects -- goes above five or six. In a real world scenario there would be significantly more than this.

Finally, the last problem with using color models to track is that a new object entering the scene does not have a color model to be tracked with. The problem becomes a catch-22: how to

track an object using a color model which you can only find by tracking the object and building the color model on the fly.

Form-based tracking methods don't suffer from the severe limitations of color based trackers because they are fundamentally position-based trackers by following locations of the head, torso and feet, for example. For that reason they are very useful.

1.3.3 Infrastructure Model and Thesis Layout

Our human tracking system is designed around what we call an "infrastructure model" of algorithm interaction. This architecture is designed to give system designers the flexibility and the basic underlying infrastructure to implement complex tracking and classification systems. At the core of the infrastructure is the image processing described in Chapter 2. The next layer of complexity is the position-based tracker which will be described in Chapter 3. Finally, the last layer we were able to complete is a simple color-model-based person identification system which works together with the other two layers. The color modeling is described in Chapter 4.

The layers of the infrastructure can all intercommunicate to influence each other. For example, a typical problem to be solved (described in more detail in Chapter 6) is that of occlusion. On the face of it, there is no mechanism to deal with two people meeting and merging into one region of movement. The position-based tracker is useless in this situation to recover the positions of the people. However, by communication among systems, the position tracker is able to first identify that there is a problem and allow the color modeling system to solve it. In this way, we can build a robust system capable of human tracking, identification and even gesture recognition in one place.

CHAPTER 2
IMAGE PROCESSING AND SEGMENTATION

2.1 Introduction

The image processing and segmentation algorithms are the first method by which we reduce the complexity of the problem. We wish to reduce all moving objects (whether noise or people we wish to track) to point objects which will be tracked by the track processor as described in Chapter 3. The image processing and segmentation should remove any spurious noise which we know to not be important. The segmentation algorithms should also assure that people moving through the scene are "complete" and not missing any section of their body. We also want the segmentation algorithms to aid the color modeling (described in Chapter 4) by providing the color data from the pixels of which a person object is composed. Figure 2-1 gives a broad overview of the image processing and segmentation data flow going from video frames to the centroids of objects which will be passed to the track processor described in Section 3.2.

2.2 Image Processing

2.2.1 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 colors in RGB (red-green-blue) colorspace. However, as noted by Zapata [12], we found that the HSV (hue-saturation-value) colorspace was more condu-

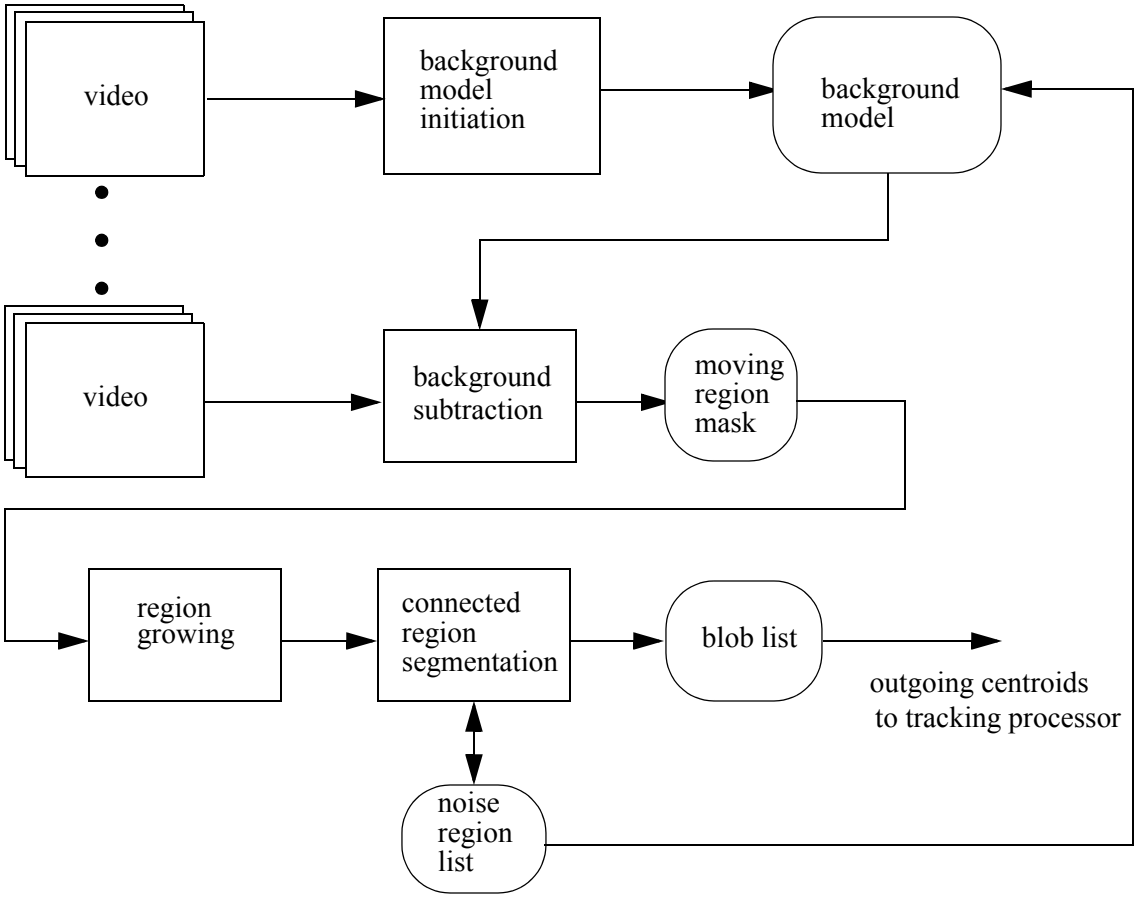


Figure 2-1: Image processing and segmentation overview

cive to reducing the effects of shadows. (See Appendix for more information about these two colorspaces and their effect on shadows.)

