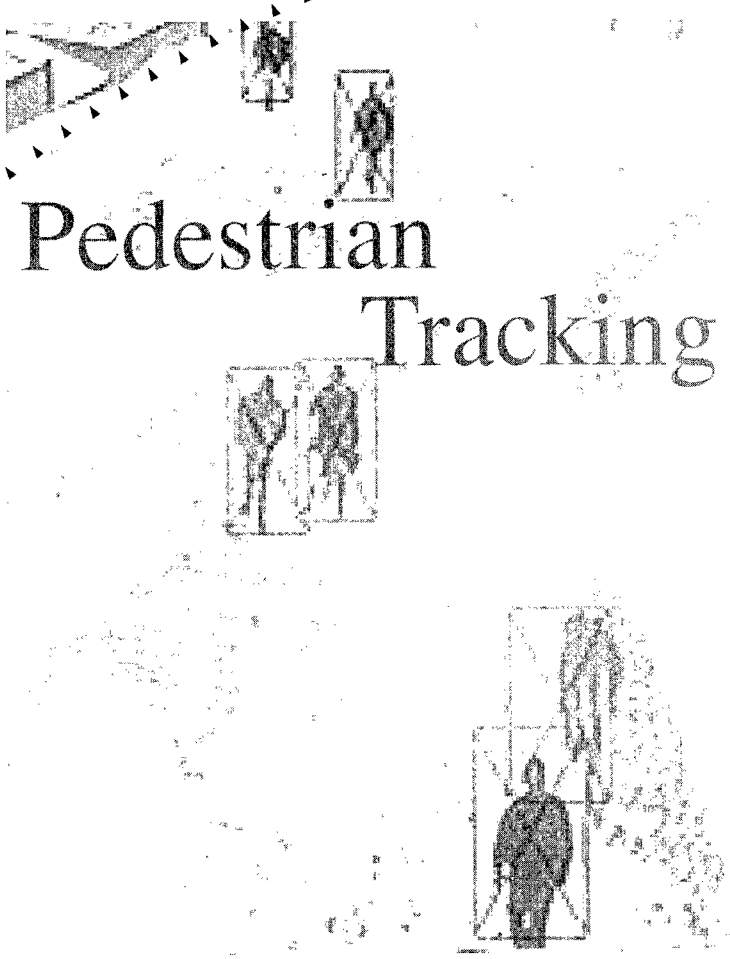
Final Report 1998-17

PB99-114076



Pedestrian Control at Intersections (Phase III)

Minnesota Guidestar

UNIVERSITY OF MINNESOTA
CENTER FOR
TRANSPORTATION
STUDIES

Technical Report Documentation Page

1. Report No. MN/RC - 1998/17	2.	3. Recipient's Accession No.
4. Title and Subtitle PEDESTRIAN CONTROL AT INTERSECTIONS - Phase III		5. Report Date April 1998 6.
7. Author(s) Osama Masoud Nikolaos Papanikolopoulos		8. Performing Organization Report No.
9. Performing Organization Name and Address University of Minnesota Department of Computer Science and Engineering 200 Union Street, S.E. Minneapolis, Minnesota 55455		10. Project/Task/Work Unit No. 11. Contract (C) or Grant (G) No. (C) 74708 TOC #14
12. Sponsoring Organization Name and Address Minnesota Department of Transportation 395 John Ireland Boulevard Mail Stop 330 St. Paul, Minnesota 55155		13. Type of Report and Period Covered Final Report - 1997 to 1998 14. Sponsoring Agency Code
15. Supplementary Notes		

16. Abstract (Limit: 200 words)

This report presents a real-time system for pedestrian tracking in sequences of grayscale images acquired by a stationary CCD (charged-coupled devices) camera. The research objective involves integrating this system with a traffic control application, such as a pedestrian control scheme at intersections. The system outputs the spatio-temporal coordinates of each pedestrian during the period the pedestrian remains in the scene. The system processes at three levels: raw images, blobs, and pedestrians. It models blob tracking as a graph optimization problem and pedestrians as rectangular patches with a certain dynamic behavior. Kalman filtering is used to estimate pedestrian parameters.

The system was implemented on a Datacube MaxVideo 20 equipped with a Datacube Max860 and on a Pentium-based PC. The system achieved a peak performance of more than 20 frames per second. Experimental results based on indoor and outdoor scenes demonstrated the system's robustness under many difficult situations such as partial or full occlusions of pedestrians.

17. Document Analysis/Descriptors tracking pedestrian tracking monitoring of intersections CCD camera pedestrian safety		18. Availability Statement No restrictions. Document available from: National Technical Information Services, Springfield, Virginia 22161	
19. Security Class (this report) Unclassified	20. Security Class (this page) Unclassified	21. No. of Pages 57	22. Price

Pedestrian Control at Intersections

Third-Year Report

Prepared by:
Osama Masoud
Nikolaos P. Papanikolopoulos

Artificial Intelligence, Robotics, and Vision Laboratory
Department of Computer Science
University of Minnesota
Minneapolis, MN 55455

April 1998

Published by:
Minnesota Department of Transportation
Office of Research Services
First Floor
395 John Ireland Boulevard, MS 330
St. Paul, MN 55155

Acknowledgements

The authors express appreciation to Gary Ries and Ron Casselius from Minnesota Department of Transportation for their help and support.

Executive Summary

This report presents a real-time system for pedestrian tracking in sequences of grayscale images acquired by a stationary CCD camera. The objective is to integrate this system with a traffic control application such as a pedestrian control scheme at intersections. The system outputs the spatio-temporal coordinates of each pedestrian during the period the pedestrian is in the scene. Processing is done at three levels: raw images, blobs, and pedestrians. Blob tracking is modeled as a graph optimization problem. Pedestrians are modeled as rectangular patches with a certain dynamic behavior. Kalman filtering is used to estimate pedestrian parameters.

The system was implemented on a Datacube MaxVideo 20 equipped with a Datacube Max860 and on a Pentium-based PC. It was able to achieve a peak performance of over 20 frames per second. Experimental results based on indoor and outdoor scenes demonstrated the system's robustness under many difficult situations such as partial or full occlusions of pedestrians.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

OVERVIEW

Many computer systems require sensory information in order to effectively interact with their environment. The information about a system's environment is important because it provides the raw data with which the system can perceive, analyze, and react to specific objects in the environment. Of all the objects that a computer system encounters, only a subset is significant to the task that the system has to accomplish. We term these to be *objects of interest*, and we concern ourselves with operations that are performed with respect to them.

The sensory information may come from any of a variety of sensors, including:

- cameras,
- global positioning systems (GPS),
- lasers,
- proximity detectors,
- radar,
- and tactile sensors.

Cameras often provide information that is richer, more complete, and covering a larger area than the other sensors listed above. In addition, the new CCD (Charge-Coupled Devices) cameras are less expensive and more accurate than the older vision sensors. However, additional challenges accompany the advantages of the vision modality. One such challenge involves the fact that the vision sensor provides no inherent signal that an object has moved into its field of view. This can be contrasted with a tactile sensor, where the presence of a nearby object is part of the sensor's signal.

Existing research in the field of computer vision has largely focused on other issues, with an understanding that actual implementations would require the addition of an object detection component. Sometimes, the detection problem can be avoided by restricting experimentation to special cases where object detection is trivial. However, this issue must also be given consideration if a computer system is to function in unpredictable, natural environments. Therefore, it is the goal of this report to present a method for automatically detecting objects of interest that may be moving at times and stationary at other times.

This report goes beyond merely detecting the presence of an object. We also connect the detection module to other important sensory components of a vision-based system. Particularly significant is the ability to find landmarks on objects of interest and to know about the projected shape of objects. In addition, tracking techniques are used to monitor objects without human intervention. Our solution to the tracking problem follows the Controlled Active Vision framework [26], which avoids a heavy reliance on a priori information through the use of *optical flow.* Optical flow is induced by any combination of camera or object motion.

One of the contributions of this report is a complete software and hardware implementation of our detection and tracking framework. In the process of constructing this system, we have selected and modified computer vision techniques which are appropriate to the visual detection problem. Many of the techniques used by our framework (e.g., frame-differencing) have also appeared in similar forms in existing research, which is a demonstration of their usefulness. Our solution to the detection problem is innovative in the way in which it has uniquely combined these techniques into a general framework that can be directly applied to real-world situations. We have made modifications to these techniques where necessary, and we have also incorporated our own ideas where the existing literature was lacking (e.g., dynamic segmentation domains). Finally, we have customized the theory to specific needs, including the application areas of transportation. This has demonstrated that our framework can provide precisely the type of information required to effectively manage a situation requiring visual detection. Results from experimentation in this area shows the potential of our approach under general conditions.

BACKGROUND

An extensive body of literature has been accumulated in the computer vision community regarding the study of motion. Most of this literature focuses on the structure-from-motion problem [35], which involves the computation of camera, object, and/or environmental parameters based on relative motion between these entities. Often, assumptions are made in visual motion research that prohibit the use of the proposed techniques in applications such as pedestrian control at intersections. A large number of previously proposed systems exhibit one or more of the following characteristics:

- Systems which avoid the visual detection issue altogether. They assume that their methods are applied on an image after the presence of moving objects has been identified and measured.
- Systems which are applied to artificially trivial conditions that do not occur in natural settings.
- Systems which detect objects of interest only while they are moving. Once objects of interest stop, they become invisible to the motion detection scheme. If the system were responsible for reporting the location of a pedestrian who stops moving in the middle of an intersection, this type of behavior could have drastic consequences.
- Systems which function properly only when the camera is either stationary or moving, but not both. To the contrary, our system generally operates under static camera conditions, but also allows the freedom of visually servoing an eye-in-hand system based upon target location.
- Systems which cope with a <u>single</u> moving target, even though several application areas involve images with several targets.
- Systems which assume that a moving object is a rigid body. Further assumptions may include a specific pattern of motion for the object of interest.

In our work, we have tried to avoid these assumptions, specifically focusing on addressing the goal of visually detecting objects of interest for transportation applications.

The majority of existing attempts at detecting moving objects has employed either optical flow or frame-differencing techniques. Optical flow methods are interesting since they naturally encompass ego-motion of the camera (although some optical flow methods have the equalizing disadvantage of actually requiring ego-motion). For instance, Jain [17], Nelson [24][25], and Thompson *et al.* [32] have compiled a collection of optical flow-based motion detection algorithms that detect a moving object as an inconsistency in some constraint on the optical flow field. Some of these optical flow-based algorithms use a constraint that is based on the orientation of motion vectors away from a focus of expansion (FOE). However, algorithms using the FOE constraint are not reliable when the distance between a moving object and the FOE is small. Another common optical flow constraint is the assumed relation between optical flow gradients and corresponding depth disparities (typically computed with a stereo vision system). Instead of using a stereo vision system in our research, we have restricted ourselves to monocular systems that can acquire visual information with relatively unsophisticated off-the-shelf sensor devices.

In contrast to detection based on optical flow, our framework shares many characteristics with other frame-differencing techniques. An example of these is the system developed by Anderson *et al.* [4]. They detect motion by adapting the Gaussian/Laplacian pyramid that Burt and others have used in a variety of computer vision systems [7][8][9]. The pyramid is applied to the difference between a current input frame and the previously input frame. This has the disadvantage of only signaling appearing and disappearing edges of a moving object. Moreover, the difference responds similarly to large objects as it does to fast ones. Both of these situations do not occur in our approach. The Anderson method [4] uses another Gaussian pyramid to facilitate subsampling down to a level that motion segmentation can be performed by a general purpose computer. However, subsampling is restricted to the logarithmic levels that are provided by a Gaussian pyramid, whereas our framework also utilizes intermediate levels. Also, real-time execution would require that a high performance image processor (like the VLSI chip developed by van der Wal [37]) be available to compute the pyramids.

