40

Event Recognition in Scenes Involving People and Cars

by

Christopher H. Tserng

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Computer Science and Engineering

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 22, 1998

Author_____
Department of Electrical Engineering and Computer Science
May 22, 1998

Certified by_____
Eric Grimson
Thesis Supervisor

Accepted by_____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Event Recognition in Scenes Involving People and Cars
by
Christopher H. Tserng

Submitted to the
Department of Electrical Engineering and Computer Science

May 22, 1998

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Computer Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

This thesis focuses on developing the ability to recognize events between cars and people. It builds on earlier work in surveillance by using the basic techniques of the existing Texas Instruments' Autonomous Video Surveillance system. Basic image differencing is used to track cars and people and a set of heuristics are used to communicate high-level information about interactions between these objects. Additional image differencing is also done to track cars and people when they overlap each other. Furthermore, many methods are employed to solve problems associated with this additional differencing. The system is designed to operate under ideal outdoor conditions (no shadows, reflections, etc.) in which the camera only has a lateral view of cars. Its capabilities include the ability to detect cars and the ability to detect when people enter or exit cars.

Thesis Supervisor: Eric Grimson
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

I would like to thank God for always watching over me and helping me to get this thesis accomplished. He gave me the strength and the wisdom to accomplish this task. Also, He was very gracious to me in allowing me to finish this thesis on time. "And my God shall supply all your need according to His riches in glory by Christ Jesus" (Philippians 4:19)

I would also like to thank Bruce Flinchbaugh, the manager of the Vision Systems Branch at Texas Instruments, for his guidance and direction in helping me find and put together this thesis. He was also very diligent in monitoring my progress on my thesis when I returned to M.I.T. Additionally, he's been a great manager the two years I worked under him at TI, always providing interesting projects for me to work on. I also want to give thanks to Tom Olson, my thesis supervisor at Texas Instruments, who provided valuable feedback on my thesis drafts, and was always available for questions. Thanks also goes to Frank Brill, a coworker at TI, who answered many of my questions on the AVS system.

I want to thank Eric Grimson for being my M.I.T. thesis supervisor despite his busy work load. Also, I'm thankful for his comments on my thesis as well as his flexibility in giving me a lot of freedom in writing my thesis.

Finally, I want to thank my family for their financial and emotional support during the writing of this thesis as well as throughout my whole M.I.T. career.

Table of Contents

1 Introduction 6
1.1 Terminology 7
1.2 Overview 8
2 The AVS System 10
2.1 Hardware 10
2.2 Software 10
2.2.1 Detecting Objects 10
2.2.2 Tracking Objects 11
2.2.3 Analyzing the Motion Graph 12
2.3 Limitations 12
2.3.1 Lighting Conditions 13
2.3.2 Dynamic Environment 13
2.3.3 Different Objects/Events 13
2.3.4 Scope 14
3 Previous Work 15
3.1 People Tracking 15
3.2 Car Tracking 16
3.3 Event Recognition 16
4 Basic Approach 18
4.1 Assumptions 18
4.2 Car Detection 19
4.3 The Old Technique 19
4.4 The New Technique 20
4.4.1 Detecting Overlapping Objects 20
4.4.2 Basic Method 22
5 Object Integrity 29
5.1 Car Resting 29
5.2 Car Moving 32
6 Robustness 35
6.1 Noise Objects 36
6.2 Non-Occluding Moving False Objects 38
6.3 Occluding False Moving Objects 40
7 Results 43
7.1 Software Sequences 43
7.1.1 Simple Sequence 43
7.1.2 Pickup Sequence 44
7.1.3 Drop Off Sequence 44
7.2 Staged Videotape Sequences 45
7.2.1 Basic Sequence 45
7.2.2 Object Integrity Sequence 45
7.3 Real Videotape Sequences 46
7.3.1 Dark Day 47
7.3.2 Cloudy Day 47

7.3.3 Cloudy Day - Extended Time 47
8 Conclusion 48
Bibliography 50

Chapter 1

Introduction

As video cameras have become increasingly cheaper to produce and purchase, their functionality has evolved beyond their original intent to merely capture sequences of images. Smart video cameras of the future will be able to provide intelligent data about what objects moved and what events occurred within their field of view. The new functionality of smart cameras provides many new applications.

One increasingly valuable application for smart cameras is intelligent surveillance. Typically surveillance cameras record hundreds of hours of video footage onto video tape which must be monitored by people in order to tell if something unusual occurred. Most of the video data is useless because nothing interesting is occurring. However, with smart cameras, people can be alerted to specific events which the smart cameras detect. As smart cameras are developed, software algorithms will need to be developed to implement the "smart" monitoring functions. These functions can be low level such as a system that assigns different alarm levels to moving objects in different areas of a scene [3] or they can be more high level in detecting specific events.

Texas Instruments (TI) has developed an Autonomous Video Surveillance (AVS) system that implements object detection and event recognition. This system implements several "smart" monitoring algorithms to analyze a video stream and extract intelligent data in real time. Originally, the system was designed to monitor indoor scenes.

This thesis describes the extensions to the existing AVS system to allow the recognition of outdoor events; specifically, the interaction of people with cars. These new capabilities would allow smart security cameras of the future to monitor outdoor parking lots and driveways and intelligently report when unusual events occur. For example, a smart

camera could signal an alarm if a person exits a car, deposits an object near the building, reenters the car, and drives away. Such interactions between people and cars are currently not recognized by the existing system.

1.1 Terminology

In order to discuss my approach, some common terms need to be defined for clarity:

- An *image* is a picture consisting of an array of pixels.
- A *video image* is one image out of the input image stream.
- A *reference image* is an image of a scene that current video images are usually differenced with.
- A *background image* is an image of the scene with only background objects in it. Background objects are objects which are not expected to move in a scene that is being monitored. The background image is usually used as the reference image.
- A *difference image* is a binary image which is the result of thresholding the absolute difference of two video images (usually the current video image with the reference image). Thresholding converts pixels whose values are above the threshold to a foreground value (typically 1) and pixels whose values are below the threshold to a background value (typically 0); thus, generating a binary image.
- *Blobs* are groups of connected pixels in a difference image which represent the change regions between the two video images that were differenced. They usually correspond to objects in the foreground of a scene.
- An *object* is an abstract entity which represents a real-world object. It also has *blobs* associated with it which correspond to the real-world object.
- A *frame* is an abstract entity which consists of a collection of objects and represents a

single video image.

- Objects between frames which correspond to each other are linked together and form a *motion graph.*

1.2 Overview

The rest of this thesis will discuss the development of event recognition of interactions between people and cars. It will show a system that can track people and cars and determine when people enter or exit resting cars. Chapter 2 will discuss the existing system and its limitations. Chapter 3 will briefly cover the work done in the areas of people tracking, car tracking, and event recognition. Chapter 4 will explain the extensions that allow the new system to recognize people entering and exiting cars. The extensions involve a new method which uses additional image differencing to track a person and a car separately even when they overlap. This new method, however, brings up other issues involving object integrity and robustness. Therefore, possible solutions to these problems are explored in Chapters 5 and 6. Finally, Chapter 7 will report the results of tests on the new system and Chapter 8 will sum up the findings of this thesis.

Figure 1.1 is an example of the type of events which the system can detect. The following sequence of images shows a car driving into a driveway, a person exiting the car and depositing a briefcase, and reentering the car and driving away:

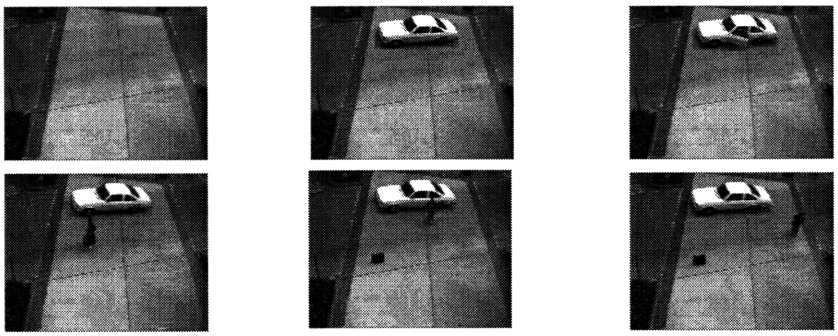


Figure 1.1: Example Sequence

The system would track both the car and the person separately and detect the person entering and exiting the car. Some of the output of the system is shown below in figure 1.2. The bounding boxes around the person and the car show that the system has identified them as separate objects (the bounding box identifying an object over the door region of the car will be explained later in Chapter 6).

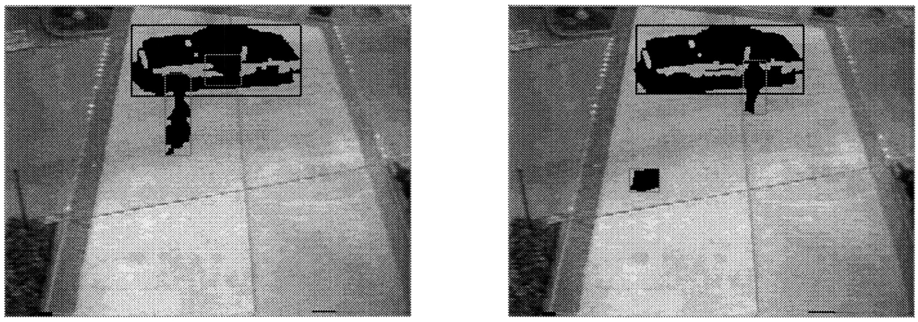


Figure 1.2: System Output

Chapter 2

The AVS System

In order to discuss the extensions to the AVS system, the basic system needs to be described. This chapter will describe the existing system and its limitations in outdoor settings. These limitations will reveal the motivation behind the expansion of the system.