Mathematical subtraction of colors 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

$$r = [(H_b - H_f) > t_H] \wedge [(S_b - S_f) > t_S] \wedge [(V_b - V_f) > t_V] \quad (2\text{-}1)$$

where H_b is the hue component of the background pixel, 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. [2] 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 color in hsv colorspace 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, Σ) 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.

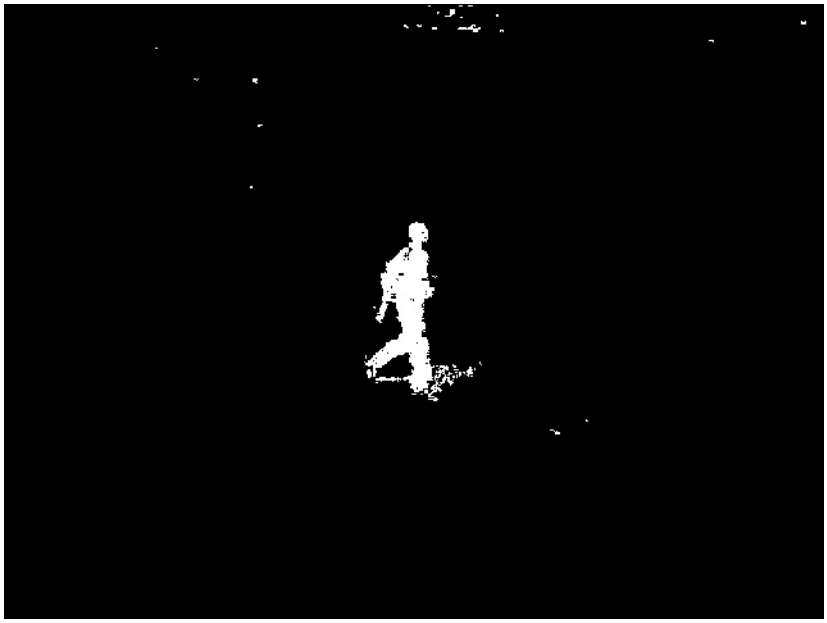


Figure 2-2: Moving region mask

2.3 Segmentation

2.3.1 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 result is a mask which can be used to extract the color information for any given blob as we will need to do later.

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 described in Chapter 3. 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 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 color. 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.

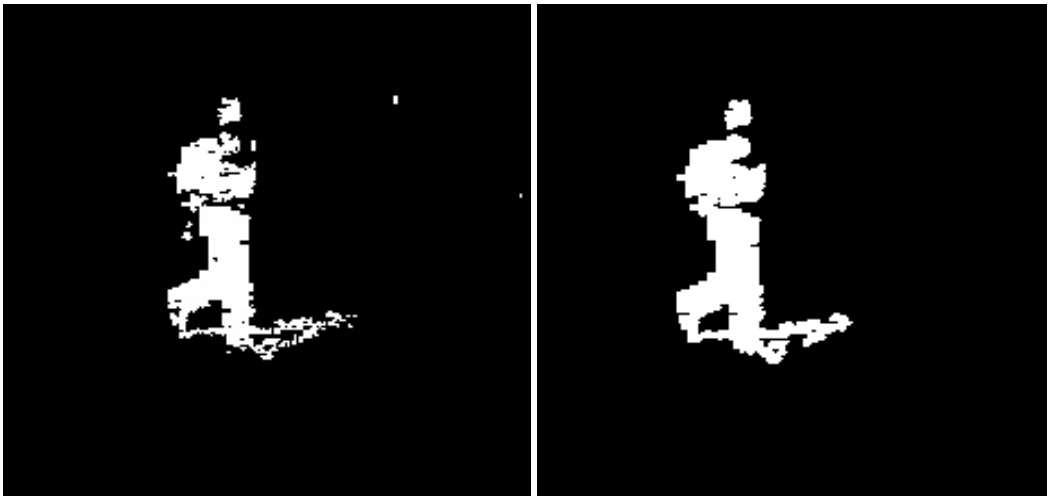


Figure 2-3: Disconnected moving region mask (left) and grown mask (right)

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 2-3 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 indiscriminantly 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. Neither of these problems are currently resolved but we have an approach to solving them in Section 6.2.

2.3.3 Background Model Feedback

Environmental lighting changes caused by clouds or sunset, for example, can cause significant problems if the background color model does not change over time. As shown in Figure 2-1, we feedback noisy pixels into the background color model. We classify regions of pixels as noisy in the connected region segmentation algorithm by thresholding the total number of pixels in the region. If it is below the threshold, the pixels in that region are automatically sent to the background model subroutine where the normal distributions are updated.

CHAPTER 3
TRACKING MOVING OBJECTS

3.1 Introduction

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.