The frame-differencing system by Dinstein [12] provides an interesting alternative to the figure segmentation approach described as a part of our framework. Dinstein uses four projections to identify the centers of multiple moving objects in a fashion that is only slightly less robust than our method. However, his method locates objects by majority vote (much like Hough transforms); as a result, the method does not provide statistics required by other portions of our framework (e.g., it cannot determine the geometric characteristics of an identified object).

Considering application-specific detection, Allen *et al.* [1][2] have proposed using frame-differencing techniques specifically for the detection of moving objects that are to be grasped by a robotic system. They use a complex motion model for tracking objects moving within a fixed plane. The robot control system used in their research is very similar to ours, utilizing many processing devices (operating at different rates) through the use of predictive filtering. However, this system requires additional modifications before it can be directly applied to situations with multiple moving objects. In addition, it assumes the use of a static camera.

Research on visual object detection has also taken place for use in intelligent transportation systems, such as the CARTRACK [39] and Autoscope [22] systems. CARTRACK detects the rear of vehicles in real-time through the use of a symmetric filter that exploits the regular rectangular form of a car. Autoscope detects the average speed of vehicles in order to monitor traffic flow through a method similar to that of Inigo [16] and Takatoo *et al.* [31]. Because shadows can interfere with the real-time monitoring of traffic objects, Kilger [20] has used transportation heuristics to develop a shadow-handling system. Finally, Mori *et al.* [23] combines motion detection and segmentation with techniques for the recognition of pedestrian and vehicle motion patterns. Although transportation problems are application areas of our framework, we have created a general system that may be used in many situations without relying on conditions particular to one specific problem set.

Considering the visual tracking portion of our framework, we rely on the Sum-of-Squared Differences algorithm discussed by Anandan [3] as a means for calculating displacement vectors. This algorithm has previously been used by Papanikolopoulos [26] in his implementations of

Controlled Active Vision, and it has been used by Tomasi *et al.* [34] to measure the suitability of feature windows for tracking. Vision tracking systems have been proposed for intelligent transportation problems, including:

- a system for tracking vehicles at road junctions [15],
- a collision avoidance system [36],
- a car-following system [19],
- lane-following autonomous vehicles [10][33],
- and a system for visually counting vehicles [27].

The use of tracking information as feedback to our robot control scheme is based on a MIMO adaptive controller of Feddema *et al.* [13]. Similar adaptive schemes have previously been used by Koivo *et al.* [21] and Weiss *et al.* [38] for the control of manipulators. Moreover, adaptive schemes have been used by Brown [6] and Dickmanns *et al.* [11] for the control of various other mechanical systems (e.g., robotic heads, satellites, and cars).

MOTIVATION AND APPROACH

Our system uses a single fixed camera mounted in an arbitrary position. We use simple rectangular patches with a certain dynamic behavior to model pedestrians. Overlaps and occlusions are dealt with by allowing pedestrian models to overlap in the image space and by maintaining their existence in spite of the disappearance of some cues. The cues that we use are blobs obtained by thresholding the result of subtracting the image from the background. Our choice of using blobs obtained after background subtraction is motivated by the efficiency of this preprocessing step even though some information is permanently lost. In a typical scene, a blob obtained this way does not always correspond to a single pedestrian. An example is shown in Figure 1. This is the main source of weakness in many of the systems mentioned above which assume a clean one-to-

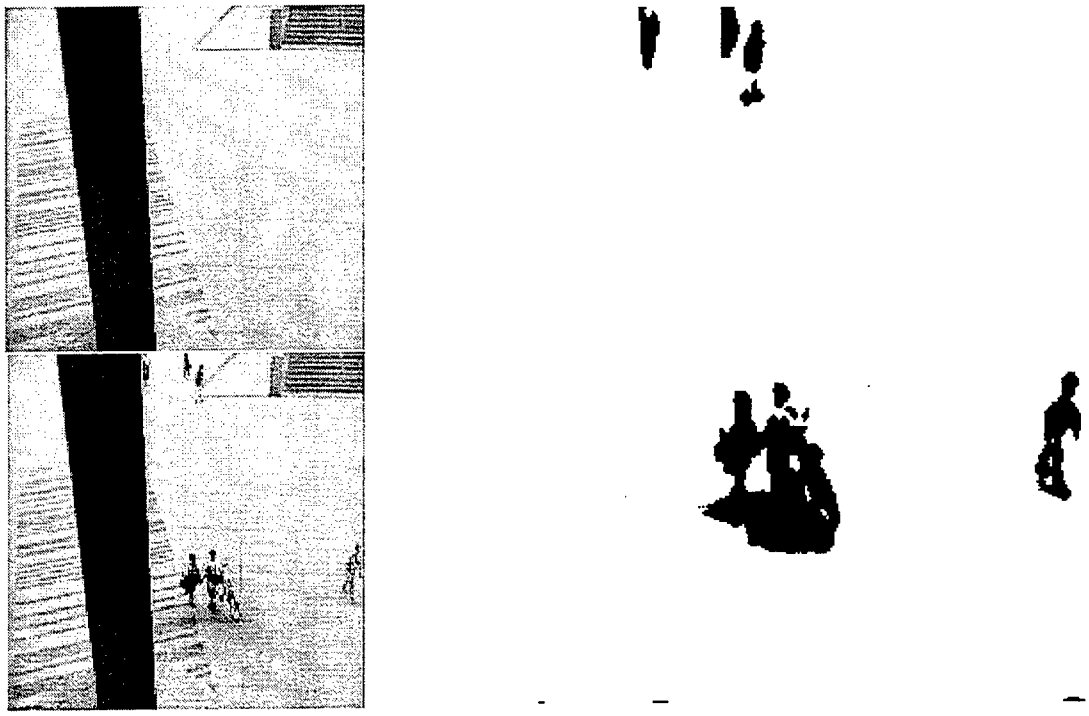


Figure 1. Top left: background image. Bottom left: foreground. Right: difference image showing that a blob does not always correspond to one pedestrian.

one correspondence between blobs and pedestrians. In our system, we allow maximum flexibility by allowing this relation to be many-to-many. This relation is updated iteratively depending on the observed blobs behavior and predictions of pedestrians behavior. Figure 2 gives an overview of the system. Three levels of abstractions are used. Each level deals with a certain type of data and retains a state of the data it produces to be used in conjunction with the data received from the lower level. The lowest level deals with raw images. It receives a sequence of images and performs background subtraction producing *difference images*. In the second level, which deals with blobs, difference images are segmented to obtain blobs which are subsequently tracked. Tracked blobs are passed on to the pedestrians level where relations between pedestrians and blobs as well as information about pedestrians is inferred using previous information about pedestrians in that level.

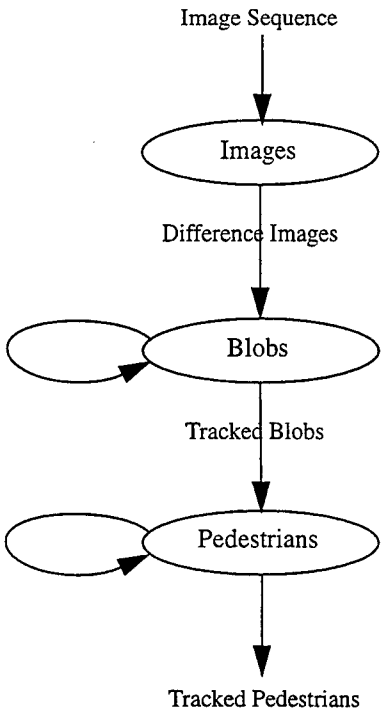


Figure 2. The three levels of abstraction and data flows among them.

CHAPTER 2

DEFORMABLE MODELS AND PEDESTRIAN TRACKING

INTRODUCTION

The system proposed in this chapter uses active deformable models to track pedestrians moving in dynamic real-world scenes. First, figure pixels are separated from a fixed or slowly evolving ground image. Then, a initial segmentation process identifies interesting pixel blobs for tracking. The output of the segmentation process is used to choose the starting position of the control points of the active deformable model.

Active deformable models have been used to track image gradient contours produced by objects. They offer a framework in which local image properties and some local or global model parameters can be combined, without the need for *a priori* knowledge of the scene object being tracked. Because the contour of a walking human body changes in shape and deforms continuously, we believe that methods using active deformable models to track contours are well suited for pedestrian tracking.

APPROACH

We propose to track the movement of pedestrians in an area observed by a stationary camera by approximating with an active deformable model the boundary of the area of image differences created by the motion of the pedestrian. As the image of the pedestrian being tracked translates and deforms in the image, continuous adjustments to the elements of the active deformable model transform and deform the boundary approximation accordingly.

There are many distracting features in the difference image, such as insignificant image deformations caused by pedestrian self-motion, the appearance of other motion in the scene, and image noise. The parameters of the active deformable model and the number of control points can be set so that these factors are ignored while following significant displacements of the pedestrian in the image plane.When the system begins operation, a background image of the scene is captured to be used as the background image, for the calculation of image differences. When a significant motion appears in the scene, as signaled by a large region in the difference image, the image is analyzed for possible pedestrians. If the image fits the criteria established for the classification of a blob as a possible pedestrian, a bounding box is determined for the pedestrian. An active deformable model is then placed around the possible pedestrian with control points on the bounding box. Once the active deformable model has been placed, its movements are controlled by the minimization of an energy equation.

The formulation of active deformable models used in this paper to approximate the pedestrian boundary draws on the work done in recent years by the computer vision community on active deformable models of contours, often referred to as "snakes." The snakes are computed based on an energy term that consists of "curvature" energy, the "image" energy, and the "constraint" energy. These terms are usually sufficient to define an active deformable model approximation of an image contour when all terms vary significantly across the neighborhood of possible control point locations. However, using our current techniques, when the active deformable model is placed, it may have several control points which are far enough from the pedestrian's image that the image gradient is unvaryingly zero throughout the neighborhood of candidate locations. For these points, the "image" term plays no role at all and they only respond to the internal energy and external constraints, rather than to a combination of image energy and constraints.

To facilitate the initial placement of the active deformable model, we have augmented the energy equation with a "model" energy term inspired by the "balloon factor" used by the computer vision community to overcome a tendency toward implosion in their active deformable

models. The "model" term is calculated as follows. First, a neighborhood of the control point in the difference image is examined. If the percentage of difference pixels set within the neighborhood falls short of a predetermined level, the control point is defined as "outside" the pedestrian's image. To bias movement of the control point toward the pedestrian's image, the locations closest to the pedestrian's image are assigned the value -1 for the "model" energy term. Other locations are assigned the value 0.

The locations closest to the pedestrian's image can be determined because the active deformable model control points are numbered counter-clockwise around the closed active deformable model. A similar energy assignment is performed for control points which are "inside" the pedestrian's image. Besides aiding initial placement of the contour, this model energy also occasionally comes into play during later tracking stages when an object moves very quickly o has been temporarily lost for some other reason (e.g., occlusion). Most past applications of active deformable models for contour tracking have attempted to capture the entire boundary of the imaged object. Therefore, the number of control points has been chosen so that the control points fall relatively close together along the image contour. This allows the active deformable model to follow small deformations, but also makes the active deformable model vulnerable to small occlusions. One solution to this difficulty is to give the model a sense of shape through the use of internal or external constraints and to weigh these terms more strongly. This solution is not suitable for tracking pedestrians for two reasons. First, the process of tracking the pedestrian's image requires that the image forces be given considerable weight or performance degrades. Second, pedestrians do not have a well-defined shape. People must move their legs and tend to swing their arms while walking. Worse, they may turn 90 or 180 degrees, presenting an entirely different set of features and silhouette to the camera. For many purposes, it is not necessary to track these deformations. In fact, they are a distraction from the important information -- the horizontal translation of the pedestrian. We have found that simply reducing the number of control points allows

the model to follow a useful approximation of a pedestrian's image boundary while ignoring deformations. Because the speed of the current algorithm is linear in the number of control points, this reduction also has performance benefits.