2.1 Hardware

No special hardware is used for the AVS system. It runs on a 133 MHz Pentium Processor running Windows 95. Video streams are captured using a Matrox Meteor framegrabber. Videotaped sequences were taken from a camcorder. Future versions of AVS will probably run on the hardware of a "smart camera". This smart camera is basically a mini-computer which is of the form factor of a camera. It would basically have an on-board processor or DSP chip that runs the AVS software.

2.2 Software

The software of the AVS system consists of algorithms designed to analyze the input video stream. The main algorithm goes through three steps in every image of a video sequence to recognize events. The three steps are: detecting objects, tracking objects, and analyzing the motion graph [10].

2.2.1 Detecting Objects

The first step used in recognizing events is finding interesting objects in the scene. This task is accomplished using a 2D change detection technique that is commonly used in image analysis. Many other methods may be used which are described later in Chapter 3. In this method, a background image of the scene to be monitored is captured. This background image ideally contains only background objects (i.e. objects which are stationary and deemed unimportant). It is used as the system's reference image. To detect objects in

the scene at a given moment, the system first takes the absolute difference of the pixel values of the current video image and the reference image. This image is then thresholded to give a difference image of motion blobs (i.e. regions which differ significantly from the background image). Then, heuristics are used to group these blobs into distinct objects. These heuristics tend to group blobs which are near each other together, preferably in the vertical direction, since the main object its looking for is people. These objects are placed in a frame. The objects in their frame form an abstract representation of the video image. The goal of this step is to locate all the important objects in a given video image.

Figure 2.1 shows an example of the process of detecting an object. It shows a reference image, current video image, and the difference image it generates. It also shows the frame with the objects in it which correspond to the blobs in the difference image. The dotted circles show the blobs which were grouped together. The dotted arrows indicate that the objects are associated with blobs in the difference image.

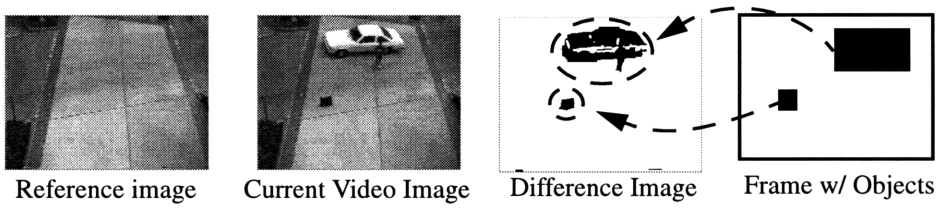


Figure 2.1: Detecting Objects Example

2.2.2 Tracking Objects

Once objects are detected in a video image, the next step is to track each object through the video sequence. This task is done by linking objects in the previous frame to their corresponding objects in the current frame. Correspondence is established by matching objects with their nearest neighbors. This matching is done by comparing an object's predicted centroid in the previous frame with all the objects' centroids in the current frame and vice versa. If two objects match each other than a strong correspondence is established, otherwise only a weak correspondence is established. Weak and strong correspon-

dences are used to recognize different events. Centroids are predicted using velocity information which is stored with each object. The path of links which follows a given object through successive frames is called an object's *track*. This track is maintained by having each object in every frame maintain a link to some object in the previous frame, and keeping all the frames around until the object leaves the scene. The objects and their tracks create a directed graph which represents the history of the motion of the objects in a video sequence. This directed graph is called a motion graph. The goal of this step is to create a motion graph for use by the next step in event recognition.

Figure 2.2 shows how two objects are tracked through four frames, creating a motion graph of the tracks of all the objects.

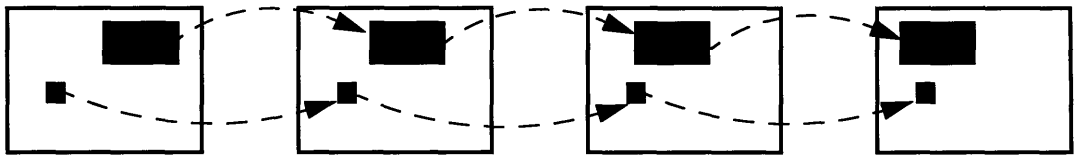


Figure 2.2: Tracking Objects Example

2.2.3 Analyzing the Motion Graph

Finally, to recognize events, the system analyzes the motion graph. For example, if a track for an object appears for the first time, an enter event is recognized. Similarly, if an object's track splits into two tracks and one track becomes stationary, a deposit event is recognized. The vocabulary of events that the system currently recognizes includes the following: enter, exit, rest, move, deposit, and remove.

2.3 Limitations

The events that AVS currently recognizes are the ones that are most common in an office environment where most interactions are between people and smaller stationary objects. However, the system is limited in that it is specifically geared towards an indoor environment with controlled conditions. It is not able to handle robustly some conditions

outdoors. There are several issues which make the system function poorly in an outdoor environment.

2.3.1 Lighting Conditions

One such issue that causes problems with AVS outdoors is the problem of lighting changes. In an indoor environment, one can have fixed lighting conditions. However, lighting is quite dynamic outside due to passing clouds and the changing position of the sun. These conditions cause classic problems such as shadows and reflections. Since AVS is based on image differencing against a single background image, lighting changes cause false objects to be detected. Furthermore, shadows and reflections also can cause objects to be a different shape than they really are. This may cause problems in object recognition. Work by Stauffer [5] and Friedman [4] use different methods that help solve the issue of lighting changes. These methods will be discussed later (Chapter 3).

2.3.2 Dynamic Environment

Another problem is the dynamic environment outside. Indoors, most objects are rigid, stationary and unchanging. With trees and grass and other non-rigid objects outside, image analysis becomes more difficult. Moving leaves and grass causes many false objects to appear using the current AVS system.

2.3.3 Different Objects/Events

Finally, the types of objects one deals with outdoors vary with those found indoors. In an indoor environment, humans are usually the only foreground objects which move. Everything else is usually rigid and stationary such as briefcases, notebooks, chairs, computers, etc. However, outdoors, humans are not usually the largest thing and are not the only things moving. Cars move around just as often as humans do in an outside environment. Cars also complicate the issue because people can get inside and outside of cars. The event vocabulary for the system outside would need to be expanded to make accommodations for these factors.

2.3.4 Scope

Because there are many limitations of the system outdoors, this thesis focuses only on one of them. It begins to handle the issue of different objects and events outdoors. This problem area was chosen because it was more desirable to add functionality to the system than to attempt to correct the classic outdoor problems of lighting and dynamic conditions. Handling these classic problems would have allowed the system to function robustly in any environment. However, it would not add any features to the system. It was decided that adding functionality was preferred over increasing the robustness of the system. Therefore, the limitations of lighting and dynamic conditions is ignored and the thesis focuses on increasing the event vocabulary of the system outdoors; specifically, the events involving the interaction of people and cars.

Chapter 3

Previous Work

Some people have done research analyzing moving cars or moving people. However, there really does not seem to be much research in the area of interactions between people and cars. This chapter will discuss work that has been done in people tracking, car tracking, and event recognition. Many of the techniques described here are alternate methods that could have been implemented in augmenting the AVS system.

3.1 People Tracking

Many systems have been developed that use various methods to track people. Often, these methods vary from those used in the AVS system. Q. Cai of the University of Texas differences consecutive frames rather than current frames with a reference frame [1]. This approach handles moving background objects nicely but will lose objects that come to a rest. It would be an approach that might be helpful outdoors in reducing the effects of subtle lighting changes. Stauffer represents each pixel in an image as a set of Gaussian distributions of the intensities likely to be seen at that point. Therefore, pixels in the current image that deviate from these distributions are labelled as foreground pixels [5]. This method works nicely in the dynamic outdoor environment, handling shadows and moving branches and leaves fairly well.

Instead of using size and aspect ratio to label people, as in the AVS system, other systems include models of what a person is supposed to look like. Pfinder [12] at the MIT Media Lab uses statistical models of shape and color to extract a persons heads, hands, and feet in an image. Likewise, the W^4 system at the University of Maryland uses a Cardboard Model of a person which attempts to model the relative positions and sizes of body

parts to identify a person's head, torso, feet, and hands [7]. Using more precise models of a person might help in modeling interactions of people with cars.

3.2 Car Tracking

Much of the research done in this area relates to monitoring highway traffic. The main concern here is segmenting cars from the road. Friedman and Stuart present a method of background subtraction where current image frames are differenced with a time-averaged background image [4]. This method also might be helpful in reducing the effects of lighting changes outside. Additionally, an advantage of this approach would be that it handles shadows nicely. Stauffer uses aspect ratio and world size information to track cars [5]. Another approach, used by Koller, employs motion and contour estimation along with an occlusion reasoning step to reliably track cars [9].