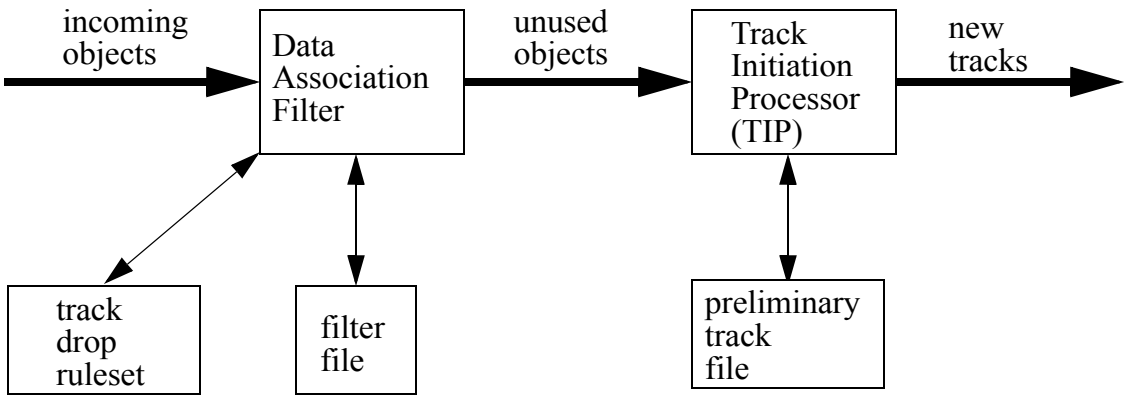


Figure 3-1: Basic operation of the tracking processor

3.2 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" [5] 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 over-lapping 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 tradeoff 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 object) 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.

In practice, the author experimentally chose to use five frames of data before declaring a track legitimate. (Using five frames also comes in handy in Section 3.3.1 where a filter needs to converge based on these past data points.) The distance discussed in the previous paragraph was chosen to be 5.0 pixels. (For future reference, all measurements are based on pixel units for ease unless otherwise stated.) These numbers were used for all the examples presented herein.

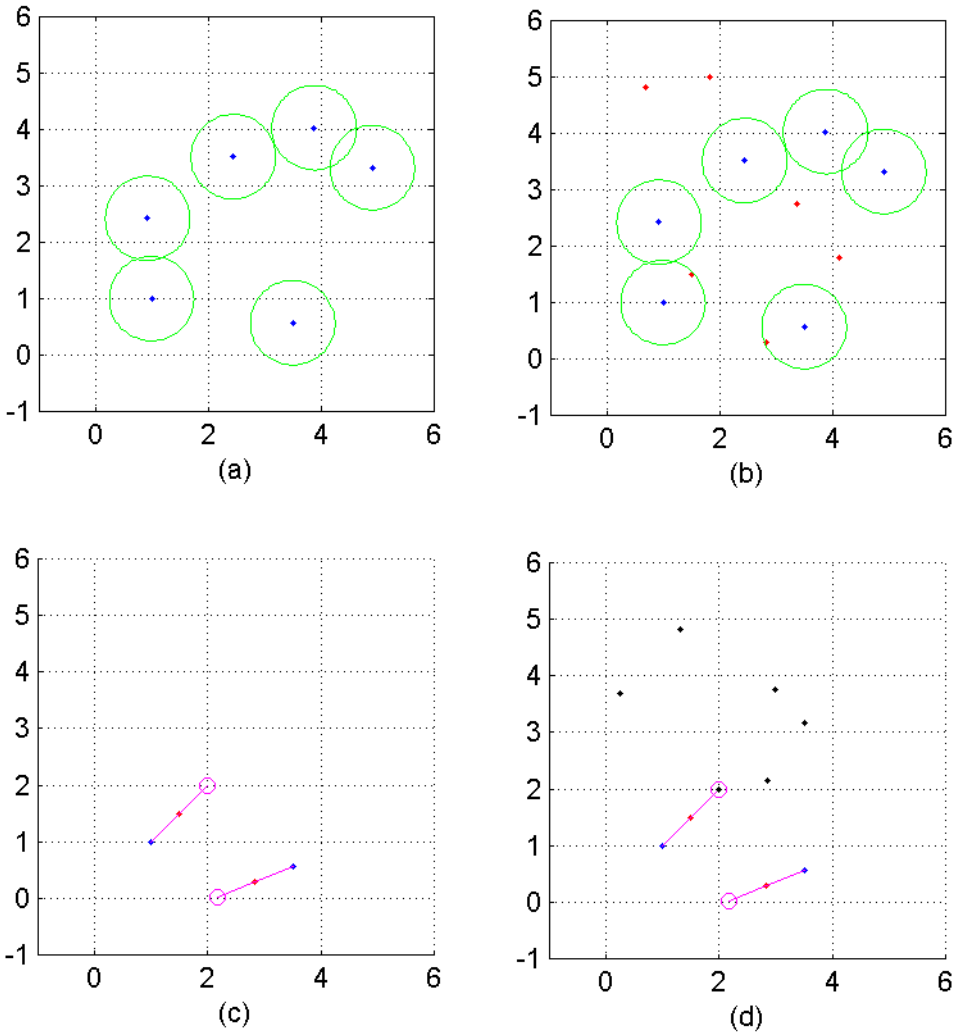


Figure 3-2: Track Initiation Processor (TIP) simulation

3.3 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.

3.3.1 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:

$$\hat{x}_k = \begin{bmatrix} x \\ y \\ \frac{dx}{dt} \\ \frac{dy}{dt} \end{bmatrix} \quad A = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3-1}$$

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 $\left(\frac{dx}{dt}, \frac{dy}{dt}\right)$. The state update matrix (A) contains the frame update rate of the camera (dt). We use the standard formulation for the linear Kalman filter given in [18]:

$$\hat{x}_k^- = A\hat{x}_{k-1} \tag{3-2}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{3-3}$$

$$K_k = P^-_k H^T (HP_k^- H^T + R)^{-1} \tag{3-4}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{3-5}$$

$$P_k = (I - K_k H)P_k^- \tag{3-6}$$

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.