EXPERIMENTAL RESULTS

The system runs at or near frame rates on the image processing system of the lab (see the next chapter). At these speeds it can successfully track motion of a walking pedestrian, even when the pedestrian's image deforms in unexpected ways such as those caused by thrusting out one's arms or kicking a leg forward in an exaggerated manner. It is also fairly robust with respect to occlusions such as when two pedestrians pass in opposite directions or a single pedestrian passes behind a large tree. Potentially, more than one pedestrian could be tracked simultaneously.

Although such a system should be equally robust with respect to occlusions caused by two tracked pedestrians passing one another, it would probably not be possible to tell whether the active governable models had continued to track the same individual. Such a system might have difficulty distinguishing between two pedestrians approaching one another and then returning the way they came and two pedestrians walking past one another.

Further development of the system will require overcoming the inherent limitations of using a difference image to provide image forces for the active governable model. These problems include short and long time-scale changes in the background caused by lighting changes or continuous regular movement of objects in the scene, for example, the rustling of leaves in the wind. The system is also vulnerable to the effects of camera self-motion. A slight jitter in the camera mount could cause many patches of noise in the difference image. Although these patches will generally be ignored once contour tracking has begun, they do disturb the initial placement of the snake.

SUMMARY

We have presented an approach to pedestrian tracking from a static camera using active deformable models. We use these models to track the boundaries of the pedestrian's image in the difference image. By using the difference image, we avoid some difficulties associated with pedestrian tracking by tracking features, such as the occlusion of features by other parts of the pedestrian. The application of active deformable models also overcomes some of the difficulties, such as the continuous deformation of the pedestrian's image during movement, which pedestrian tracking poses to rigid model-based approaches.

CHAPTER 3

VISUAL DETECTION AND TRACKING HARDWARE

ESSENTIAL COMPONENTS

The purpose of this chapter is to outline the hardware components that we use to implement the visual detection and tracking framework described in this report. From an abstract viewpoint, there are a few general components that are essential for any implementation of the framework. These essential components are:

- an input source of video information,
- image frame transmission unit(s),
- processing unit(s) for performing computations,
- and any desired display or output processing devices.

Naturally, applying the detection framework to robotic problems will require the following additional components:

- robot hardware
- and a control system for driving the robot.

Different instantiations of the above list will produce systems that are able to support the detection framework with different levels of success. It is particularly important that the computation take place on a high performance processing unit. Also important is that the transmission of image frames should take place in an intelligent manner. Systems capable of transmitting frames between parallel hardware elements will allow the framework to be feasible for some real-time applications that are not possible with serial forms of image transmission.

THE MINNESOTA VISION PROCESSING SYSTEM

For our experimentation, we use a system called the Minnesota Vision Processing System (MVPS) [28]. As shown in Figure 3, the MVPS can receive input from either live camera feed or from recorded imagery. Diverse applications of the MVPS are possible because a camera can be mounted in several ways (e.g., fastened to a static tripod, to a passenger car, or to a robot's end-effector), and several forms of recorded imagery are possible (e.g., playback from a VHS video-cassette or a Silicon Graphics movie file). The transmission of image frames is performed by a Datacube MaxVideo 20 video processor. Because of the pipeline architecture of the MaxVideo

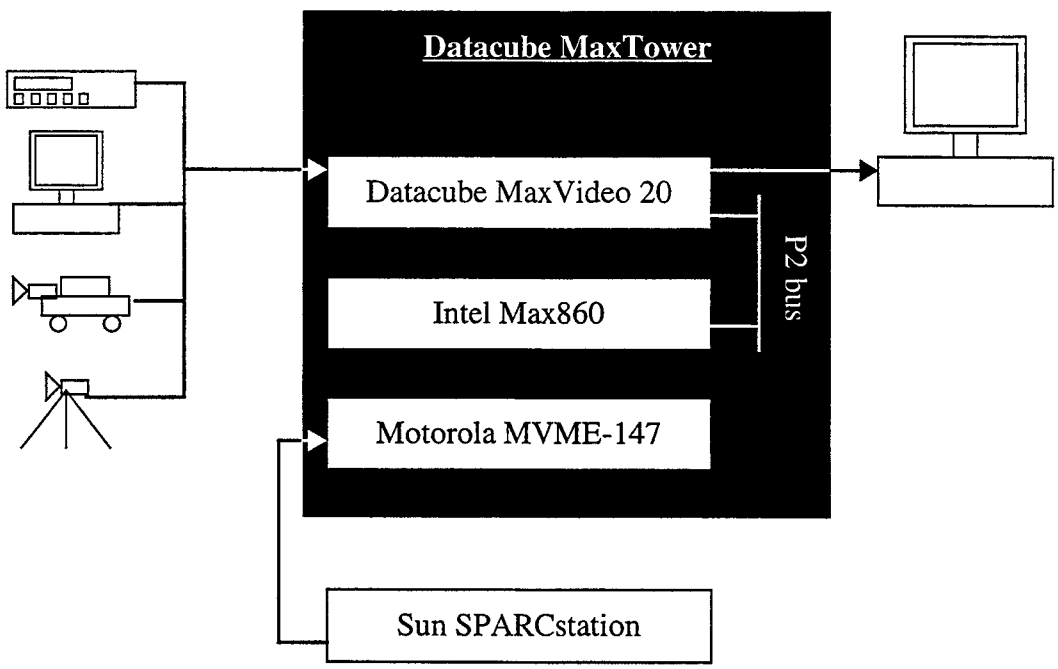


Figure 3. The Minnesota Vision Processing System.

20, all of its calculations are performed at a rate equal to that of the incoming image frames. The MaxVideo 20 contains image processing elements that are programmed with Datacube's Image-flow software libraries to compute the figure and ground images.

The limited collection of MaxVideo 20 processing elements cannot perform the more complicated algorithms required for figure segmentation. Instead, figure segmentation occurs on a separate Intel Max860 processing unit. This RISC processor has a peak performance rating of 80 Megaflops and receives image frames from the MaxVideo 20 through a 20 Megahertz P2 pixel bus. The segmentation of a single test object (200 by 300 pixels) in a 512 by 480 pixel image takes 150 milliseconds. This time was clearly reduced through the use of multiple, smaller domains. The Max860 processor is also used for any correlation computation, because of their computationally intensive nature.

A couple of other devices serve to support the vision system. The high-level control of the MVPS is performed by software executed on a VME-based Motorola MVME-147 Single Board Computer (SBC) running the OS-9 real-time operating system. The MVME-147 is able to drive the other processing units through a portable 7-slot VME chassis. Finally, a Sun workstation is used to develop the software source code and to store multiple versions of the system.

HARDWARE PLATFORM MIGRATION

When considering application of the system, hardware cost was a major issue due to the high cost of the Datacube Maxtower system. Therefore, alternative platforms have been considered. The new hardware consists of a dual Pentium II 200 Mhz PC equipped with a Matrox Genesis DSP board. The system was successfully ported to the new hardware without any loss of performance.

CHAPTER 4

ISSUES RELATED TO DIVERSE OPERATING CONDITIONS

OVERVIEW OF THE SYSTEM

As discussed in previous chapters, the pedestrian tracking system is composed of a single camera mounted at one side of the crosswalk. The signal received from the camera is fed into the Datacube vision processor which performs pedestrian detection and tracking at frame rate. The Datacube then sends the proper command to the traffic light controller. The system components and connections are depicted in Figure 4.

IRRELEVANT EVENTS

One desired property of the system is to limit tracking to pedestrians crossing the street. An intersection is usually a crowded place and not everyone in the view of the camera is crossing the street. Moreover, there could be other moving objects in the scene which should not be taken into consideration. This case will be considered below when we discuss false targets.

Our solution to the first problem was to segment the scene into two regions: interesting and ignored. The interesting region is assigned at setup time. It is simply the projection of the three-dimensional region where pedestrians crossing the street may possibly occupy to the image frame.

The average pedestrian height, the width and location of the crosswalk, and the edges of the two sidewalks are all taken into account to compute this three-dimensional region. Figure 5(a) depicts such a region where the region boundaries are shown as dashed lines. The interesting region is the projection of this region to two-dimensions. It is shown in Figure 5(b) as the region inside the bold line.

Having done this, everything outside the interesting region can be safely ignored. Notice, however that parts of the scene which should be ignored may still show up in the interesting region. Even though false target elimination, which is discussed below, may be able to eliminate many objects that should be ignored, it won't eliminate objects if they were pedestrians. We propose two solutions to this problem. The first is to have multiple cameras positioned in a way such that the intersection of their interesting regions yields only the projection of the base of the three-dimensional region. The output of each camera is processed separately and later when a decision about the presence of a pedestrian needs to be made, all cameras must agree. The second solution is to have one camera but to require that part of the pedestrian (which is normally the feet) to touch the base. Although this solution is more straightforward, it may miss some real targets in cases where the pedestrian legs have a color very close to that of the street.

FALSE TARGETS

Normally, the scene being captured by the camera contains many types of objects in addition to pedestrians. The most common of these are small objects such as moving tree branches and leaves or vehicles that come in the field of view.

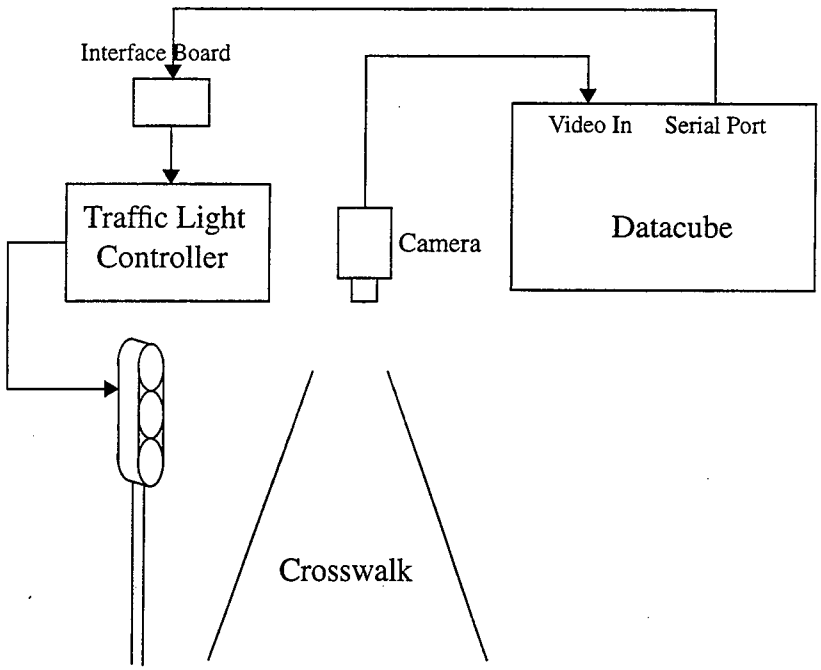


Figure 4. Pedestrian Tracking System Components and Connections.