3.3 Event Recognition

As stated earlier, not much has been done with recognizing events involving people and cars. Ismail Haritaoglu of the University of Maryland implemented a simple object detector and tracker that uses most of the same methods as the AVS system. The event of a person parking a car and exiting car is given as an example sequence for this system. In this example, his system is able to recognize that one object (the car) brought the other object (the passenger) into the scene [6]. However, the system cannot tell that a car brought a person into the scene. It only deals with low level "objects". This example is similar to the types of events that the modified AVS system is supposed to recognize. Stauffer employs an action recognition system based mainly on motion recognition. Common tracks of moving objects are clustered together, and any track which deviates from these clusters are marked as unusual events [5]. Johnson and Hogg construct probability

density functions to describe object trajectories in a monitored scene, and use them to detect unusual events which deviate from normal trajectories [8].

Chapter 4

Basic Approach

One can take many approaches to detecting people entering and exiting a car. This chapter will analyze the assumptions given in tackling the problem. It will also describe the method using for detecting cars. Finally, it will describe the two approaches that were taken. The first approach utilized many of the features of the existing AVS system to recognize events, while the second approach created a new technique for recognizing events.

4.1 Assumptions

Given the focus of the research on people and car interactions, many assumptions need to be made to eliminate the other outdoor variables that were described previously (Section 2.3). These assumptions allow the modified AVS system to handle people and car interactions without worrying about other issues. These assumptions are:

1. Cloudy day (no shadows, reflections, dramatic lighting changes)

2. Static environment (no moving trees, moving grass, etc.)

Furthermore, additional assumptions are necessary to simplify the task of handling people and car interactions. These assumptions are:

1. Camera only sees lateral views of cars

2. No cars in the scene overlap one another

3. Overlapping people and cars cannot be moving simultaneously

The first assumption exists to provide a uniform view of cars so that the various orientations of cars do not need to be taken into consideration. The second assumption exists because it is difficult to deal with occluding objects of the same type. Finally, the third assumption exists because it is difficult to keep track of two moving objects which overlap

one another. Under all these assumptions, the modified AVS system is able to recognize two new events: people entering cars, and people exiting cars. The remainder of this chapter will explore the methods used to analyze and label these new events.

4.2 Car Detection

The first thing that was done to expand the event recognizing capability of the current system was to give the system the capability to distinguish between people and cars. Many methods were considered to detect cars. One method would be to locate various features of the car similar to the way Pfinder [12] locates body parts of people. However, given the assumption that cars would be laterally viewed, all cars would usually have the same physical characteristics. Hence, the system detects objects that are cars by using an object's size and its aspect ratio. This method is the same method used to detect objects that are people in the existing system. If an object is greater than 3.5 feet high and greater than 11.0 feet wide, and has an aspect ratio (height/width) less than 0.7, then the object is labelled as a car. The size of an object in feet is obtainable because the AVS system has an image coordinate to world coordinate mapping. This mapping is based on quadrilaterals that map horizontal planes in an image to horizontal areas in a floor map [10]. Once the system has an idea of which objects are cars, it can begin to analyze the motion graph to recognize people and car interactions. It is assumed that all moving objects that are not cars are people for simplicity's sake.

4.3 The Old Technique

The first approach to detecting people entering and exiting cars utilized the existing event recognition technique of AVS. For example, in the old system, if an object's track split into two, and one object is stationary, then the system calls this event a deposit. Now, in order to detect a person exiting a car, the "deposit" recognition was modified to check if

the stationary object was a car. Similarly, if two objects merged into one, and one of the objects was a car, then the system would report a person entering a car. These techniques followed the same approach used to detect deposit and removal events in the old system.

However, this approach was abandoned because it did not perform very well. This technique's lack of performance was caused mainly because it made the false assumption that people and car objects merging and separating correspond to people entering and exiting cars. This assumption caused the system to frequently produce both false positives and false negatives. For example, if a person were to walk past the front of a car, this technique would detect a person entering a car and a person exiting a car because it saw a person blob merging and then splitting from a car blob. Furthermore, if a person exited a car, deposited an object, and re-entered the car, all within a close distance to the car, the system would never segment the person and a car because it is hard to segment objects which are close together; thus, no events would be detected. Clearly, another method was needed to detect people entering and exiting cars.

4.4 The New Technique

The second approach to detecting people entering and exiting cars utilized a new technique which involves multiple reference images. This technique is based on a method of detecting objects even when they are overlapping the car. This capability would allow the system to actually detect the moment a person entered or exited the car rather than when they overlapped the car, as in the previous method.

4.4.1 Detecting Overlapping Objects

One way to detect objects overlapping a car is to maintain a *reference car image*. A reference car image is a subimage which contains an unoccluded picture of the car in the image. Then, *current car images*, subregions of current video images which correspond to the location of the car, would be differenced with that reference car image to create an

overlap car image. The blobs in this image would represent overlapping objects. This method sounds reasonable. However, a problem arises because many objects do not just overlap the car, but also overlap the normal background. For example, a person standing in front of a car may only have his midsection overlapping the car. This occurrence causes problems because the system now has to group corresponding blobs in the overlap car image with the blobs in the regular *foreground difference image*. The foreground difference image is calculated by differencing the current video image with the reference image.

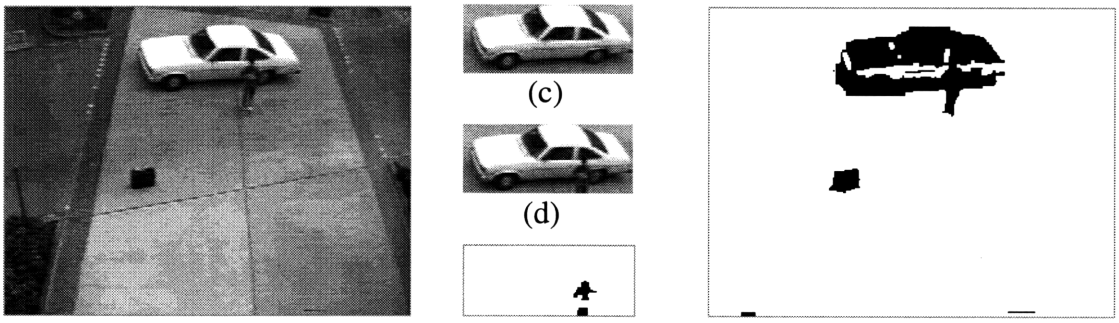


Figure 4.1: (a) Current Video Image. (b) Foreground Difference Image. (c) Reference Car Image. (d) Current Car image. (e) Overlap Car image

In figure 4.1, this technique is demonstrated. The current car image (d) is differenced with the reference car image (c) to obtain blobs representing overlapping objects (e). However, looking at the foreground difference image (b), one can see that a correspondence would need to be made between the feet of the person in the foreground difference image and the rest of the person in the overlap car image. This correspondence could not easily be done.

Another method which solves this correspondence problem, involves actually updating the reference image with the car reference image. Thus, the new foreground difference images would contain all objects excluding the car (Figure 4.2) However, in this method, the problem is that the car is no longer a blob in the foreground difference image. For this method to work, a blob representing the car needs to be maintained as well. This problem

is not as difficult as the correspondence problem and so this was the method that was implemented.

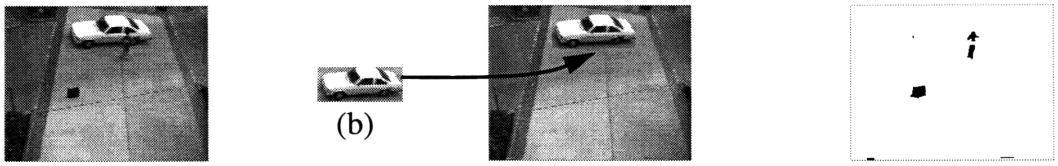


Figure 4.2: (a) Current Video Image. (b) Car Reference Image. (c) Updated Reference Image. (d) Foreground Difference Image

To use this method of detecting overlapping objects, two issues need to be dealt with:

1. When to difference current video images with updated reference images.

2. How to obtain a reference car image.

For simplicity's sake, it was decided to difference current video images with updated reference images only when a car comes to a rest. This choice was made because we were interested in events in which people enter and exit cars. These events would normally occur when a car is resting. Furthermore, this choice makes it easy to obtain a car reference image. A car reference image is just the initial image of the resting car. The consequences of these choices as well as alternative choices will be discussed later in the chapter 6 (Robustness).

Using this technique, it is easy to detect when people enter and exit a car. If an object disappears when it is overlapped with a car, it probably entered the car. Similarly, if an object appears overlapped with a car, it probably exited the car. The next section will describe the basic algorithm that implements this new technique.