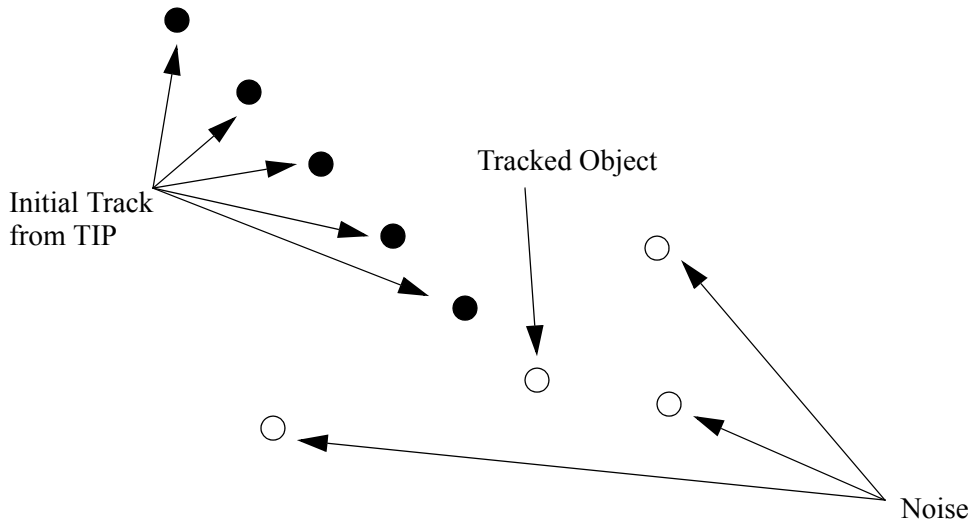


Figure 3-3: The data association problem

3.3.2 Nearest Neighbor 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 neighbor 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,

$P_{\bar{k}}$. 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.

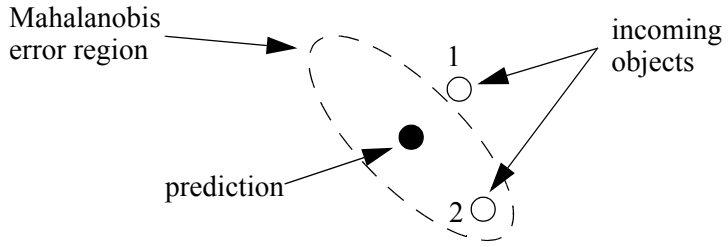


Figure 3-4: Mahalanobis distance versus Euclidean distance

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, $P_{\bar{k}}$, 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 assocated with the filter which gave the prediction. Therefore, we use the Mahalanobis distance to determine the nearest neighbor.

One problem with a simple nearest neighbor 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.

3.4 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. 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.

3.5 Conclusion

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.

Figure 3-5 shows a sequence of six frames of a video each five frames apart beginning in the upper left-hand corner and continuing left-to-right to the bottom. The video was captured at 30 frames per second. There are two people entering the scene walking next to one another. In the first image, they are bounded in blue to indicate that the tracker currently considers them to be noise. (All untracked movement regions are considered noise and are bounded by blue.) By the

third image, the person on the left is recognized as a target moving according to the assumptions given above. Therefore that person is bounded in a green box and assigned a filter identification number (zero) shown inside the box. It isn't until the fifth image that the other individual is also picked up as a trackable object. That person is assigned the next filter identification number (one). The two people are tracked in this way through the video.