SMALL OBJECTS

Our system is motion sensitive and therefore will pick up the motion of tree leaves due to wind. Such objects should be regarded as false. Although the interesting region is usually lower than trees, it is still possible that tree leaves may show up in the interesting region. Therefore, this case had to be handled by our system. We found that requiring that the size of the detected object to be larger than a certain value was very effective in eliminating most of the false targets due to small objects. We went one step further and defined other requirements for an object to be recognized as a pedestrian. One of these requirements is the ratio between width and height. These

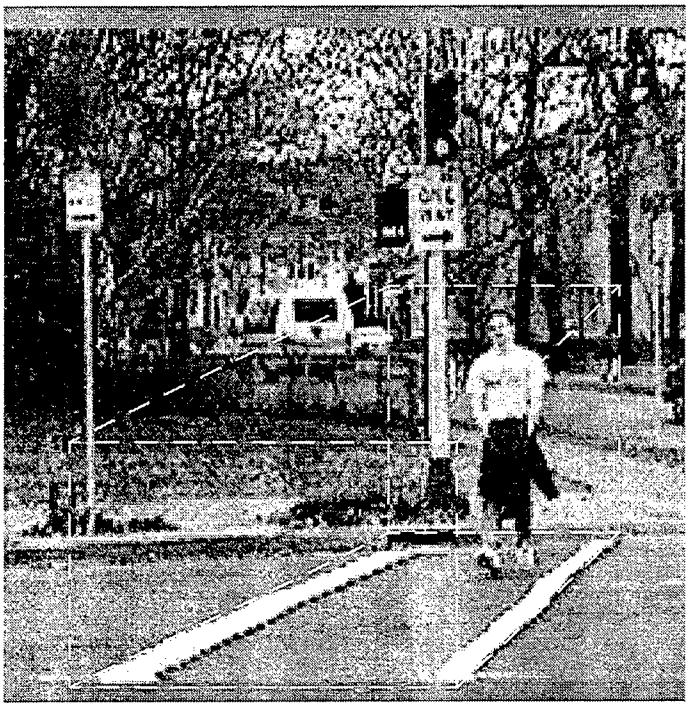
requirements, however, were not enforced in a strict manner so that we do not miss pedestrians with abnormal proportions such as those on wheelchairs. However, it is always necessary that the size of the object be large enough in order to accept the object as a pedestrian.

VEHICLES

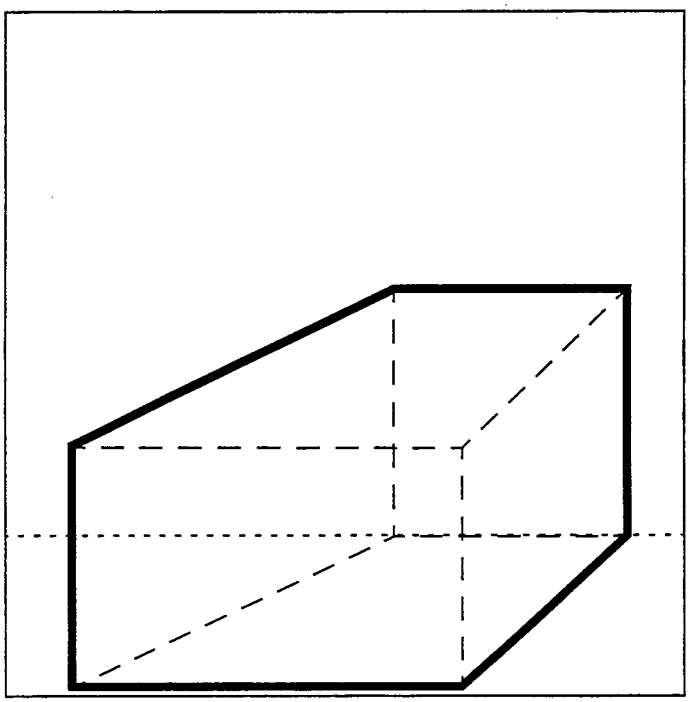
Although vehicles are not supposed to pass the crosswalk lines, they sometimes do. If we do not handle this case, the system may think that a pedestrian is positioned in the crosswalk and make the undesired response of keeping the walk signal on. Another place where vehicles may show up is in a region in the background of another street which happens to be part of the interesting region. Again, if a vehicle in that region becomes stationary, the same undesired result will occur. Our system handles this situation by requiring that objects do not remain stationary. If an object is not moving, it is simply ignored. Doing this, however, may lead to mistakenly ignoring a pedestrian because he or she had stopped momentarily. We solve this by allowing stops for only a short time. One thing that is worth mention here is that in the case a single camera is used, the camera mounting location should be chosen carefully so not to allow another street to show up in the background and within the interesting region.

CROWDS

In some intersections, large groups of people usually cross simultaneously. It would be extremely difficult to isolate each individual because they overlap. This does not pose any problems though because we really do not need to isolate individuals. The whole group of pedestrians can be considered as a single moving object. Therefore, we should not have an upper limit on the size of allowable objects.



(a)



(b)

Figure 5. Interesting Region Calculation.

SHADOWS

In computer vision, isolating an object from its shadow is a very difficult problem. Shadows will be picked up by the system simply because they have a different color from the ground and because they move (as the pedestrian moves). However, because shadows are always attached to pedestrian, they will always be detected as part of the pedestrian. The only side effect is that the pedestrian will seem larger in size but this does not cause any problems. Figure 6 shows how the system picked up shadows attached to pedestrians and considered everything as one unit.

CHANGES IN LIGHT CONDITIONS

Our pedestrian tracking system is based on image differencing where intensity values of the image are subtracted from a pre-stored background to detect changes. If the lighting conditions change due to a cloud that came in the way of the sun for example, a global change in intensity values will occur resulting in a large difference between the image and the background. This would cause the system to think that a huge object showed up in the scene. We solve this problem by allowing the background to be updated with time. An exponential update equation is used to gradually change the background so that it captures the lighting conditions as they change. This method yielded acceptable results. The only side effect was that the system goes temporarily blind right after the change but soon stabilizes and converges to the new condition.

EXPERIMENTS IN DIVERSE WEATHER CONDITIONS

We have tested our system indoors and outdoors in a variety of weather conditions. These range from intersections early in the morning or late at night (with very little light) to intersections with snow or rain. The system was successful in the majority of the cases. It failed in cases of extreme snow (since there was not enough contrast to pick pedestrians) and in cases of intersec-

tions with no light. We have made several modifications in order to have a system robust to clouds, strong winds, and busy intersections. The average number of failures was 3% and this number consisted mainly of false positive alarms (mainly detections of vehicles as pedestrians). However, it will be beneficial if Mn\DOT allows us to use the system in a real intersection in order to make the necessary modifications that will allow us to build a robust and efficient system.

SUMMARY

Even though visual tracking systems that track human bodies are useful in many applications, our system was designed with one application in mind. This application is pedestrian control at intersections. In many aspects of our design, the system was tailored to suit this application. Our main goal was to achieve an acceptable performance in terms of speed and accuracy as required by this application. In this report, we consider several issues specific to the application of pedestrian control at intersections.

Figure 6. Shadows Become Part of the Detected Pedestrian.

CHAPTER 5

BLOB ANALYSIS

INTRODUCTION

A static camera model has the advantage that the scene background remains constant to a certain extent. This allows us to get a good approximation of the location of changes in the image by subtracting the image from the background and then thresholding the result using an appropriate threshold value. This approach has been used extensively. The resulting image, which we call the difference image, is a binary image with 1's in places where there are changes and 0's everywhere else. Of course, changes captured this way may or may not correspond to moving objects. Shadows, image noise, changes in lighting conditions, changes in weather conditions, and small changes in the camera's intrinsic parameters are some examples of changes that show up in the difference image but do not correspond to moving objects. It is also possible that some moving objects do not introduce any change to the difference image, such as a moving object which is very similar in color to the background beneath it. This problem is commonly encountered in computer vision. For example, optical flow and motion field do not always correspond to each other.

There are several techniques which are usually employed to reduce these effects. Adaptive background update, low pass filtering and ignoring small clusters of 1's in the difference image are some of these techniques. We use the latter two techniques in our system. The input to the sys-

tem is a stream of gray scale images of size 512×480. For efficiency reasons, the images are subsampled to half the number of pixels on each side. A threshold value of 20 is used to obtain the difference image.

BLOB ANALYSIS

The concept of blobs is not new. In this level, we present a novel approach to track blobs regardless of what they represent. The tracking scheme attempts to describe changes in the difference image in terms of motion of blobs and by allowing blobs to merge, split, appear, and vanish. Robust blob tracking was necessary since the pedestrians level relies solely on information passed from this level.

Blob Extraction

Once a difference image is computed, connected segments of 1's are extracted using border following. Another way to extract connected components is to use the raster scan algorithm. The advantage of the latter method is that it extracts holes inside the blobs while border following does not. However, for the purpose of our system, holes do not constitute a major issue. Moving pedestrians usually form solid blobs in the difference image and if these blobs have holes, they may be still considered part of the pedestrian. Border following has the extra advantage of being more efficient. While raster scan algorithm has to traverse every pixel in the image, with border following, the interior of blobs does not need to be considered. Thus, the larger the total area of all the blobs in the image, the faster the segmentation process becomes. The following parameters are computed for each blob b:

1. Perimeter,

2. Area, denoted by $A(b)$: the number of pixels inside the blob,

3. Bounding box: the smallest rectangle surrounding the blob,

4. Density, denoted by $D(b)$: $A(b)$ / Bounding box area,

5. Velocity, denoted by $\mathbf{V}(b)$, calculated in pixels per second in horizontal and vertical directions.

Blob Tracking

When a new set of blobs is computed for frame i, an association with frame $(i-1)$'s set of blobs is sought. Ideally, this association can be an unrestricted relation. With each new frame, blobs can split, merge, appear or disappear. The relation among blobs can be represented by an undirected bipartite graph, $G_i(V_i, E_i)$, where $V_i = B_i \cup B_{i-1}$. B_i and B_{i-1} are the sets of vertices associated with the blobs in frames i and $i-1$, respectively. We will refer to this graph as a *blob graph*. Since there is a one-to-one correspondence between the blobs in frame i and the elements of B_i, we will use the terms blob and vertex interchangeably. Figure 7 shows how the blobs in two consecutive frames are associated. The blob graph in the figure expresses the fact that blob 1 split into blobs 4 and 5, blob 2 and part of blob 1 merged to form blob 4, blob 3 disappeared, and blob 6 appeared.