4.4.2 Basic Method

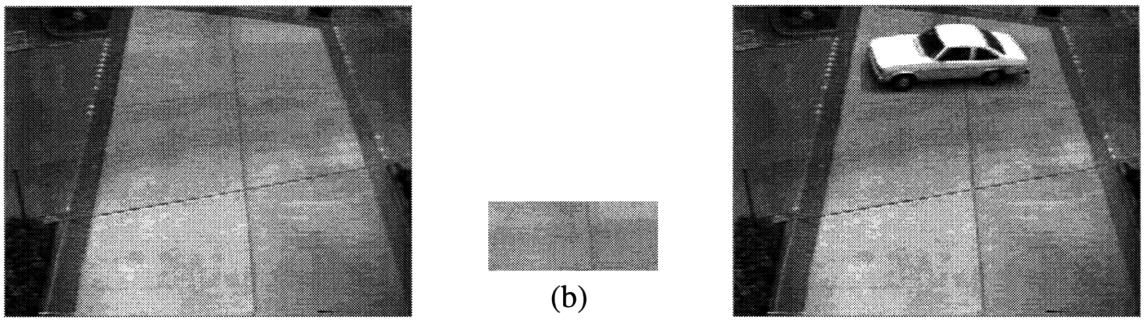
When a car comes to rest, the following sequence of events occurs:

1. The car object is removed from its frame and stored

2. The car image is merged with the background image creating an updated reference

image containing the car. (This car image is the reference car image)

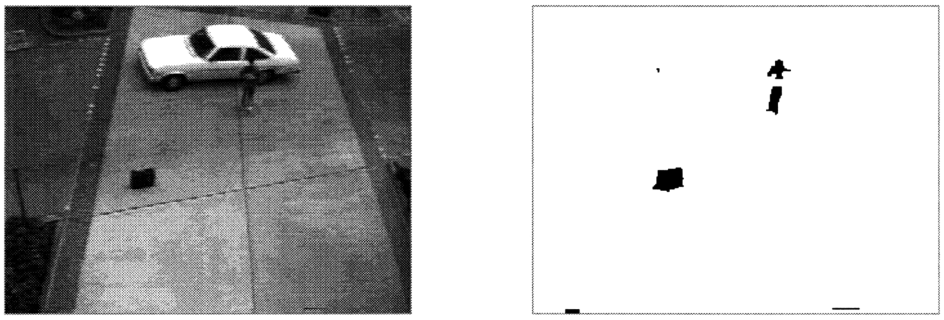
3. The *car background image*, the region of the original background image that is replaced by the car image, is stored



(a) (b) (c)

Figure 4.3: (a) Background image. (b) Car background image. (c) Updated reference image

For each successive frame, two difference images are generated. One difference image, the *foreground difference image*, is calculated by differencing the current video image with the updated reference image. The foreground difference image will contain all the blobs that represent objects other than the car, including ones that overlap the car.



(a) (b)

Figure 4.4: (a) Current Video Image. (b) Foreground difference image

The other difference image, the *car difference image*, is calculated by differencing only the *current car image*, the region of the current video image which corresponds to the location of the car, with the car background image. The car difference image contains

blobs that represent the resting car.

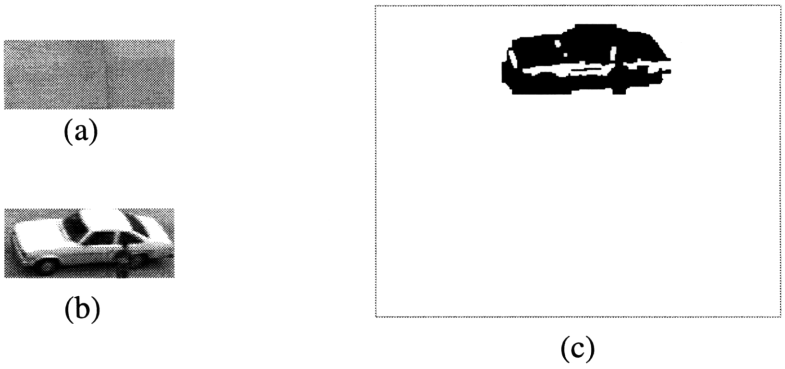


Figure 4.5: (a) Car background image. (b) Current car image. (c) Car difference image

To create the motion graph, the blobs in the foreground difference image are grouped into objects using the normal grouping heuristics and placed in the current frame. On the other hand, every blob in the car difference image necessarily represents the car, so they are all grouped into one current car object and placed in a special *reference frame*. Normal links occur between objects in the previous frame and objects in the current frame. Additionally, the stored car object, which was removed from its frame, (from Step 1) is linked to the current car object which is in the reference frame. In any given sequence, there is only one reference frame.

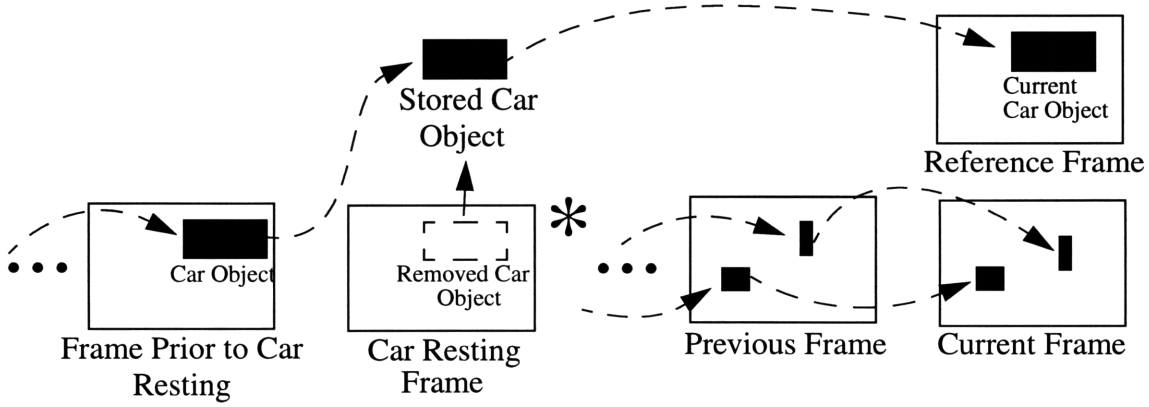


Figure 4.6: Creation of the Motion Graph. The *'ed frame represents the frame prior to the background image being updated

Figure 4.6 demonstrates the creation of this new motion graph. As indicated by the dotted lines, all objects maintain their tracks using this method. Notice that even though the car

object disappears from future frames (due to the updated reference image), an exit event is not detected because its track is maintained throughout every frame. Using this method, the system is able to keep track of the car object as well as any objects overlapping the car. If an object appears intersecting a car object, an INCAR event is reported. If an object disappears while intersecting a car object, an OUTCAR event is reported. Figure 4.7 shows the output of the system. In this case, the system currently knows of three distinct objects: the briefcase, the car, and the person. The goal of detecting overlapping objects has been achieved because the system knows about the person even though its blob overlaps with the car's blob. Therefore, if this blob disappears while its overlapped with the car, then an INCAR event will be reported.

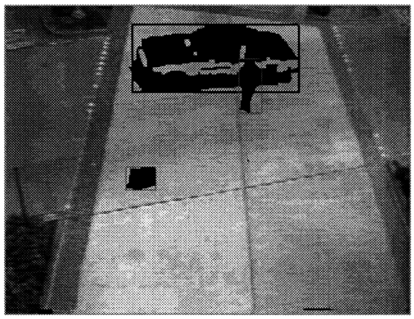


Figure 4.7: Final Output of System

The system will continue to operate in this manner until the car in the reference frame begins to move again. When the car moves again, the system should revert to its normal single reference image state. The system detects the car's motion based on the movement of its centroid. It compares the position of the centroid of the stored car object, which represents the original position of the car at rest, with the centroid of the current car object, which represents the current position of the car. Other methods could be used to detect motion such as monitoring the velocity associated with the car object. However, the more accurate method seemed to be monitoring the position of the centroid. Figure 4.8 shows

the slight movement of the car.

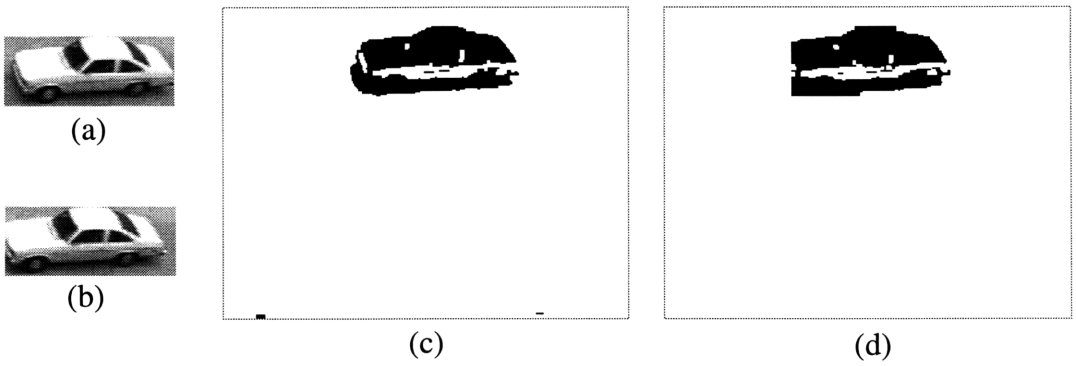


Figure 4.8: (a) Reference car image. (b) Moving car image. (c) Reference car difference image. (d) Moving car difference image

By testing several values, it was determined that a centroid movement of 6.5 pixels was satisfactory in signifying the movement of the car. Smaller values caused noise to be mis-taken for car movement and larger values detected movement too late. Therefore, once the centroids differ by more than 6.5 pixels, the system detects that the car is moving and the following sequence of events occur to restore the system to its original state:

1. An object representing the moving car is created in the current frame
2. The stored car object is linked to this new moving car object in the current frame
3. Objects in the previous frame that intersect the moving car are removed from that frame
4. The car background image is merged with the updated reference image to restore the original reference image