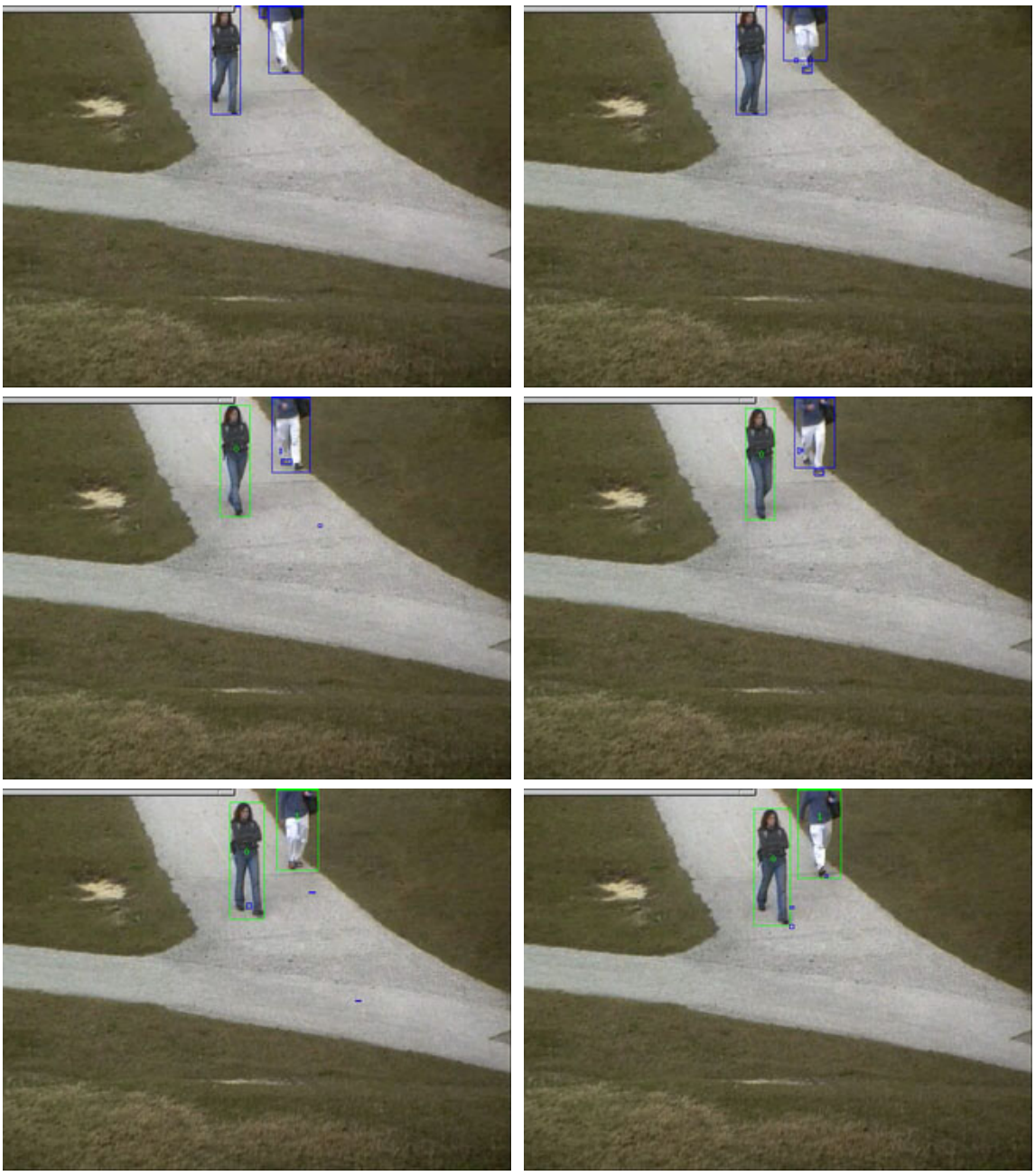


Figure 3-5: People tracking using the position-based tracking processor

CHAPTER 4
COLOR MODELING

4.1 Introduction

We chose to use color 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 color. 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 modeling the colors contained in the pixels of the person as segmented by the algorithms described in Chapter 2. The color model has to be able to be generated (or 'trained') quickly but must also provide a way to compare the color of incoming pixels to an already-generated color model for recognition purposes. The color model must also be practically usable in that each model generated will have to be stored in a database.

4.2 Gaussian Mixture Modeling

Figure 4-1a shows a typical person segmented from a video sequence by the image processing described in Chapter 2. Figure 4-1b shows a graph of each pixel in that image mapped in RGB (red-green-blue) colorspace. We need to be able to model those pixels in RGB colorspace 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 colors in colorspace) is statistically spread as k gaussians. Figure 4-1c shows the data from the segmented image modeled 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 colorspace. The rest of the data is modeled by the remaining ellipse (blue) which has a

greater spread because the data it models is much more spread than the data modeled by the green and red ellipses.

4.2.1 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 (4-1) and (4-2),

$$p(x|\phi_i) = p(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\mu_i)^T\Sigma^{-1}(\boldsymbol{x}-\mu_i)\right] \tag{4-1}$$

$$\phi_i = \{\mu_i, \Sigma_i\} \tag{4-2}$$

where μ_i and Σ_i give the mean and covariance for the ith component density, respectively.

The EM algorithm applied to the mixture-of-Gaussians problem produces three parameter update equations found in [19],

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

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

$$\overline{\Sigma_i} = \frac{\sum_{j=1}^{n} P(\omega_i|x_j, \Theta)(\boldsymbol{x}_j-\overline{\mu_i})(\boldsymbol{x}_j-\overline{\mu_i})^T}{\sum_{j=1}^{n} P(\omega_i|\boldsymbol{x}_j, \Theta)},\ i \in \{1, 2, ..., k\} \tag{4-5}$$

where

$$\Theta_i = \{\phi_i, P(\omega_i)\} \tag{4-6}$$

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

$$p(x|\Theta) = \sum_{i=1}^{k} p(\boldsymbol{x}|\phi_i)P(\omega_i) \tag{4-7}$$

where $P(\omega_i)$ gives the probability of the *i*th component density, $p(\boldsymbol{x}|\phi_i)$.

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

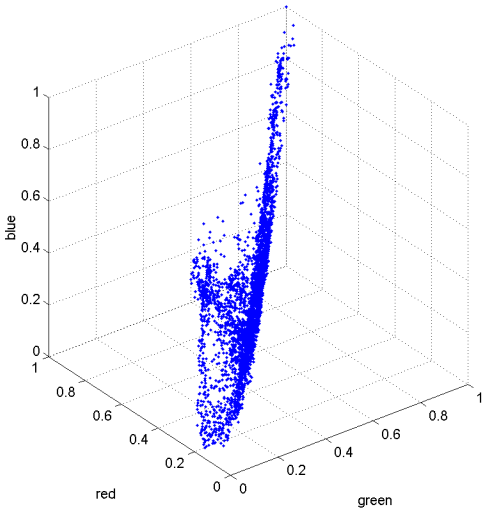
$$P(\omega_i|\boldsymbol{x}_j, \Theta) = \frac{p(\boldsymbol{x}_j|\phi_i)P(\omega_i)}{p(\boldsymbol{x}_j|\Theta)} \tag{4-8}$$

remembering that $\boldsymbol{x}_j$ is the set of data we wish to model.

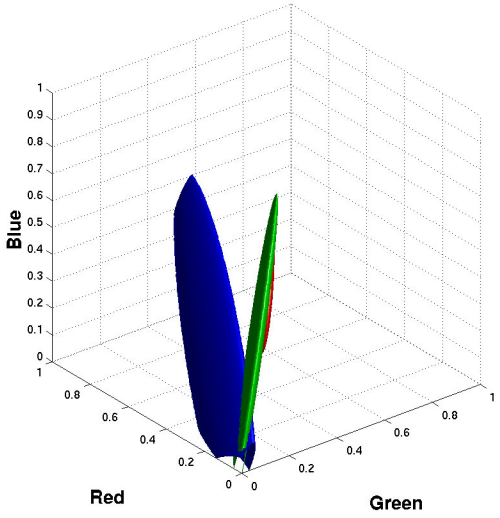
4.2.2 Training the Mixture Model

When a person is unrecognized by pre-existing mixture models found in the color model database described in Section 1.1, a new model needs to be trained. If we make the replacement suggested by Equation (4-8) into Equations (4-3), (4-4) and (4-5) we will have three EM update equations capable of being implemented in software. However, notice that although all three update equations share the expression $P(\omega_i|\boldsymbol{x}_j, \Theta)$, 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 4-1: Example of gaussian mixture modeling. a) Segmented person image from image processing. b) Colorspace diagram for pixels in person image. c) Mixture of three gaussians model for the color data.