The process of blob tracking is equivalent to computing G_i for $i = 1, 2, \ldots, n$, where n is the total number of frames. Let $N_i(u)$ denote the set of neighbors of vertex $u \in V_i$, $N_i(u) = \{v | (u, v) \in E_i\}$. To simplify graph computation, we will restrict the generality of the graph to those graphs which do not have more than one vertex of degree more than one in every

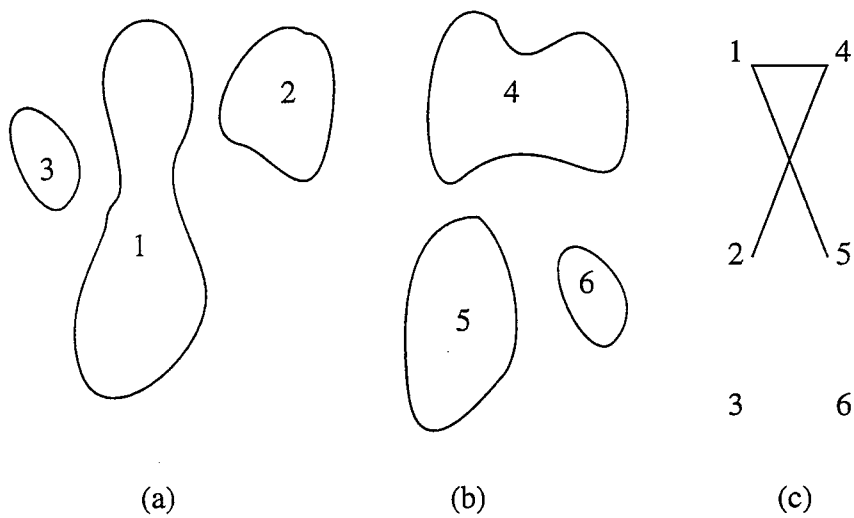


Figure 7. (a) Blobs in frame $(i-1)$. (b) Blobs in frame i. (c) Relationship among blobs.

connected component of the graph. Mathematically, this can be expressed as:

$$\forall (u, v) \in E_i, \left|N_i(u)\right| > 1 \Rightarrow \left|N_i(v)\right| = 1 . \tag{1}$$

This is equivalent to saying that from one frame to the next, a blob may not participate in a splitting and a merging at the same time. We refer to this as the *parent structure constraint*. According to this constraint, the graph in figure 7(c) is invalid. If, however, we eliminate the arc between 1 and 5 or the arc between 2 and 4, it will be a valid graph. This restriction is reasonable assuming a high frame rate where such simultaneous split and merge occurrences are rare.

There are exponentially many ways a general bipartite graph can be constructed. In fact, given two sets of vertices of sizes m and n, there are 2^{mn} different possible graphs. This number is reduced with the parent structure constraint but still remains exponential. To further reduce this number, we use another constraint which we call the *locality constraint*. With this constraint, vertices can be connected only if their corresponding blobs have a bounding box overlap area which is at least half the size of the bounding box of the smaller blob. This constraint, which signifi-

cantly reduces possible graphs, relies on the assumption that a blob is not expected to be too far from where it was in the previous frame. This is also reasonable to assume if we have a relatively high frame rate. We refer to a graph which satisfies both the parent structure and locality constraints as a *valid* graph.

To find the optimum G_i, we need to define a cost function, $C(G_i)$, so that different graphs can be compared. A graph with no edges, i.e. $E_i = \varnothing$, is one extreme solution in which all blobs in V_{i-1} disappear and all blobs in V_i appear. This solution has no association among blobs and should therefore have a high cost. In order to proceed with our formulation of the cost function, we define two disjoint sets, which we call *parents*, P_i, and *descendents*, D_i, whose union is V_i such that $D_i = \bigcup_{u \in P_i} N_i(u)$. P can be easily constructed by selecting from V_i all vertices of degree more than one, all vertices of degree zero, and all vertices of degree one which are only in B_i. Furthermore, let $S_i(u) = \sum_{v \in N_i(u)} A(v)$ be the total area occupied by the neighbors of u. We now write the formula for the cost function as

$$C(G_i) = \sum_{u \in P_i} \frac{|A(u) - S_i(u)|^2}{max(A(u), S_i(u))}. \tag{2}$$

This function favors associations in which blobs do not change much in size. It also favors changes in large blobs' sizes to changes in small blobs' sizes which is intuitively reasonable.

Using this cost function, we can proceed to compute the optimum graph. First, we notice that given a valid graph $G(V, E)$ and two vertices $u, v \in V$, such that $(u, v) \notin E$, the graph $G'(V, E \cup \{(u, v), (v, u)\})$ has a lower cost than G provided that G' is a valid graph. If it is not

possible to find such a G', we call G *dense*. Using this property, we can avoid some useless enumeration of graphs which are not dense. In fact, this observation is the basis of our algorithm to compute the optimum G.

Our algorithm to compute the optimum graph works as follows: A graph G is constructed such that the addition of any edge to G makes it violate the locality constraint. There can be only one such graph. Note that G may violate the parent structure constraints at this moment. The next step in our algorithm systematically eliminates just enough edges from G to make it satisfy the parent structure constraint. The resulting graph is valid and also dense. The process is repeated so that all possible dense graphs are generated. The optimum graph is the one with the minimum cost.

At the end of this stage, we use a simple method to calculate the velocity of each blob, v, based on the velocities of the blobs at the previous stage and the computed blob graph. The blob velocity will be used to initialize pedestrian models as described later. If v is the outcome of a splitting operation, it will be assigned the same velocity as the parent blob. If v is the outcome of a merging operation, it will be assigned the velocity of the biggest child blob. If v is a new blob, it will be assigned zero velocity. Finally, if there is only one blob, u, related to v, the velocity is computed as

$$\mathbf{V}(v) = \beta\frac{(\mathbf{b}_v - \mathbf{b}_u)}{\delta t} + (1-\beta)\mathbf{V}(u) \tag{3}$$

where $\mathbf{b}_v$ and $\mathbf{b}_u$ are the centers of the bounding boxes of v and u, respectively, β is a weight factor set to 0.5, and δt is the sampling interval since the last stage.

CHAPTER 6

MODELLING PEDESTRIANS

INTRODUCTION

Our pedestrian tracking scheme has three levels. The first level is the images level, the second level is the blob level and the final level in our hierarchy is the pedestrians level (please see previous report for detailed description of these levels). In all these levels, filtering is essential. Filtering incorporates our understanding of the model and smooths the input data. This report will describe in detail the filtering schemes we tried. Let us focus our attention to the pedestrians level. The input to this level is tracked blobs and the output is the spatio-temporal coordinates of each pedestrian. The relationship between pedestrians and blobs in the image is not necessarily one-to-one. A pedestrian wearing clothes which are close in color to the background may show up as more than one blob. Partially occluded pedestrians may also result in more than one blob or even in no blobs at all if the pedestrian is fully occluded. Two or more pedestrians walking close to each other may give rise to a single blob. For this reason, it was necessary to make the pedestrians level capable of handling all the above cases. We do this by modeling the pedestrian as a rectangular patch with a certain dynamic behavior (this is combined with a sophisticated filtering scheme). We found that for the purpose of tracking, this simple model adequately resembles the pedestrian shape and motion dynamics. We now present this model in more detail and then describe how tracking is performed.

PEDESTRIAN MODEL

Pedestrians usually walk with a constant speed. Moreover, the speed of a pedestrian usually changes gradually when the pedestrian desires to stop or start walking. Three different approaches for pedestrian modeling have been attempted. The latter two are based on the assumption that the scene has a flat ground. This assumption is necessary so that back projection from the scene to the image plane can be performed with the knowledge of the camera geometry to determine the expected dimensions and dynamic behavior of the pedestrian in the image coordinate system. Small variations in ground elevation will still be tolerated especially in distant areas. This restriction can be removed if the scene topology can be determined *a priori*.

1. 2-D dynamics and 2-D shape:

   The pedestrian is modeled as a fixed size rectangular patch whose dimensions are similar to the projection of the dimensions of an average size pedestrian located somewhere near the middle of the scene. The patch is assumed to move with a constant velocity in the image coordinate system.

2. 2-D dynamics and 3-D shape:

   The pedestrian is modeled as a rectangular patch whose dimensions depend on its location in the image. The dimensions are equal to the projection of the dimensions of an average size pedestrian at the corresponding location in the scene. As in the first approach, the patch is assumed to move with a constant velocity in the image coordinate system.

3. 3-D dynamics and 3-D shape:

   The rectangular patch dimensions are as in the previous approach but the patch is assumed to move with constant velocity in the scene coordinate system.

In all these approaches, the patch acceleration is modeled as zero-mean, Gaussian noise to accom-

modate for changes in velocity. Given a sampling interval δt, the discrete-time dynamic system for the pedestrian model can be described by the following equation:

$$\mathbf{x}_{t+1} = \mathbf{F}\mathbf{x}_t + \mathbf{v}_t, \tag{4}$$

where $\mathbf{x} = \begin{bmatrix} x & \dot{x} & y & \dot{y} \end{bmatrix}^T$ is the state vector consisting of the pedestrian location, (x, y) and velocity, $(\dot{x}, \dot{y})$, $\mathbf{F}$ is the transition matrix of the system given by $\begin{bmatrix} 1 & \delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}$, and $\mathbf{v}_t$ is a sequence of zero-mean, white, Gaussian process noise with covariance matrix $\mathbf{Q}$. In the first two approaches above, (x, y) is the image coordinates of the bottom left corner of the patch, while in the third approach, x is the ground distance between the center line of the camera and the left side of the patch, and y is the ground distance between the camera optical center and the patch. $\mathbf{Q}$ is computed as $\begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix} q$ where $\mathbf{A} = \begin{bmatrix} \frac{(\delta t)^3}{3} & \frac{(\delta t)^2}{2} \\ \frac{(\delta t)^2}{2} & \delta t \end{bmatrix}$ and q represents the variance of the acceleration.

PEDESTRIAN TRACKING

Tracking pedestrians depends on the current state of pedestrians as well as the input to pedestrians level which is the tracked blobs. In our system, we use Kalman filtering (KF) in the case of the first two modeling approaches and extended Kalman filtering (EKF) in the case of the third approach to estimate pedestrian parameters. We maintain a many-to-many relationship between pedestrians and blobs and then use it to provide measurements to the filter. The next five sections

describe one tracking cycle.

Relating Pedestrians to Blobs

We represent the relationship between pedestrians and blobs as a directed bipartite graph, $GP_i(VP_i, EP_i)$, where $VP_i = B_i \cup P$. B_i is the set of blobs computed from the ith image. P is the set of pedestrians. An edge (p, u), $p \in P$ and $u \in B_i$ denotes that blob u participates in pedestrian p. We call GP_i a pedestrian graph. Given a blob graph, $G_i(V_i, E_i)$, and a pedestrian graph, GP_{i-1}, EP_i is computed as follows:

$$EP_i = \{(p, u) | (u, v) \in E_i \wedge (p, v) \in EP_{i-1}\}. \tag{5}$$

In other words, if a pedestrian was related to a blob in frame $(i-1)$ and that blob is related to another blob in the ith frame (through a split, merge, etc.), then the pedestrian is also related to the latter blob.

Prediction

Given the system equation above, the prediction phase of the Kalman filter is given by the following equations:

$$\begin{aligned} \hat{\mathbf{x}}_{t+1} &= \mathbf{F}\mathbf{x}_t, \\ \hat{\mathbf{P}}_{t+1} &= \mathbf{F}\mathbf{P}_t\mathbf{F}^T + \mathbf{Q}. \end{aligned} \tag{6}$$

Here, $\hat{\mathbf{x}}$ and $\hat{\mathbf{P}}$ are the predicted state vector and state error covariance matrix, respectively. $\mathbf{x}$ and $\mathbf{P}$ are the previously estimated state vector and state error covariance matrix.