5. Normal differencing continues

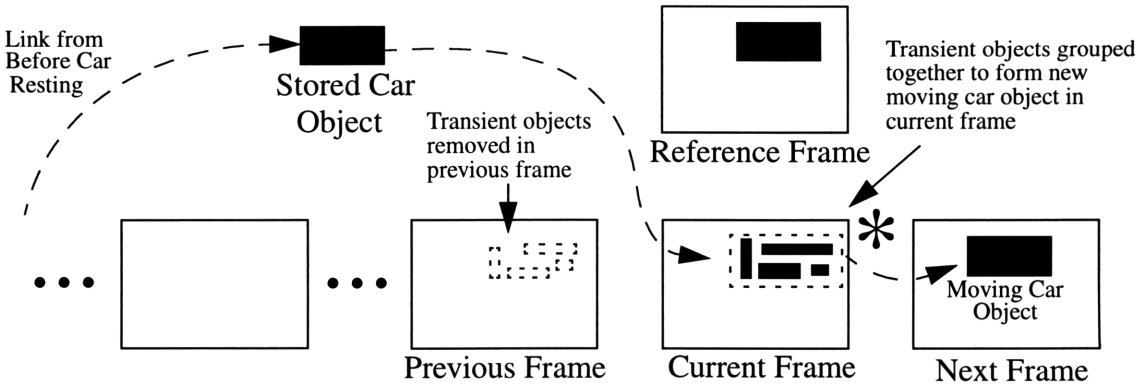


Figure 4.9: Restoration of Normal Differencing. The *'ed frame represents the last frame prior to the original reference image being restored.

Figure 4.9 demonstrates how the system is restored to its original state. Note how there is one continuous track that represents the path of the car throughout all time. Therefore, no false enter or exit events are detected.

When the car begins to move again, transient blobs appear in the foreground difference image due to the fact that the car is in the updated reference image as seen in figure 4.10.

(a) (b) (c)

Figure 4.10: (a) Updated reference image. (b) Current video Image. (c) Foreground difference image

Therefore, to create a new moving car object in the current frame (Step 1), these transient objects, which are identified by their intersection with the location of the resting car, are grouped together as one car object. If there are no transient objects, a copy of the stored car object is inserted into the current frame. This way, there is definitively a car object in the current frame for the stored car object to link with.

Transient objects might also appear in the frame prior to the one in which the system detects the car is moving. These objects result from the initial motion of the moving car that is not detected by the system. The transient objects must be removed from the previous frame in order to prevent them from being linked to the new moving car object that was just created in the current frame. This linking would cause problems because the motion graph would then have two objects, the transient objects and the stored car object, linked to the current car object.

After the steps described above occur, the system continues as normal until another car comes to rest.

Chapter 5

Object Integrity

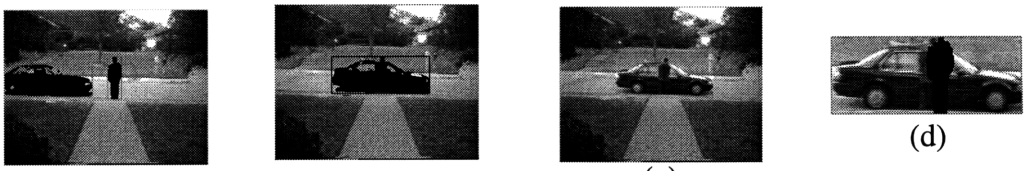
Using the basic method described in the previous chapter, the system can detect people entering and exiting cars as long as the car is unoccluded when it comes to rest and when it begins to move again. However, a new issue is encountered if we take away this assumption of an unoccluded car. How does the system maintain the integrity of objects which occlude a car when it comes to rest or when it moves again? These situations will be dealt with individually. Again, the implementation is running under the assumption that neither the car nor the object can be moving at the same time.

5.1 Car Resting

An object occluding the car when the car comes to a rest causes a problem because the object becomes updated into the reference image. Because the object is now in the reference image, it no longer appear as a foreground object, and is lost. This problem can be dealt with by somehow recording the fact that an object was present when it gets updated into the reference image. This correction would allow the system to continue to track the object in successive frames. In order to minimize changes to the system, an object is recorded as having been present by modifying the reference image to create a pseudo-object. The rest of this section will describe the creation of this pseudo-object.

When an object's and a car's blobs merge together, the object along with its location is stored. Then, when the car comes to rest, all the usual mechanisms occur to update the reference image. However, the reference car image in the updated reference image is further modified. The pixels corresponding to the last known location of the occluding object's blob are set to black. This black blob in the updated reference image is called *fake background*. The result of this modification is to recreate the occluding object in the

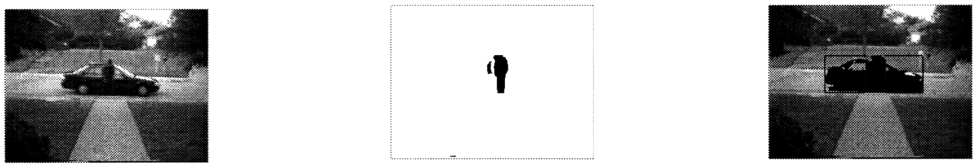
next frame even though the car has been merged into the reference image. Therefore, the integrity of the object is maintained even when the car comes to a rest. It is important that the object does not move once its blob merges with the moving car blob. Otherwise, the last known location of the object will be incorrect resulting in the fake background being in the wrong area of the car reference image.



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

Figure 5.1: (a) Final output image prior to merging blobs. (b) Final output image after car comes to rest. (c) Updated reference image. (d) Modified car reference image with fake background.

The sequence in figure 5.1 shows a person waiting to be picked up by the car. In image (a) the person is standing still waiting to be picked up. In image (b), the blobs are merged and the car comes to a rest, so the updated reference image (c) is created. The reference car image of that image is modified (d) to put black pixels in the last known location of the person blob.



(a) (b) (c)

Figure 5.2: (a) Current video image. (b) Foreground difference image. (c) Final output image.

In Figure 5.2, one can see how the foreground difference image (b) now contains a blob representing the person. This person blob results from the person in the current video image (a) differing from the black pixels (fake background) in the updated reference image modified previously. As one can see in the final output image (c), there are two objects, one representing the car, and one representing the person; thus, object integrity is

maintained.

However, this fix creates a new problem. Once the object begins moving again. the fake background will cause a false object to appear. False objects will appear if the object moves to a different location. This problem is already solved because occluding objects which appear are already checked for their validity as will be seen later (Section 6.3). The other problem occurs if the object enters the car. Then, the once valid object becomes invalid, but it is not checked for its validity since it is not a new appearance. Therefore, objects which come to a rest or objects that still exist when the car begins to move, are checked to see if they are false due to fake background. This check is done by seeing if over 70% of the pixels in the object's blob are black. If this is true, than the appropriate pixels of the reference car image representing the fake background are updated with the pixels from the current car image.

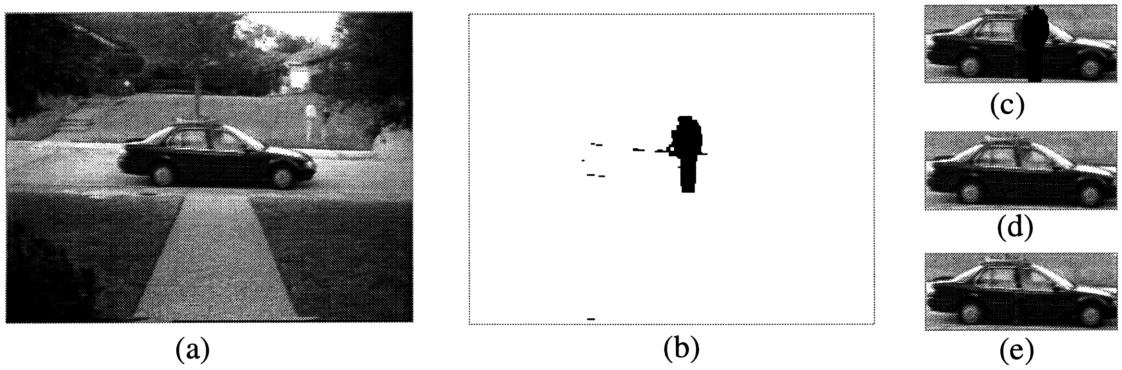


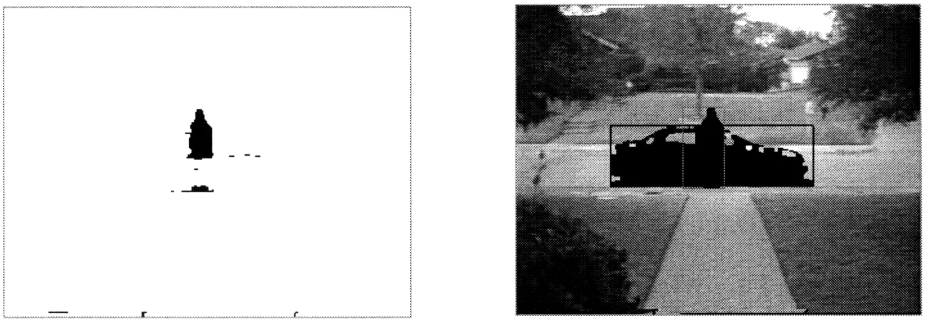
Figure 5.3: (a) Current video image. (b) Foreground difference image. (c) Reference car image. (d) Current car image. (e) Corrected reference car image.