From experimental data, a person image such as that in Figure 4-1 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.

To reduce the computing time of the EM algorithm so as to be able to use it in a real-time system, we chose to fix the number of iterations of the algorithm for each frame of data. Instead of converging the model on one set of color data from a tracked person, we chose to replace the data each frame. In this way we assure that any colors not showing in one frame will be modeled eventually. Assuming that the colors respresented in each frame to not change significantly, the fact that we do not converge the model on one set of data will not matter. The model will converge over a series of frames and thus a series of sets of data.

The three-Gaussian mixture model shown in Figure 4-1 was converged over a series of frames of the video shown in Figure 4-3.

$$\text{loglikelihood} = \sum_{i=1}^{n}\left[\log\left(\sum_{j=1}^{\kappa}\frac{1}{(2\pi)^{d/2}|\Sigma_j|^{1/2}}\exp\left[-\frac{1}{2}(\boldsymbol{x}_i-\mu_j)^T\Sigma_j^{-1}(\boldsymbol{x}_i-\mu_j)\right]\right)\right] \qquad (4\text{-}9)$$

4.2.3 Completing the Training Process

We need a method to determine when the color model is sufficiently trained so it can be placed in the database. A validation data set is needed to generate an unbiased evaluation of the model. We use the incoming color data from a new frame as the validation data which is tested against the model to determine its log-likelihood using Equation (4-9).

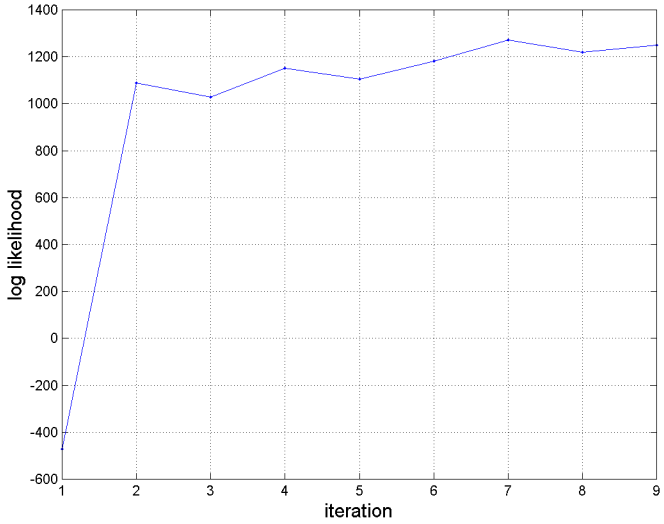


Figure 4-2: Convergence of the EM algorithm

If the log-likelihood plateaus over a series of frames as shown in Figure 4-2, we assume that the model has converged. It may be difficult to recognize a plateau because unlike using EM on one set of data where it is proven to improve each iteration (even if very insignificantly), changing the data each iteration means that there is a possibility it will not improve from one frame to the next.

4.2.4 Comparing Color Data to the Database Models

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 color data to the color 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 a log likelihood above that threshold, we create a new model for that person as described above.

4.2.5 Example

Figure 4-3 shows a typical set of output images from the algorithm incorporating tracking and color modeling. (Frames are non-consecutive from the top-left to the bottom-right.) The first image shows the track processor acquiring a track on the person. The person is now identified in

Figure 4-3: Person tracking and classification

two ways: a name and a number. The name (“ann” in this example) is related to the color model. That is, the color model for this person is named “ann”. The second number is again the filter number as described in Section 3.3. The second frame shows a problem with image processing causing the person to be classified as two objects. The tracker loses the person because of it. (The version of code which made this movie specifically did not incorporate the region growing algorithm described in Section 2.3.2 so that we would get this error.) The third frame shows that the person is again whole but not yet tracked. The fourth frame shows that once the track processor acquires the person again, the color data of that person was compared to the pre-existing models in the database and was correctly classified as “ann.” The final two frames demonstrate that it keeps the track and classification.

CHAPTER 5
RESULTS

5.1 Experimental Setup

The system we implemented was run on pre-recorded video sequences. The image processing, tracking and classification algorithms alone took a total of approximately 0.3 seconds per 640x480 frame of data on an Intel PIII-700 MHz system running the Linux operating system. (The EM algorithm ran one iteration per frame on the move shown in Figure 4-3.) However, our system was not designed for speed and was highly unoptimized. We think that the execution time could be reduced significantly by using special compilers to optimize the code. Many of the more time-consuming operations are highly parallelizable in that they don't rely on information before or after their execution in order to complete. To effectively take advantage of this we would need a compiler which utilized the SIMD (Single Instruction Multiple Data) instructions of modern processors. We could also run the algorithm on a smaller, subsampled image. Using 320x240 images instead of full 640x480, we could achieve an approximate four times speed improvement. If the speed increase scales linearly, we could expect to analyze 13 frames per second. Further speed improvements are discussed in Section 6.5.

5.2 Tracking

We have successfully demonstrated position-based tracking over multi-second sequences. Figure 3-5 shows a typical sequence using position-only tracking of two people. We have shown that the tracking system is robust against noise typical of that which cannot be removed using image processing techniques. Acquiring tracks on people entering a scene and dropping tracks when a person leaves the scene is successfully implemented.