Calculating Pedestrian Positions

In this step, we use the predicted pedestrian locations as starting positions and we apply the following rule to update the locations of pedestrians: Move each pedestrian, p, as little as possible so

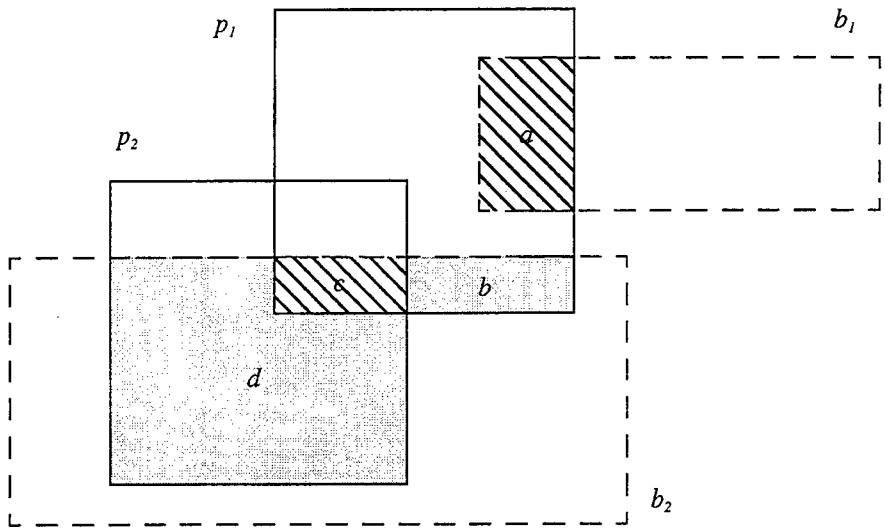


Figure 8. Overlap area. Pedestrians p_1 and p_2 share blob b_2 while b_1 is only part of p_1 (See text for overlap area computation).

that it covers as much as possible of its blobs, $\{u|(p, u) \in EP_i\}$; and if a number of pedestrians share some blobs, they should all participate in covering all these blobs. There are many vague terms in this rule which need to be specified. First of all, the amount by which a pedestrian covers a blob implies a measure of overlap area. We have already used the bounding box as a shape representation of blobs. However, since the blob bounding box area may be quite different from the actual blob area, we will include the blob density in the computation of pedestrian-blob overlap area. Let $BB(p)$ be the bounding box of a pedestrian p, and $BB(b)$ be the bounding box of a blob, b. The intersection of $BB(p)$ and $BB(b)$ is a rectangle which we denote its area as $X(BB(p), BB(b))$. The overlap area between p and b is computed as $X(BB(p), BB(b)) \times D(b)$. When more than one pedestrian share a blob, the overlap area is computed this way for each pedestrian only if the other pedestrians do not also overlap the intersection area. If they did, that particular overlap area is divided by the square of the number of pedestrians whose boxes overlap the area. Figure 8 illustrates this situation. The overlap area for p_1 is com-

puted as $a \times D(b_1) + b \times D(b_2) + \frac{c \times D(b_2)}{2^2}$. For p_2, the overlap area is $d \times D(b_2) + \frac{c \times D(b_2)}{2^2}$.

The problem of finding the optimum locations of pedestrians can be stated in terms of the overlap area measure that we just defined. We would like to place pedestrians such that the total overlap area of each pedestrian is maximized. We restate the optimization problem as the problem of finding the minimum total overlap area arrangement of pedestrians which has the least sum of square distances between old and new locations of the pedestrian. We do not attempt to solve the problem optimally because of its complexity. Instead, we resort to a heuristic solution using relaxation. First, a small step size is chosen. Then, each pedestrian is moved in all possible directions by the step size and the location which minimizes the overlap area is recorded. Pedestrian locations are then updated according to the recorded locations. This completes one iteration. In each following iteration, the step size is increased. In our implementation, we start with a step of one pixel and double the step size in each iteration until a maximum of 32 pixels.

The resulting locations form the measurements that will be fed back into the KF or the EKF to produce the new state estimates. Moreover, we use the overlap area to provide feedback about the measurement confidence by setting the measurement error standard deviation, which is described below, to be inversely proportional to the ratio of the overlap area to the pedestrian area. That is, the smaller the overlap area, the less confident the measurement is considered.

Estimation

A measurement is a location in the image coordinate system as computed in the previous section, $\mathbf{z} = \begin{bmatrix} u & v \end{bmatrix}^T$. Measurements are related to the state vector by

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t) + \mathbf{w}_t, \tag{7}$$

where $\mathbf{h}$ is the measurement function and $\mathbf{w}_t$ is a sequence of zero-mean, white, Gaussian measurement noise with covariance $\mathbf{R}_t$ given by $\begin{bmatrix} \sigma_t^2 & 0 \\ 0 & \sigma_t^2 \end{bmatrix}$. The measurement error standard deviation, σ_t, depends on the overlap area computed in the previous section. In the case of 2-D dynamics modeling approaches in which pedestrian locations are in image coordinates, $\mathbf{h}$ is a linear function given by

$$\mathbf{h}(\mathbf{x}) = \mathbf{H}\mathbf{x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{x}. \tag{8}$$

In this case, the state estimation equations for the Kalman filter are

$$\begin{aligned} \mathbf{K}_{t+1} &= \hat{\mathbf{P}}_{t+1}\mathbf{H}^T(\mathbf{H}\hat{\mathbf{P}}_{t+1}\mathbf{H}^T + \mathbf{R}_t)^{-1}, \\ \mathbf{x}_{t+1} &= \hat{\mathbf{x}}_{t+1} + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{H}\hat{\mathbf{x}}_{t+1}), \\ \mathbf{P}_{t+1} &= (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})\hat{\mathbf{P}}_{t+1}, \end{aligned} \tag{9}$$

where $\mathbf{K}_{t+1}$ is the Kalman gain at $t+1$. In the third modeling approach, however, pedestrian locations are expressed in world coordinates resulting in $\mathbf{h}$ being a non-linear function which performs projection into image coordinates. In this case, the extended Kalman filter was used. We let $\mathbf{H}$ be the Jacobian of $\mathbf{h}$. The EKF state estimation equations become

$$\begin{aligned} \mathbf{K}_{t+1} &= \hat{\mathbf{P}}_{t+1}\mathbf{H}^T(\mathbf{H}\hat{\mathbf{P}}_{t+1}\mathbf{H}^T + \mathbf{R}_t)^{-1}, \\ \mathbf{x}_{t+1} &= \hat{\mathbf{x}}_{t+1} + \mathbf{K}_{t+1}(\mathbf{z}_{t+1} - \mathbf{h}(\hat{\mathbf{x}}_{t+1})), \\ \mathbf{P}_{t+1} &= (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})\hat{\mathbf{P}}_{t+1}. \end{aligned} \tag{10}$$

The estimated state vector $\mathbf{x}_{t+1}$ is the outcome of the pedestrians level. When using a 3-D shape modeling technique we also compute the dimensions of the pedestrian patch based on the

estimated pedestrian location.

REFINEMENT

At the end of this stage, we perform some checks to refine the pedestrian-blob relationships since pedestrians have been relocated. This step improves the performance of filtering. These steps can be summarized as follows:

a. If the overlap area between a pedestrian and one of its blobs becomes less than 10% of the size of both, it will no longer be considered belonging to this pedestrian. This serves as the splitting procedure when two pedestrians walk past each other.

b. If the overlap area between a pedestrian and a blob that does not belong to any pedestrian becomes more than 10% of the size of either one, the blob will be added to the pedestrian blobs. This makes the pedestrian re-acquire some blobs that may have disappeared due to occlusion.

c. Select one of the blobs that do not belong to any pedestrians. If the area of this blob is large enough (1000 pixels in our implementation), create a new pedestrian centered around the bounding box of the blob with initial velocity equal to that of the blob. This step actually serves as the initialization stage.

d. Select one of the blobs which is already assigned one or more pedestrians but can accommodate more pedestrian patches. Create a new pedestrian for this blob as in c. This handles cases in which a group of people form one big blob which does not split. If we do not do this step, only one pedestrian would be assigned to this blob.

CHAPTER 7

HANDLING CROWDS

INTRODUCTION

Some pedestrian intersections are naturally much more crowded that others. A system that tracks pedestrians at intersections should be able to adapt to this parameter for successful operation. In this report we illustrate the problem of crowded intersections and show how our system performs in these situations. We also present some solutions to problems with our system.

LIMITATION OF THE PEDESTRIAN TRACKING SYSTEM

Recently, we have been able to successfully track an arbitrary number of pedestrians simultaneously. However, as the number of pedestrians increase, system performance degrades due to the limited processing power. When the number of pedestrians exceeds 12, the processing rate reaches below 3 frames per second. This makes it hard to keep accurate track of every pedestrian because of the large displacement among pedestrian locations in the processed frames.

TESTING ON SAMPLE VIDEO SEQUENCES OF CROWDED INTERSECTIONS

Sample video sequences of crowded intersections were collected in order to perform extensive testing of the system. Figures 9-15 show several snapshots taken from different sequences. Results from the tests can be summarized in the following.

1. **Difficulty of obtaining a background image.**

   One of the system requirements is an empty background scene which does not contain any pedestrians or vehicles. In a crowded intersection, it is almost always the case that there are pedestrians or vehicles present. A solution to this problem is given below.

2. **Interference of vehicles.**

   Intersections crowded with pedestrians are usually crowded with vehicles as well. Our system was designed to track objects that can be modeled as pedestrians and therefore expects only pedestrians in the scene. This is easily solved by manually selecting the crosswalk area of the intersection where only pedestrians are expects to be present. However, as in Figure 11, vehicles may occlude pedestrians even in these areas.

3. **Inability to isolate individuals.**

   The primary problem when dealing with crowded intersection is the difficulty associated with isolating individual pedestrians. Occlusions (Figure 14) and camera distance (Figure 15) are the main factors contributing to this problem. It is often difficult even for the human eye to isolate pedestrians. A solution to this problem is given below.

4. **System performance.**

   This is a direct consequence of the previous problem. As the system tries to isolate pedestrians, it may end up with a large number causing a degraded performance. This is due to limited processing power.

5. **Indefinite waiting.**

   Another related problem is that in case where the presence of pedestrians is to be used to control the walk signal, it is desired not to keep the walk signal for ever. The flow of pedestrians in a crowded intersection can last for a long time.

POSSIBLE SOLUTIONS

1. **Difficulty of obtaining a background image.**

   Instead of trying to obtain an empty background image, adaptive update methods can be used here. By averaging a history of image sequence over time, an image very close to the empty background image can be obtained. This will require a higher detection threshold in order to ignore false targets.

1. **Interference of vehicles.**

   When a vehicle such as a bus occludes a crosswalk, tracking is temporarily lost. This problem can be solved by setting up the camera in such a way that vehicles cannot be present between the camera and the pedestrians.

2. **Inability to isolate individuals.**

   When the objective is to check for the existence pedestrians (either in the crosswalk or waiting to cross), isolating individual pedestrians becomes unnecessary. Instead, we can try to detect pedestrians and track them as a group rather than individuals. This is why we introduced the crowd mode. Whenever the system performance degrades, it automatically switches to crowd mode where it no longer attempts to track individual pedestrians but groups of people.

3. **System performance.**

   If we adopt the solution above, system performance is automatically improved.

4. **Indefinite waiting.**

   This is more of a traffic optimization problem. An obvious solution is to have a maximum time limit for the walk signal beyond which the presence of pedestrians has no effect on the controller.

Figure 9. Intersection 1 at downtown Minneapolis.

Figure 10. Intersection 2 at downtown Minneapolis.

Figure 11. Intersection 3 at downtown Minneapolis.

Figure 12. Intersection 4 at downtown Minneapolis.

Figure 13. Intersection 5 at downtown Minneapolis.

Figure 14. Intersection 6 at downtown Minneapolis.

Figure 15. Intersection 7 at downtown Minneapolis.