Figure 5.3 represents the moment the person in the previous example finally enters the car (a). The foreground difference image (b) still shows a person blob due to the fake background in the updated reference image. Therefore, the reference car image (c) with the fake background is updated with pixels from the current car image (d), to create the corrected reference car image (e).

5.2 Car Moving

An object occluding a resting car when the car begins to move again causes problems because the object gets grouped with the car object and its track is lost. Therefore, when the car moves away, the remaining object causes the system to report a false enter event. A new method was implemented to handle this situation. When a car begins to move again, the usual mechanisms occur to restore the original reference image. However, when the system is grouping objects in the current frame to create a new moving car object, it checks to see if any of the objects correspond to a real object. This check is done by seeing if the object has a large enough track. Objects which just correspond to the motion of the car do not have long tracks associated with them. Next, the system moves backwards through the object's track to find an old state of the object's blob in which the blob is less than three feet wide and whose area is not changing substantially. This step is done in order to filter out car motion objects and find the actual blob which represents the occluding object. The blob representing the occluding object in the current frame is incorrect because of the motion of the car. The car motion creates transient objects that get grouped with the real occluding object creating an incorrect blob. Therefore, this backtracking step is necessary. Once the actual blob is found, it is stored. Once the car moves away from the object and splits into two blobs, the stationary blob is checked to see if it matches the stored blob. If it does, it is linked back to that object. Because of this match-

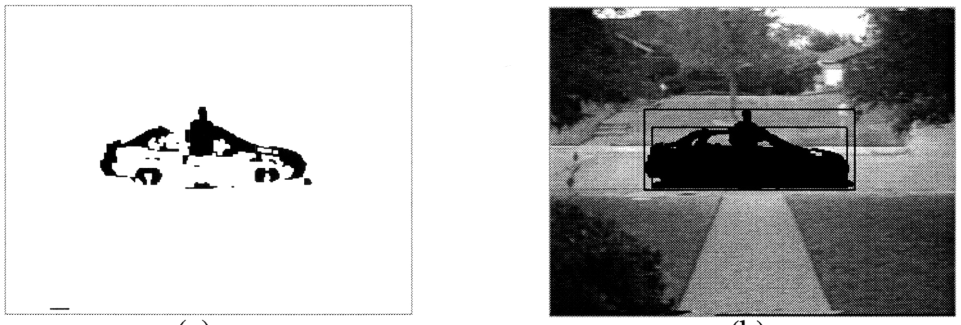
ing step, it is important that the object does not move while the car moves away. In this way, the occluding object's integrity is maintained even after the car has moved away.



(a) (b)

Figure 5.4: (a) Foreground difference image. (b) Final output image.

In figure 5.4, a person has just exited a resting car as seen in the final output image (b). The person is recognized by the system as a real object as seen in the foreground difference image (a).



(a) (b)

Figure 5.5: (a) Foreground difference image. (b) Final output image.

In figure 5.5, the car begins to move again so that the transient blobs begin to appear in the foreground difference image (a). These blobs get grouped together to form one object as seen in the final output image (b), but the blob still corresponds to the person object from the previous figure.

Figure 5.6: Backtracking to find a valid object.

Now, the system backtracks through the blob's track until it finds a blob that most likely corresponds to the actual object (the *'ed blob). This blob is stored for later use.

Figure 5.7: Final output image

Finally, when the car moves away and the person blob is separated from the car blob, the "new" person blob is compared with the blob which was stored earlier. It is similar in appearance and at the same location, so it is linked to this object. Once again, object integrity is maintained.

Chapter 6

Robustness

Given the basic method, the system is subject to frequent errors. These errors mainly result from the fact that a bad car image is updated into the reference image. A bad car image is an image in which there are objects other than the car in it. The result of a bad car image being updated into the reference image is that false objects appear which cause false events to be reported. Two solutions exist for this problem.

One solution is to implement methods which ensure that a good image of the car is updated with the reference image. This solution involves taking a better reference car image rather than just the image of a car when it comes to a rest. One way of doing this would be to look at every possible view of a car as it enters a scene and to pick the best view of the car as the reference car image. This method would be difficult because the system would need a heuristic to decide what the "best view" is. Additionally, the system would have to align this "best view" with the position of the car in the current frame. Another way of getting a better car image would be to follow a method employed by Friedman and Russell [4] to update their background image. They basically use a time averaged background image to account for lighting changes. They use a method called exponential forgetting in which each successive image is weighted such that its contribution to the final image decreases exponentially as time goes on. This technique could be applied to the car image by making the car image a time-averaged image of all the views of the car seen so far. Therefore, occluded views of the car would have minor impact on the car image. However, the system would still need to perform alignment between the views of the cars. The advantage is that you don't have to decide what is the "best view" of the car.

The second solution is to implement methods which correct bad car images which are updated into the reference image. One method of doing this would be to use the selective updating technique [11] of background images. The concept of selective updating is that a valid background image is maintained by replacing pixels of the background image with pixels of the current image which are thought to be current background pixels. This general method was used because it was simpler. Most of the algorithms described later will deal with deciding which pixels to replace.

To use the selective updating technique, the system would need to determine which objects are actually due to background. Then, all the system needs to do is eliminate those false objects. One method of doing this would involve checking every object in every frame to see if its a valid object. However, this would be computationally expensive. So the preferred method, which was the method implemented, would involve only checking an object for its validity when it first appears. This method operates on the assumption that an object which is valid when it appears will always be valid. The following methods provide different heuristics to determine if an object is valid, and if it is not, to remove it and correct the cause of it.

6.1 Noise Objects

One source of false objects using the new technique is noise. The noise results from small motions in the car. For example, if the system updates the reference image with the car image when the car is not quite at rest, the resting car will be slightly displaced from the car in the updated reference image, causing false objects to appear. However, almost all these situations have one thing in common. The false blobs in the foreground difference image created by the displacement resemble the edges of the car. Figure 4.10c shows the foreground difference image generated when a car is displaced slightly from its original resting position. As one can see, the blobs resemble the edges of the car. Therefore,

whenever a new object appears, a binary edge image is created for the car in the reference image. This binary edge image is created using the Sobel edge detection technique. Next, the pixels in the object's blob are compared to the pixels in the edge image. If more than 70% of the blob's pixels match the pixels in the edge image than the system detects classifies the object as noise and discards it. Then, the updated reference image is corrected. This correction is done by replacing the pixels in the reference car image which correspond to the object's blob with the corresponding pixels in the current car image; thus eliminating the noise object from future frames. The following figures will step through the entire process.

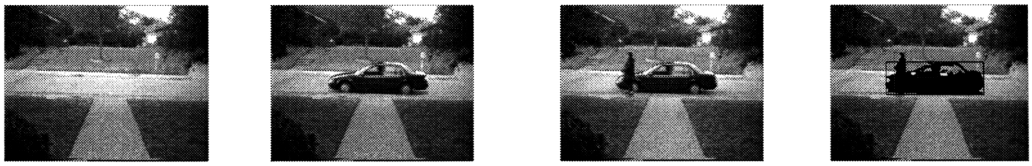


Figure 6.1: (a) Background image. (b) Updated reference image. (c) Current video image. (d) Final output image - Note: there is an object which appears over the window of the car which is due to noise.

Figure 6.1 shows the setting for a situation in which a person exits a car. After they close the car door, it causes some noise object to appear over the window area of the car door.

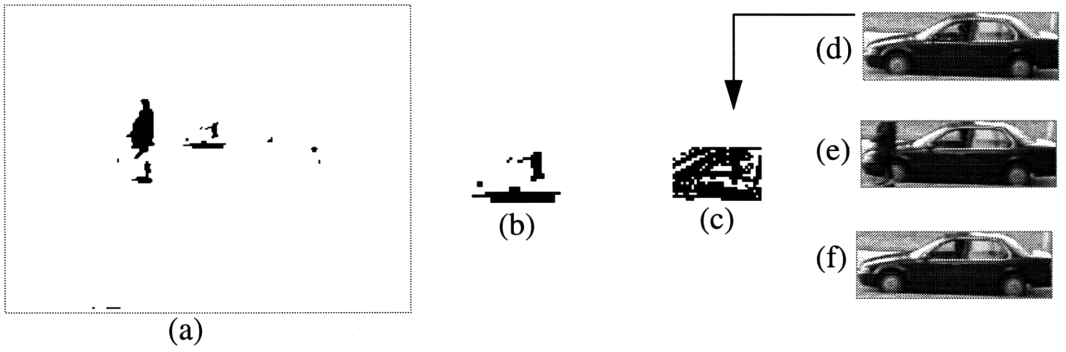


Figure 6.2: (a) Foreground difference image. (b) Object blob from foreground difference image. (c) Edge image. (d) Reference car image. (e) Current car image. (f) Corrected reference car image.

Figure 6.2 shows how the system compares the object's blob's pixels (b) to the edge pixels

(c) and determines that over 70% of the pixels match. Therefore, pixels in the reference car image (d) are updated with the pixels in the current car image (e) to create the new corrected reference car image (f). In this case, the difference is hard to tell.