The conclusion of the aforementioned video is shown in Figure 6-2 where the tracking fails due to occlusion. Occlusion is the most common failure mode of the algorithm and needs more rigorous treatment than the position-based tracker alone can provide. We discuss resolving this problem in Section 6.6.

5.3 Color Modeling

We have successfully demonstrated generating color models as tracking is occurring and using those color models to identify a person if tracking is lost then regained. Figure 4-3 shows this situation where tracking is lost but when its regained the person is successfully classified based on their color model.

5.4 Conclusions

In this thesis, we presented an approach to solving the human tracking problem non-invasively from video captured by standard cameras. We used two general methods to solve the problems of tracking and identification. First, we used a position-based tracking system to initiate, continue and end tracking of the person. Second, we use a color model of the tracked person to identify them as they enter and leave the field of view of the camera. We envision this whole system to be implemented over many cameras to afford surveillance over a significant area.

CHAPTER 6
FUTURE WORK

6.1 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.

The second movement region caused by the real background is stationary and for that reason the track initiation processor will determine it to be a trackable object because it follows all of three of our assumptions given in Section 3.2. Its velocity, however, is zero or near zero. Knowing this, we can make the assumption that all stationary objects are noise. Being noise, we can send

their pixels to the background model updating code. Eventually, the background model will be the real background and not our assumed one.

Most of this background-finding scheme is already in place. It has not yet been fully integrated, however, and some testing would need to be done to make it robust against too much noise as would occur when the background model is trained on very few images.

6.2 Efficient Region Growing and Shrinking

As described in Section 2.3.2, 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 color 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.

6.3 Camera Models 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. The central database (described in Section 1.3.3) would then use this information to generate tracks over wide areas.

6.4 Joint Probabilities

The infrastructure of this system is one of layered control. The lowest level control over the output is the position-based tracking system described in Chapter 3. Above that level is the classification system using color models described in Chapter 4. There is little or no overlap or data sharing between the two. A much better system would be to incorporate a color model into the tracking system. This hybrid system would work by classifying incoming unclassified objects as belonging to a particular track/person using both position and color. Such systems have been used previously and are known as joint probability data association systems. Using a system like this may improve tracking in noisy environments.

Another method of tracking (especially throuhg occlusion described in Section 6.6) is to track basic parts of the human form rather than simply the centroid of the person. Haritaoglu et al. [8] use this method successfully. They use a simple “cardboard” model of the tracked person. In this way, the tracking problem becomes more complex yet more constrained by the form of a person.

6.5 Speed Improvements

One simple method of improving how long it takes to analyze a single video frame is to use a significantly subsampled image to find motion regions of interest before looking to the whole frame for details. For example, instead of using a whole 640x480 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.

6.6 Occlusion Resolution

Occlusion -- when the view of a moving object becomes obstructed by any other object -- is a considerable problem in human tracking. Figure 6-1 shows a typical occlusion where the person goes behind a stationary signpost. As can be seen from the sequence (which starts at the upper-left and continues left-to-right down) we eventually lose tracking on this person when the vertical sign cuts the person's image in two. The software still recognizes that there is movement but it does not classify the movement as being the person we were tracking.

Occlusion needs to be resolved in two steps to be effective. The first step is to recognize (detect) that occlusion has occurred. The second is to resolve the occlusion or "see-through" the occlusion to retain the track on our person. Detection of occlusion is often the more difficult problem.

To resolve the occlusion of Figure 6-1, it may be possible to detect that the track was dropped prematurely -- meaning not due to the edge of the screen -- and use the color model we converged over the previous frames to ascertain if the untracked regions (bounded by blue) are probably part of the person we just lost track of. That would be one approach. Another approach to resolving occlusion behind stationary objects is to not resolve it but rather wait for the person to emerge. When the person emerges, the tracker will re-acquire. The missing data from when the person was behind the object could be interpolated.

Figure 6-2 shows another occlusion where two people merge because they are touching and thus seen as one motion region. Resolving this occlusion involves first detecting that the two people (marked '0' and '1' in the images) merged into the one region. Second, the occlusion can be resolved using a technique mentioned in Section 1.3.2 by Khan, et al. [2] and Nechyba [6]. Unlike trying to resolve a multiple-person region using only color information from an otherwise unlim-

ited number of color models as they tried, we know which two persons occluded and which color models to use to resolve the occlusion with color.

Figure 6-1: Occlusion

Figure 6-2: Two-person occlusion

APPENDIX
HSV COLORSPACE

During the image processing described in Chapter 2, the background color model and the incoming (foreground) image is converted from the standard RGB (red-green-blue) colorspace to the HSV (hue-saturation-value) colorspace. The HSV colorspace is more commonly used for background subtraction because it has some useful properties such as reducing the effect of shadows on the subtraction process.