CHAPTER 8

RESULTS AND CONCLUSIONS

EXPERIMENTAL RESULTS

The three modeling approaches were tested. The 3-D shape models performed noticeably better than the 2-D shape model. The 3-D dynamics model, however, only slightly outperformed the 2-D dynamics model. The system was tested on several indoor and outdoor image sequences. Several outdoor sequences in different weather conditions (sunny, cloudy, snow, etc.) have been used. In most cases, pedestrians were tracked correctly throughout the period they appeared in the scene. Scenarios included pedestrians moving at a slow or very high speeds, partial and full occlusions, bicycles, and several pedestrian interactions. Interactions between pedestrians included occlusion of one another, repeated merging and splitting of blobs corresponding to two or more pedestrians walking together, pedestrians walking past each other, and pedestrians meeting and then walking back in the direction they came from. The system has a peak performance of over 20 frames per second. In a relatively cluttered image with about 6 pedestrians, the frame processing rate dropped down to about 14 frames per second.

Figure 16 shows 16 snapshots spanning a sequence of 8.4 seconds. The sequence demonstrates the system behavior against occlusions, both partial and full. Figure 17 shows 12 snapshots from a scene under different weather conditions. The snapshots span a sequence of 35 seconds. We also performed a pedestrian counting experiment for a sequence of 12 minutes in which 124 pedestri-

ans were counted manually. The system gave a count of 130 making it successful by over 95%. Most of the failures were due to bicyclists who were double counted because the blob they generated was closer to the size of two pedestrians.

There are other cases where the system failed. Those include highly crowded images. Other inevitable failures occur when a pedestrian is almost similar in color to the background. In this case, if a pedestrian box is tracking this pedestrian, it will be prone to clamp to other nearby pedestrians having bigger blobs. Also, when a pedestrian becomes totally occluded but then reappears at an unexpected location, the pedestrian box will loose track. Finally, in cases where two pedestrians walk very closely around each other, their pedestrian boxes may get interchanged erroneously.

CONCLUSIONS AND FUTURE RESEARCH

We presented a real-time model-based pedestrian tracking system capable of working robustly under many difficult circumstances such as occlusions and ambiguities. For each pedestrian in the view of the camera, the system produces location and velocity information as long as the pedestrian is visible. This data can be used by a scheduling algorithm to control walk signals at an intersection in order to increase the safety and efficiency of existing traffic systems.

There are several issues that still need to be addressed. Spatial interpretation of blobs is one such issue. In the current system, the only spatial attribute of blobs taken into consideration is the blob area. The shape of the blob can give a good clue on its contents. Since the camera is mounted in a fixed location, use of a priori known scene topology is another issue that can be considered. Although blobs obtained from difference images can be sufficient to decide the location of pedes-

trians in many cases, the intensity information may be useful to resolve certain ambiguities. The use of such information in the form of statistical distribution of intensities may add to the robustness of the current system and is well worth pursuing.

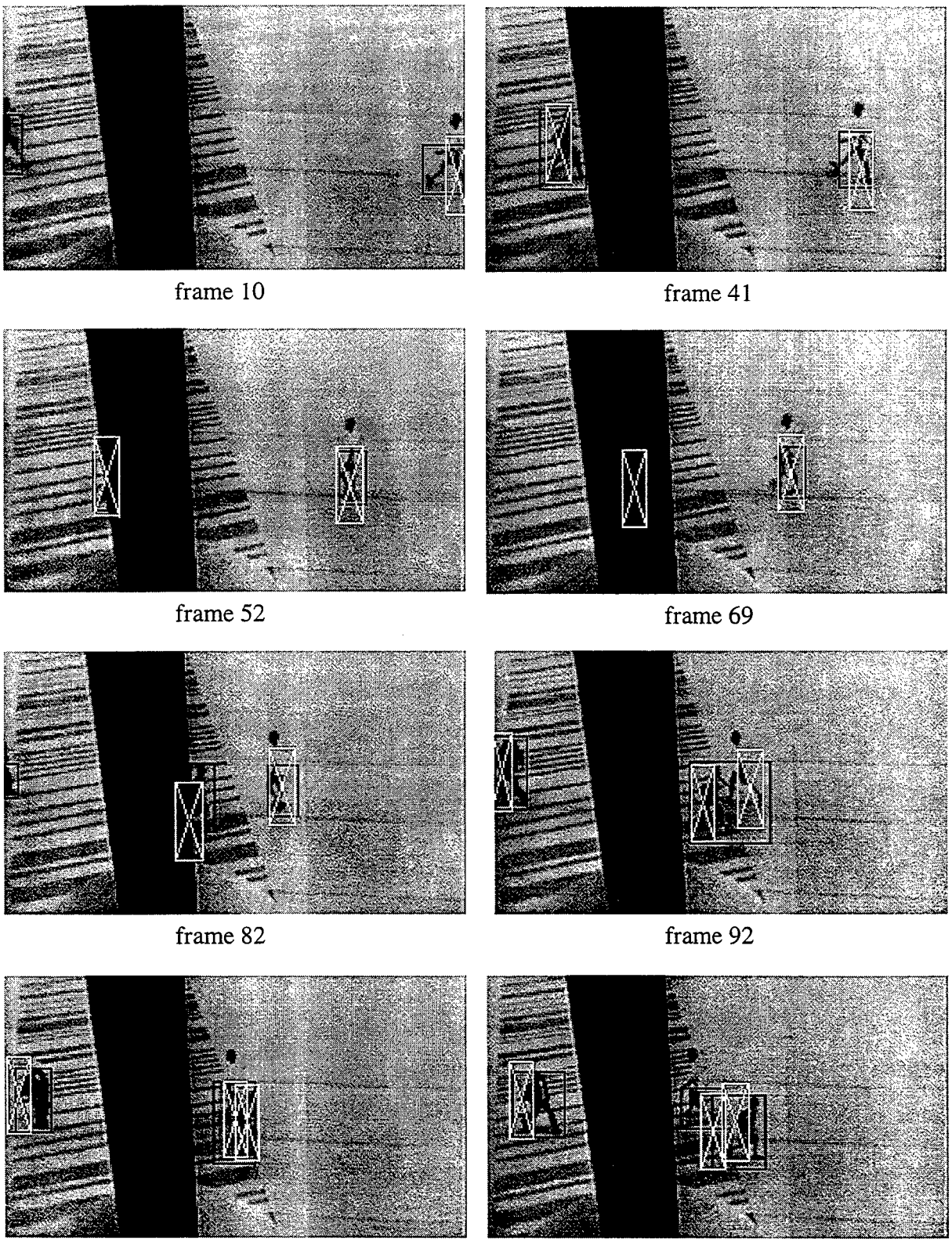
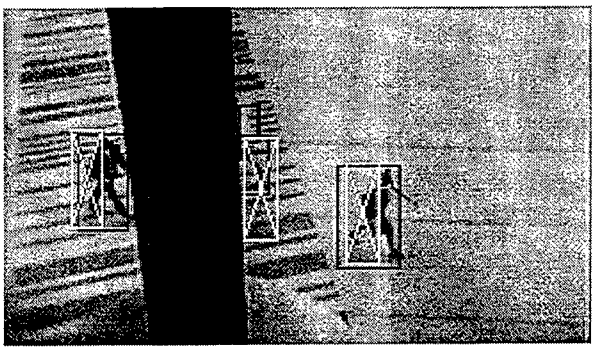
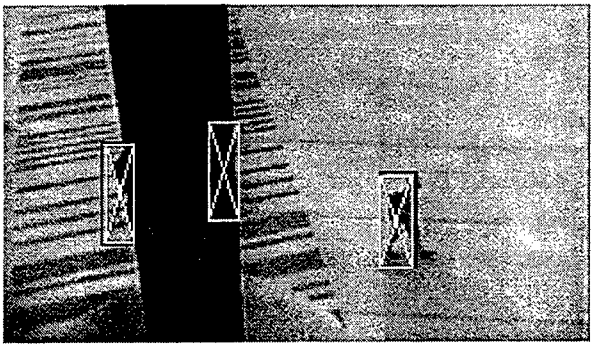


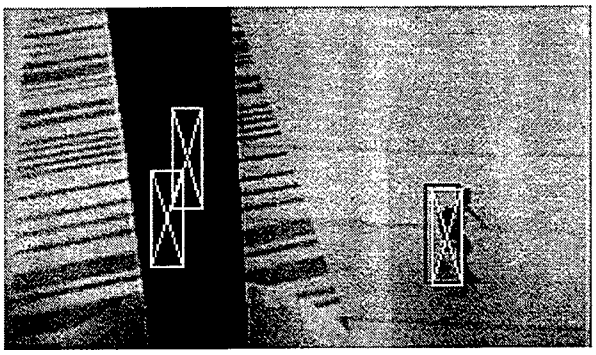
Figure 16. A number of snapshots from the input sequence overlaid with pedestrian boxes shown in white and blob boxes shown in black.



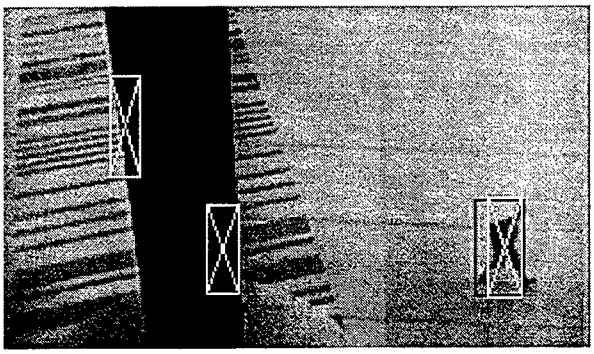
frame 120



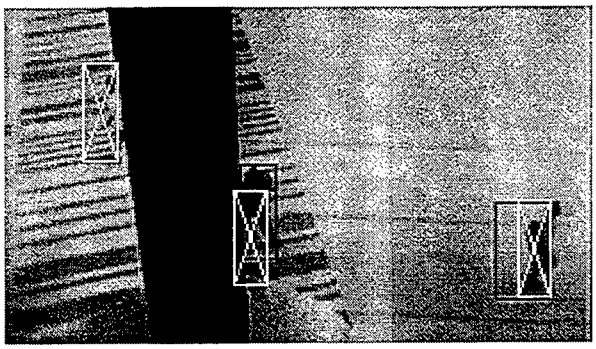
frame 129



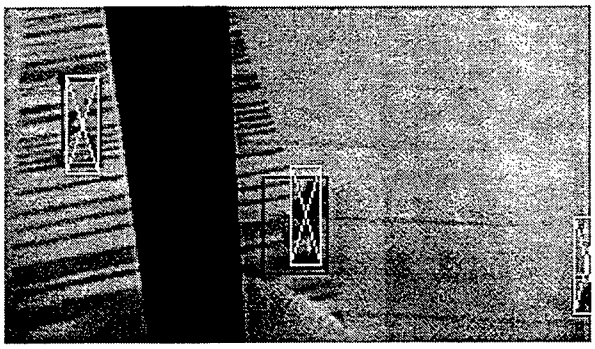
frame 141



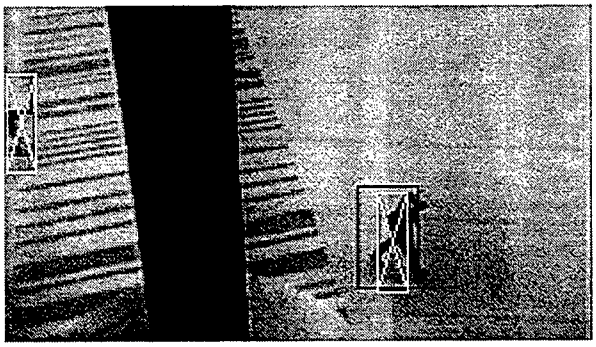
frame 156



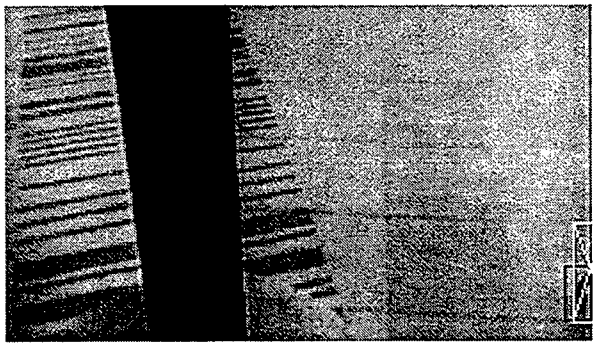
frame 164



frame 182



frame 206



frame 262

Figure 16. (continued).