6.2 Non-Occluding Moving False Objects

Another source of false objects arises from reference car images which contain non-occluding moving objects in them. These bad reference car images occur because of over-aggressive grouping of blobs to form objects. Hence, non-car objects get merged with car objects and the resulting object is labelled as a car. When this "car" comes to rest, its image updates the reference image producing an incorrect updated reference image. The result is that when the non-car object has moved to a different location, a false object appears in its old location because it is in the updated reference image. The most general way of fixing this problem would be to improve the segmentation phase of the system. If the system were better able to segment objects, than the likelihood of grouping another object with the car would be smaller. Thus, the system could be more confident that its reference car image only contains a car. However, improving segmentation is a more difficult task that just correcting bad car reference images.

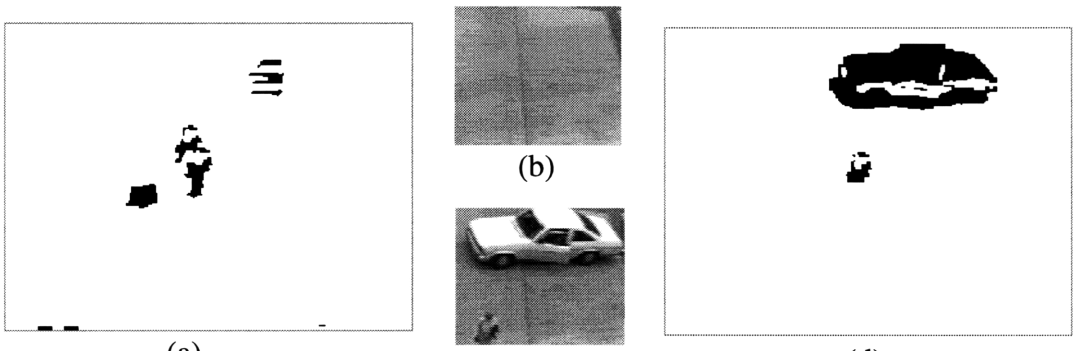
To prevent the detection of false objects which result from these bad car reference images, the system checks to see if appearing objects are valid. Objects are valid if and only if they appear when the current frame is differenced with the original reference image. All other objects, are due to revealed background resulting from an object, which accidentally got updated into the reference image, moving away. This method is implemented by validating an object anytime it first appears in the scene. This is done by doing the following. The pixels of the object's blob in the foreground difference image is compared to the corresponding pixels in the car difference image. If less than 50% of the pixels match, then the object is most likely a false object. This method works because the car

difference image will only contain blobs for objects that are objects in the current frame since it is differencing with the background car image, which is a subimage of the original reference image. Therefore, if the foreground difference image has a blob that does not correspond to a blob in the car difference image, then the blob necessarily represents a false object. The following figures will step through an example of this process.

(a) (b)

Figure 6.3: (a) Updated reference image. (b) Current video image

In figure 6.3, the person walking by the car is incorrectly grouped with the car when the car comes to a rest. Therefore, the whole region containing the person and the car are merged into the updated reference image (a). In the current video image (b), the person continues walking which will cause a false object to appear.



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

Figure 6.4: (a) Foreground difference image. (b) Car background image. (c) Current car image. (d) Car difference image

In figure 6.4, the foreground difference image (a) contains blobs representing objects that appear because of a difference between the current video frame and the updated reference

image shown in figure 6.3a. The car difference image (d) contains blobs representing objects that appear because of a difference between the current car image (c) and the car background image (b). One can see that the top half of the person blob in the foreground difference image (a) does not exist in the car difference image (d).

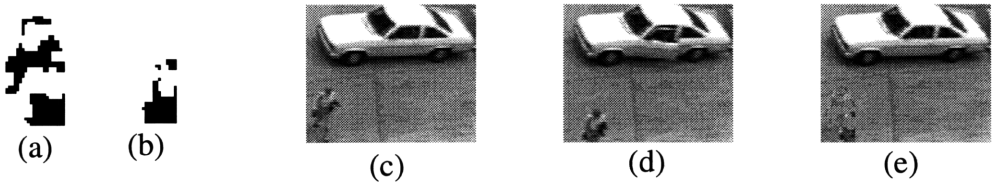


Figure 6.5: (a) Object blob from foreground difference image. (b) Corresponding object blob from car difference image. (c) Reference car image. (d) Current car image. (e) Corrected reference car image.

Figure 6.5 demonstrates how the reference car image is updated. Less than 50% of the pixels of the object's blob in the foreground difference image (a) matches the corresponding blob in the car difference image (b). Therefore, the pixels in the reference car image (c) which correspond to the pixels of (a) are replaced by the corresponding pixels in the current car image (d). This process results in the corrected reference car image (e).

6.3 Occluding False Moving Objects

Not only can false objects arise from non-occluding objects being included in the reference car image, but they can also arise from occluding objects that are accidentally included as part of the reference car image. Bad reference car images are taken when the system updates the reference image with a car image that has a moving object occluding the car. This error generates false objects in that when the occluding object moves away, false objects appear due to the object being in the updated reference image.

The problem here is very similar to the problem of objects in a reference image which get removed, exposing hidden background, and causing ghost objects to appear. This

problem of "exposed background" is handled in the same manner as Jonathan Courtney handled it in his video indexing system [2].

To correct for this problem, anytime an object appears in the scene, it is validated by doing the following. The system generates a binary edge image for both the reference car image and the current car image using the Sobel edge detection technique. Then, the edge pixels of the object's blob in the foreground difference image are compared with the edges in both edge images. If more pixels match the reference car edge image than the current car edge image, then the object is considered false and is discarded [2]. Finally, the reference car image is corrected in a manner similar to that of non-occluding moving objects. The following figures will step through an example of this process.

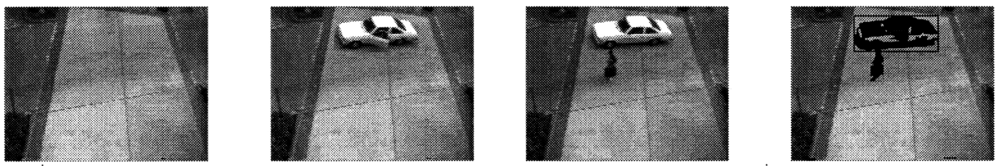


Figure 6.6: (a) Background image. (b) Updated reference image. (c) Current video image. (d) Final output image - Note: there is an object which appears over the door of the car which is due to the open door in the updated reference image.

Figure 6.6 shows an instance where the system recognizes that the car is resting kind of late. Hence, the car door is in the process of being opened when the car image is merged to create the updated reference image (b). Therefore, after the person completely exits the car and closes the door in the current video image (c), there is an extra object which

appears over the area of the car door, as evident in the final output image (d).

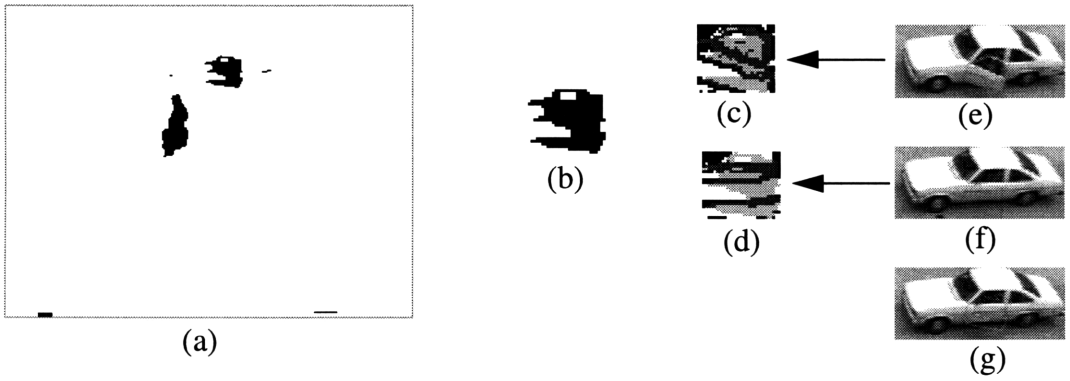


Figure 6.7: (a) Foreground difference image. (b) Object blob from foreground difference image. (c) Edge image from reference car image. (d) Edge image from current car image. (e) Reference car image. (f) Current car image. (g) Corrected reference car image.

Figure 6.7 shows how the system compares the edges' of the object's blob (b) to the edge image (c) of the reference car image (e) and the edge image (d) of the current car image (f). The edge images in figure 6.7 have the object's blob overlaid on top of them in light gray. Black pixels represent pixels that are only in the edge image. Dark gray pixels represent pixels that are both in the edge image and in the object's blob. As one can see, more of the edges of the blob coincide with the reference edge image (c) than with the current edge image (d). Therefore, the pixels which correspond to the object's blob in the reference car image (e) are replaced with the corresponding pixels in the current car image (f) creating the corrected reference car image (g).

Chapter 7

Results

To test the principles behind the modified AVS system, sequences of video that represented ideal events were captured to disk. These sequences represented events which the modified system should be able to recognize. Next, staged sequences of repeated events were recorded and run on the system to test robustness. Finally, longer sequences were recorded and run directly from video tape to test how the system would work under real conditions.

7.1 Software Sequences

These sequences of video were captured to disk. Each sequence represented a case which the system should be able to reliably detect people/car events. The advantages of using these sequences were the fact that their results were deterministic and that they had less noise in them. Videotaped sequences tend to have a lot of noise in them.

7.1.1 Simple Sequence

The first sequence was filmed from the 3rd story of an office building overlooking the driveway in front of the building. A car drives up and a person exits the car, walks away, deposits a briefcase, and finally reenters the car. Then, the car drives away. In this segment, the system successfully detects the person exiting the car. However, the person

entering the car is missed because the person gets grouped with a person walking near the car. Frames from this sequence are shown in figure 7.1.