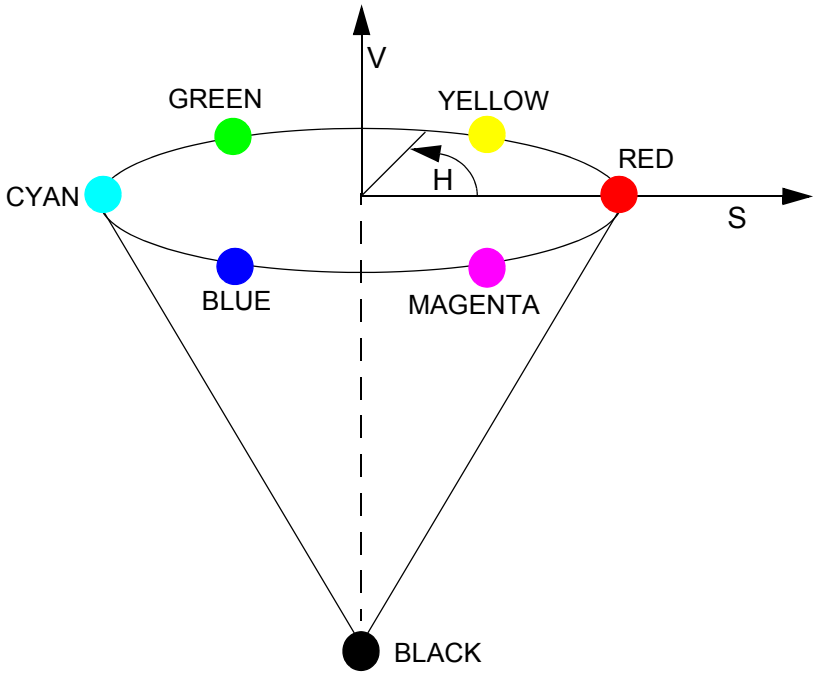


Figure A-1: HSV color cone

The HSV color cone is shown in Figure 1. The hue (H) is given by an angular coordinate around the central axis and gives the "tint" of the color. Note that complementary colors are 180° - out of phase from one another. Saturation (S) is the distance from the central axis of the cone to the outside. It gives the "brightness" of a color. Value (V) is the final component and measures the

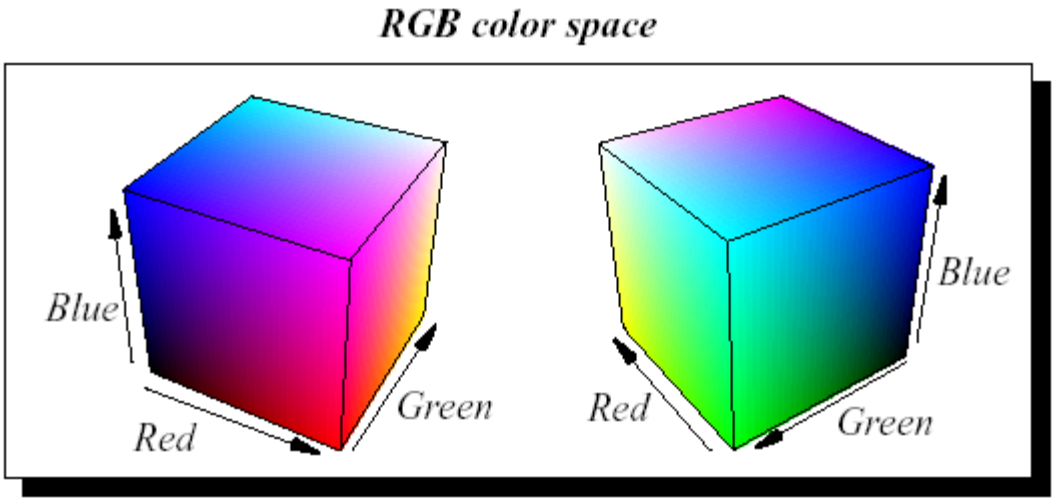


Figure A-2: RGB color cube

amount of "blackness" of the color with the bottom of the code being absolute blackness. By arranging the colors in this space, like shades of a color are "close" to one another. Changes in lighting conditions caused by shadows will change the value (V) of a color but the hue (H) tends to remain less effected. We can exploit this lighting independence of hue to reduce the effects of shadows.

The RGB color cube is shown in Figure 2 as a comparison. Similar shades of a given color are not near one another and may not even be continuous. The RGB colorspace has no channel (R-G-B) that is independent or at least somewhat independent of changes in lighting. All channels are effected to differing degrees. It becomes difficult to alleviate the effects of shadows in this color space.

REFERENCES

[1] G. Rigoll, S. Eickeler and S. Muller, “Person Tracking in Real-World Scenarios Using Statistical Methods,” *Proc*. *IEEE Int. Conf. on Automatic Face and Gesture Recognition*, pp. 342-7, 2000.

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

[3] 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.

[4] 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.

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

[6] M. Nechyba, “Multi-Car Resolution,” Unpublished work, Department of Electrical and Computer Engineering, University of Florida, 2000.

[7] I. Haritaoglu, D.Harwood, and L.Davis, "Who, When, Where, What: A Real Time System for Detecting and Tracking people," *Proceedings of the Third Face and Gesture Recognition Conference*, pp. 222-227, 1998.

[8] I. Haritaoglu, D. Harwood and L. Davis, “W4S: A Real-time System for Detection and Tracking People,” European Conference on Computer Vision 1998.

[9] S. Ju, M. Black, and Y. Yacoob. “Cardboard People: A Parameterized Model of Articulated Image Motion,” Submitted to the Second International Conference on Automatic Face and Gesture Recognition, 1996.

[10] J. Yang, W. Lu, and A. Waibel, (1997), "Skin-Color Modeling and Adaptation," *Proceedings of ACCV'98* (Technical Report CMU-CS-97-146, School of Computer Science, Carnegie Mellon University, 1997).

[11] 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.

[12] 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.

[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] S. Nichols, “Improvement of the Camera Calibration Through the Use of Machine Learning Techniques,” Master’s Thesis, Department of Electrical and Computer Engineering, University of Florida, 2001.

BIOGRAPHICAL SKETCH

Patrick O'Malley was born in Evanston, Illinois, in 1977. He grew up in Pullman, Washington, moved to Miami, Florida, and began attending the University of Florida in 1996. He earned a Bachelor of Science degree in computer engineering from that university in December 2000. He has worked as a research assistant in the Machine Intelligence Lab at the University of Florida, specializing in robotics, machine learning and intelligent systems while pursuing a Master of Science degree in electrical engineering.