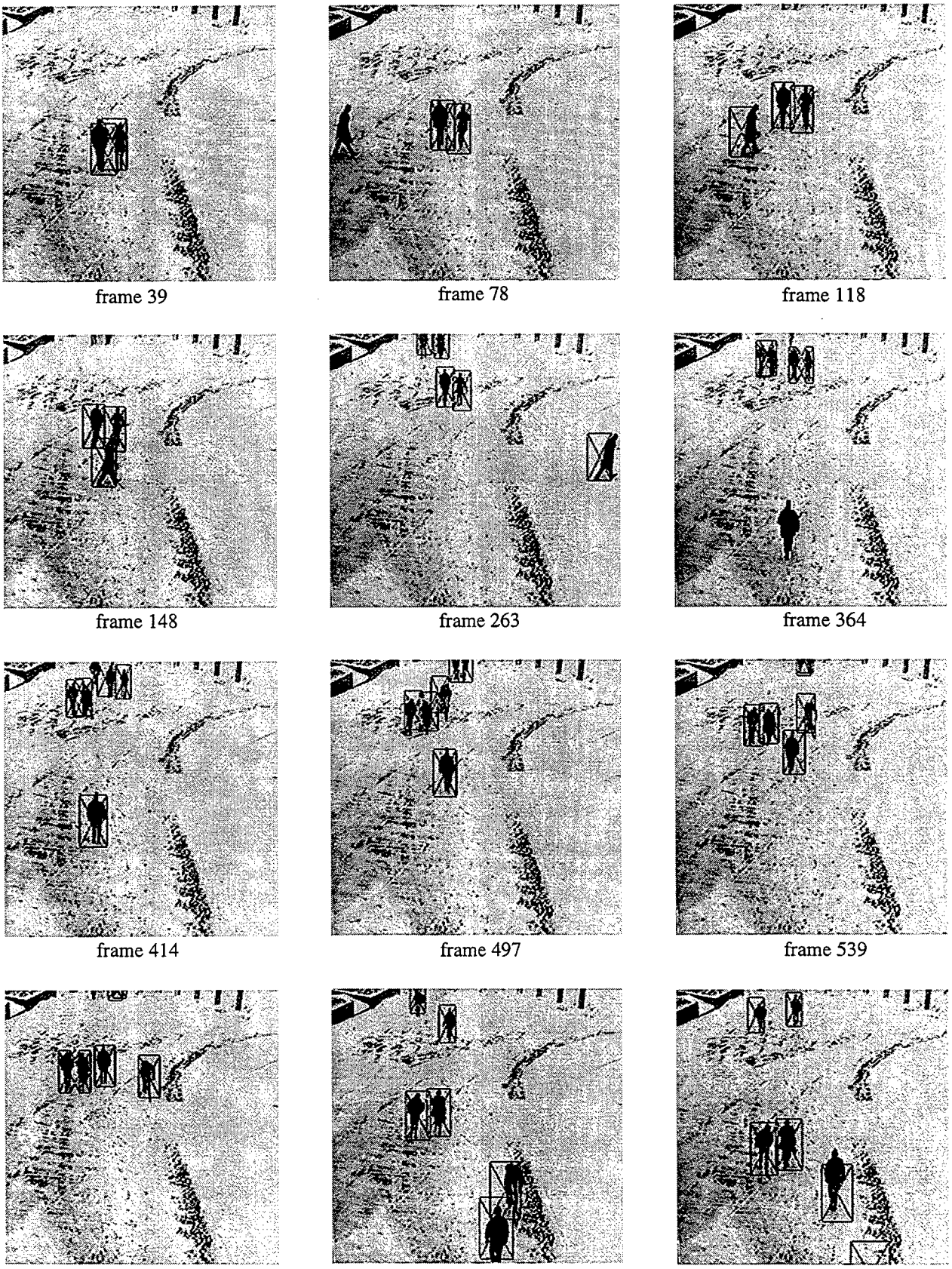


Figure 17. A number of snapshots from the input sequence in a snowy afternoon overlaid with pedestrian boxes shown in black.

REFERENCES

1. Allen, P.K., Timcenko, A.,Yoshimi, B., and Michelman, P., "Automated tracking and grasping of a moving object with a robotic hand-eye system," IEEE Trans. Robotics and Automation 9(2):152-165, 1993.

2. Allen, P.K.,Yoshimi, B., and Timcenko, A., "Real-time visual servoing," In Proc. IEEE International Conference on Robotics and Automation, pp. 851-856. April, 1991.

3. Anandan, P., "A computational framework and an algorithm for the measurement of visual motion," International Journal of Computer Vision 2(3):283-310, 1988.

4. Anderson, C.H., Burt P.J., and Van der Wal, G.S., "Change detection and tracking using pyramid transform techniques," In Proc. SPIE Intelligent Robots and Computer Vision, pp. 72-78. 1985.

5. Athans, M., Systems, Networks, and Computation: Multivariable Methods, McGraw-Hill, New York, 1974.

6. Brown, C.M., "Centralized and decentralized Kalman filter techniques for tracking, navigation, and control," In Proc. DARPA Image Understanding Workshop, pp. 651-675. 1989.

7. Burt, P.J., and Adelson, E.H., "A multiresolution spline with applications to image mosaics," ACM Trans. Communications 2:217-236, 1983.

8. ---, "The Laplacian pyramid as a compact image code," IEEE Trans. Communications 31(4):532-540, April, 1983.

9. Burt, P.J., Hong, T.H., and Rosenfeld, A., "Image segmentation and region property computation by cooperative hierarchical computation," IEEE Trans. Systems, Man, and Cybernetics 11:802-809, 1981.

10. Dickmanns, E.D., Mysliwetz, B., and Christians, T., "An integrated spatio-temporal approach to automatic visual guidance of autonomous vehicles," IEEE Trans. Systems, Man, and Cybernetics 20(6):1273-1284, November, 1990.

11. Dickmanns, E.D., and A. Zapp, A., "Autonomous high speed road vehicle guidance by computer vision," In Proc. 10th IFAC World Congress, July, 1987.

12. Dinstein, I., "A new technique for visual motion alarm," Pattern Recognition Letters 8(5):347-351, December, 1988.

13. Feddema, J.T. and Mitchell, O.R., "Vision guided servoing with feature-based trajectory generation," IEEE Trans. Robotics and Automation 5(5):691-699, 1989.

14. Horn, B.K.P., Robot vision., MIT Press, Cambridge, 1986.

15. Houghton, A., Hobson, G.S., Seed, L., and Tozer, R.C., "Automatic monitoring of vehicles at road junctions," Traffic Engineering Control 28(10):441-453, October, 1987.

16. Inigo, R.M., "Application of machine vision to traffic monitoring and control," IEEE Trans.

Vehicular Technology 38(3):112-122, August, 1989.

17. Jain, R.C., "Segmentation of frame sequences obtained by a moving observer," IEEE Trans. Pattern Analysis and Machine Intelligence 6(5):624-629, 1984.

18. Kass, B., Witkin, A., and Terzopoulos, D., "Snakes: Active contour models," International Journal of Computer Vision 1(4):321-331, 1987.

19. Kehtarnavaz, N., Griswold, N.C., and Lee, J.S., "Visual control of an autonomous vehicle (BART)—the vehicle-following problem," IEEE Trans. Vehicular Technology 40(3):654-662, August, 1991.

20. Kilger, M., "A shadow handler in a video-based real-time traffic monitoring system," In Proc. IEEE Workshop on Applications of Computer Vision, pp. 11-18. 1992.

21. Koivo, A.J., and Houshangi, N., "Real-time vision feedback for servoing of a robotic manipulator with self-tuning controller," IEEE Trans. Systems, Man and Cybernetics 21(1):134-142, 1991.

22. Michalopoulos, P.G., "Vehicle detection video through image processing: the autoscope system," IEEE Trans. Vehicular Technology 40(1):21-29, February, 1991.

23. Mori, H., Charkari, N.M., and Matsushita, T., "On-line vehicle and pedestrian detections based on sign pattern," IEEE Trans. Industrial Electronics 41(4):384-391, August, 1994.

24. Nelson, R.C., "Qualitative detection of motion by a moving observer," International Journal of Computer Vision 7(1):33-46, 1991.

25. ---, "Qualitative detection of motion by a moving observer," In Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp. 173-178. 1991.

26. Papanikolopoulos, N.P., "Controlled active vision," Ph.D. Thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1992.

27. Pellerin., C., "Machine vision for smart highways," Sensor Review 12(1):26-27, 1992.

28. Smith, C.E., Brandt, S.A., and Papanikolopoulos, N.P., "Controlled active exploration of uncalibrated environments," In Proc. IEEE Conference on Computer Vision and Pattern Recognition, pp. 792-795. 1994.

29. Smith, C.E., Brandt, S.A., Richards, C.A., and Papanikolopoulos, N.P., "Visual tracking for intelligent vehicle-highway systems," Technical Report TR 94-48, University of Minnesota, Department of Computer Science, August, 1994.

30. Stewart, D.B., Schmitz, D.E., and Khosla, P.K., "CHIMERA II: A real-time multiprocessing environment for sensor-based robot control," In Proc. Fourth International Symposium on Intelligent Control, pp. 265-271. September, 1989.

31. Takatoo, M., Kitamura, T., Okuyama, Y., Kobayashi, Y.,Kikuchi, K., Nakanishi, H. and Shibata, T., "Traffic flow measuring system using image processing," In Proc. SPIE, pp. 172-180. 1990.

32. Thompson, W.B., and Pong, T.C., "Detecting Moving Objects," International Journal of

Computer Vision 4(1):39-57, 1990.

33. Thorpe, C., Hebert, M.H., Kanade, T., and Shafer, S.A., "Vision and navigation for the Carnegie-Mellon NAVLAB," IEEE Trans. Pattern Analysis and Machine Intelligence 10(3):362-373, 1988.

34. Tomasi, C., and Kanade, T., "Detection and tracking of point features," Technical Report CMU-CS-91-132, Carnegie Mellon University, School of Computer Science, 1991.

35. Ullman, S., The interpretation of visual motion, MIT Press, Cambridge, 1979.

36. Ulmer, B., "VITA—an autonomous road vehicle (ARV) for collision avoidance in traffic," In Proc. Intelligent Vehicles 1992 Symposium, pp. 36-41. 1992.

37. Van der Wal, G.S., and Burt.,P.J., "A VLSI pyramid chip for multiresolution image analysis," International Journal of Computer Vision 8(3):177-189, 1992.

38. Weiss, L.E., Sanderson, A.C., and Neuman, C.P., "Dynamic sensor-based control of robots with visual feedback.," IEEE Journal of Robotics and Automation RA-3(5):404-417, October, 1987.

39. Zielke, T., Brauckmann, M., and Von Seelen, W., "CARTRACK: computer vision-based car-following" In Proc. IEEE Workshop on Applications of Computer Vision, pp. 156-163. 1992.