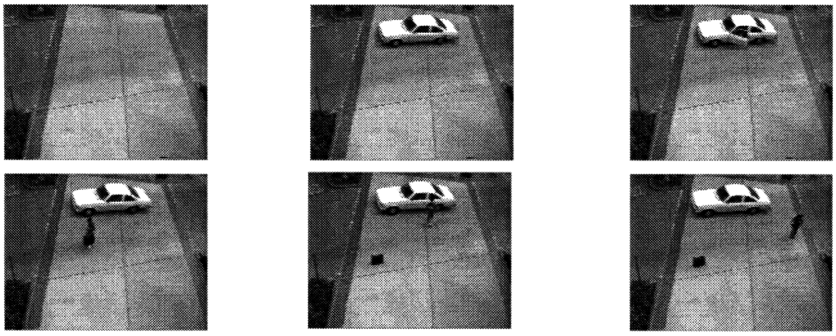


Figure 7.1: Sample Simple Sequence

In the next sequence, the car drives up and a person exits the car, walks away, removes the briefcase, and finally reenters the car. Again, the car drives away. In this segment, both the person entering and exiting the car are recognized. In both these sequences, there was only the one false negative mentioned earlier and no false positives.

The number of false objects that the system eliminated using the robustness techniques (Chapter 6) were as follows:

- 4 Noise Objects
- 3 Non-Occluding False Objects

7.1.2 Pickup Sequence

This sequence was filmed in front of a house looking at the street in front of the house. In the sequence, a person walks into the scene and waits at the curb. A car drives up, picks up the person, and drives away. The system correctly detects the person entering the car. There are no false positives or negatives.

The number of false objects that the system eliminated using the robustness techniques were as follows:

- 3 Noise Objects

7.1.3 Drop Off Sequence

This sequence was also filmed in the same location as the previous one. In this sequence, a car drives up and a person is dropped off. The car drives away with the person still standing in the same location. Then, the person walks off. The system correctly detects the person exiting the car and does not report a false enter event, when the car moves away. No false objects needed to be eliminated in this sequence.

7.2 Staged Videotape Sequences

These sequences consist of repeated staged events that were done to test the robustness of the system. A camcorder was used to record these sequences on videotape. Then the system was run on the videotaped data.

7.2.1 Basic Sequence

This sequence was designed to test the system's basic method of detecting people entering and exiting cars. In this sequence, a car drives up and comes to a rest. A person exits the car and walks away from the car. Then, the person walks back to the car and enters it and the car drives off. This event is repeated five times. In the sequence, 6 cars are seen, 5 INCAR events occur, and 4 OUTCAR events occur.

	Number of Cars	INCAR events	OUTCAR events
Ground Truth	6	5	4
Correct	6	4	3
False Positives	1	2	4
False Negatives	0	1	1

Table 7.1: Results of Staged Sequence 1

Table 7.1 shows the results of the system. In general, there seems to be many false positives due to the fact of noise in the videotaped images.

7.2.2 Object Integrity Sequence

This sequence was designed to test the system's method for handling object integrity. In this sequence, a car drives up and rests. A person exits the car, waits for the car to drive away, and walks away. Then, the person reenters the scene and waits for the car to drive up. The person enters the car and the car drives away. This sequence of events is repeated four times. In the sequence, 12 cars are seen, 4 INCAR events occur, and 4 OUTCAR events occur.

	Number of Cars	INCAR events	OUTCAR events
Ground Truth	12	4	4
Correct	6	2	3
False Positives	1	0	0
False Negatives	6	2	1

Table 7.2: Results of Staged Sequence 2

Table 7.2 shows the results of running the system on this sequence. This sequence contained a lot of noise which caused the system to miss a lot of cars and perform so poorly. These results show also that the method for maintaining object integrity might need some modifications.

7.3 Real Videotape Sequences

These sequences were run on the system straight from a videotape. These were all run at a higher threshold because of a lot of noise on the videotape. However, this tended to decrease the performance of the system. Additionally, the modified system seems to be very sensitive to exactly which video images are processed. Therefore, the results of running the system on this data were non-deterministic. The results reported here are for one run of the system. Furthermore, extra reference images were taken manually to account

for lighting changes in these sequences. Some of these sequences contain the software sequences described in the previous section.

7.3.1 Dark Day

This is a 15 minute sequence that was recorded from the 3rd floor of a building on a fairly dark day. In that time span, 8 cars passed through the camera's field of view. The system detected 6 cars correctly and one false car (due to people grouped together). One car that was not detected was due to its small size. The other car was undetected because the system slowed down (due to multiple events occurring) and missed the images with the car in them.

In this sequence, two people entered a car. However, both events were missed because the car was not recognized as resting due to the dark lighting conditions on this rainy day.

7.3.2 Cloudy Day

This is a 13 minute sequence in the same location as before except it is a well-lit cloudy day. In this time span, 9 cars passed through the camera's field of view and all of them were detected by the system. There were a total of 2 people entering a car and 2 people exiting a car. The system successfully detected them all. Additionally, it detected an extra person exiting a car due to a person walking near a car.

7.3.3 Cloudy Day - Extended Time

This is a 30 minute sequence in the same location as before. In this time span, 28 cars pass through and all of them were detected. The system detected one person exiting the car when there were actually a total of 3 people. The 2 people were missed because the car was on the edge of the camera's field of view and so it was not recognized immediately as a car.

Chapter 8

Conclusion

The work described in this thesis allows the TI AVS system to now detect simple events involving people and cars. The system performs very well on recorded software sequences, detecting virtually every event with few false positives or negatives. On about an hour and a half of videotaped data, the system detected 17 out of 26 events, reporting 7 false positives.

In general, more tests need to be run to determine how well the modified AVS system performs. So far, no live tests have been done; all have been done from videotaped sequences. Live tests would be useful because they would eliminate the problem of noise that is prevalent with the videotaped data. However, it seems like the best way to test the system is to use videotape to record long sequences in a busy driveway and somehow clean up the videotape signal. For simple cases where the car is unoccluded when it is resting and when it is moving, the system seems to reliably detect people entering and exiting cars. In more complicated sequences the system is more prone to error.

This research represents the initial move of the TI AVS system to function outdoors. It presents a unique method for handling interactions between people and cars by additional image differencing. The approach is a combination of a variety of fairly simple routines which happen to make the system work. Future work would involve strengthening object integrity as well as analyzing and correcting frequent errors the system makes. Many of the methods implemented to correct errors are specific solutions to general problems of which there are probably more than one way of solving. Other solutions could be explored. Additionally, any improvements to the general vision techniques of segmentation would help the system as well. Furthermore, future research could relax some of the

tight assumptions that were made in developing this modified system. For example, different orientations of cars could be handled by using other methods besides aspect ratio for detecting cars. One could use motion information about the car, such as its direction, to determine its orientation. Also, a car model could be added to help determine a car object's orientation. Finally, one could generalize the technique used in this system to handle any overlapping objects in a scene.

Bibliography

[1] Q. Cai, A. Mitiche, and J. K. Aggarwal. Tracking Human Motion in an Indoor Environment. In *Proceedings of International Conference on Image Processing*, pp. 215-218, 1995.

[2] J. D. Courtney. Automatic Video Indexing via Object Motion Analysis. In *Pattern Recognition*, V. 30, No. 4, April 1997.

[3] J. A. Freer, B. J. Beggs, H. L. Fernandez-Canque, F. Chevrier, and A. Goryashko. Automatic Intruder Detection Incorporating Intelligent Scene Monitoring With Video Surveillance. In *European Conference on Security and Detection*, pp. 109-113, 1997.

[4] N. Friedman and S. Russell. Image Segmentation in Video Sequences: A Probabilistic Approach. In *Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence*, San Francisco, pp. 175-181, 1997.

[5] W. E. L. Grimson, C. Stauffer, R. Romano, and L. Lee. Using adaptive tracking to classify and monitor activities in a site. Submitted to *Computer Vision and Pattern Recognition*, 1998.

[6] I. Haritaoglu. Simple Object (People) Detector and Tracker. Available as http:// www.umiacs.umd.edu/~hismail/PDT/index.html

[7] I Haritaoglu. W4- Real time detection and tracking of people and their parts. Submitted to *FGR98*.

[8] N. Johnson and D. Hogg. Learning the Distribution of Object Trajectories for Event Recognition. In *British Machine Vision Conference*, Birmingham, UK, 1995.

[9] D. Koller, J. Weber, and J. Malik. Towards realtime visual based tracking in cluttered traffic scenes. In *Proceedings of the Intelligent Vehicles '94 Symposium*, pp. 201-206, October 1994.

[10] T. J. Olson and F. Z. Brill. Moving Object Detection and Event Recognition for Smart Cameras. In *Proceedings of the 1997 DARPA Image Understanding Workshop*, New Orleans, pp. 159-175, May 1997.

[11] N. L. Seed and A. D. Houghton. Background updating for real-time image processing at TV rates. In *Proceedings of SPIE*, V. 901, pp. 73-81, 1988.

[12] C. Wren, A. Azarbayejani, T. Darrell, A. Pentland. Pfinder: Real-Time Tracking of the Human Body. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 780-785, V. 19, No. 7, July 1997.