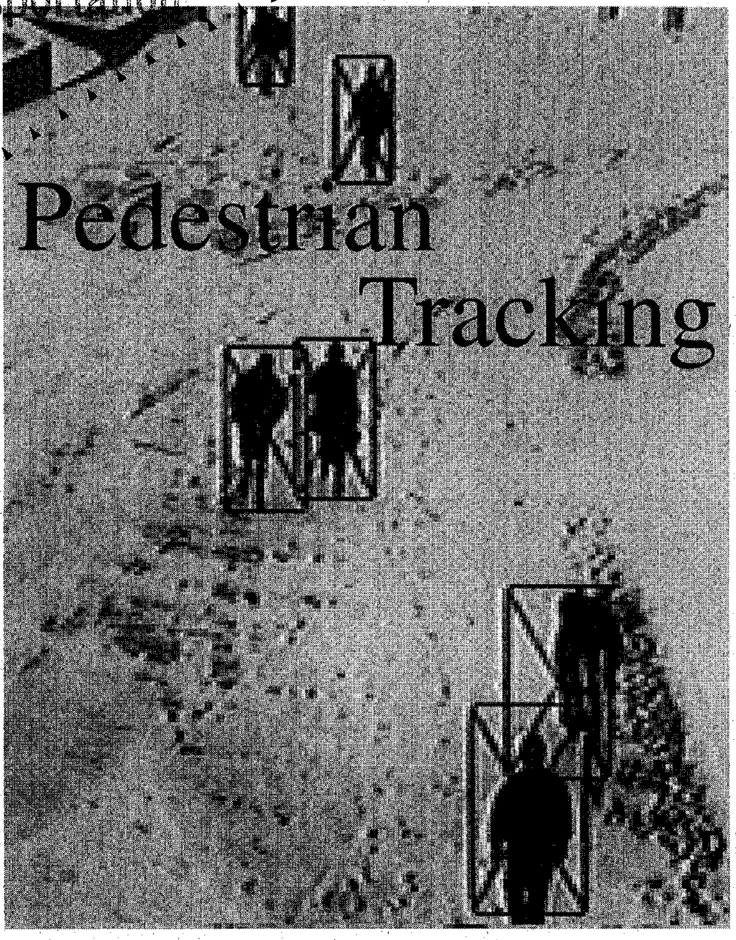
Report Number 97-02



Pedestrian Control at Intersections (Phase I)

UNIVERSITY OF MINNESOTA
CENTER FOR TRANSPORTATION STUDIES

Research

0.1 36752693

Technical Report Documentation Page

1. Report No. MN/RC - 97/02	2.	3. Recipient's Accession No.
4. Title and Subtitle PEDESTRIAN CONTROL AT INTERSECTIONS (PHASE I)		5. Report Date October 1996
		6.
7. Author(s) Nikolaos P. Papanikolopoulos, Osama Masoud and Charles A. Richards		8. Performing Organization Report No.
9. Performing Organization Name and Address Artificial Intelligence, Robotics, and Vision Laboratory Department of Computer Science University of Minnesota Minneapolis, MN 55455		10. Project/Task/Work Unit No.
		11. Contract (C) or Grant (G) No. (C) 72447 TOC #159
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 1994 - 1996
		14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract (Limit: 200 words)

This report describes a real-time system for tracking pedestrians in sequences of grayscale images acquired by a stationary camera. The system outputs the spatio-temporal coordinates of each pedestrian during the period when the pedestrian is visible. Implemented on a Datacube MaxVideo 20 equipped with a Datacube Max 860, the system achieved a peak performance of over 30 framers per second. Experimental results based on indoor and outdoor scenes have shown that the system is robust under many difficult traffic situations.

The system uses the "figure/ground" framework to accomplish the goal of pedestrian detection. The detection phase outputs the tracked blobs (regions), which in turn pass to the final level, the pedestrian level. The pedestrian level deals with pedestrian models and depends on the tracked blobs as the only source of input. By doing this, researchers avoid trying to infer information about pedestrians directly from raw images, a process that is highly sensitive to noise. The pedestrian level makes use of Kalman filtering to predict and estimate pedestrian attributes. The filtered attributes constitute the output of this level, which is the output of the system. This system was designed to be robust to high levels of noise and particularly to deal with difficult situations, such as partial or full occlusions of pedestrians. The report compares vision sensors with other types of possible sensors for the pedestrian control task and evaluates the use of active deformable models as an effective pedestrian tracking module.

17. Document Analysis/Descriptors CCD Cameras Pedestrian Control Computer Vision Techniques	Visual Tracking Visual Detection	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 79	22. Price

Pedestrian Control at Intersections (Phase I)

Final Report

Prepared by

Nikolaos P. Papanikolopoulos
Osama Masoud
Charles A. Richards
Artificial Intelligence, Robotics, and Vision Laboratory
Department of Computer Science
University of Minnesota
Minneapolis, MN 55455

October 1996

Published by

Minnesota Department of Transportation
Office of Research Administration
200 Ford Building Mail Stop 330
117 University Avenue
St Paul Minnesota 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 describes a real-time system for tracking pedestrians in sequences of grayscale images acquired by a stationary camera. The output of the system is the spatio-temporal coordinates of each pedestrian during the period the pedestrian is visible. The system uses the “figure/ground” framework in order accomplish the goal of pedestrian detection. The output of the detection phase is the tracked blobs (regions) which are in turn passed to the final level, the pedestrians level. The pedestrian level deals with pedestrian models and depends on the tracked blobs as the only source of input. By doing this, we avoid trying to infer information about pedestrians directly from raw images, a process that is highly sensitive to noise. The pedestrian level makes use of Kalman filtering to predict and estimate pedestrian attributes. The filtered attributes constitute the output of this level which is the output of the system. Our system was designed to be robust to high levels of noise and particularly to deal with difficult situations such as partial or full occlusions of pedestrians. The report compares vision sensors with other types of possible sensors for the pedestrian control task and evaluates the use of active deformable models as an effective pedestrian tracking module.

The system was implemented on a Datacube MaxVideo 20 equipped with a Datacube Max860 and was able to achieve a peak performance of over 30 frames per second. Experimental results based on indoor and outdoor scenes have shown that the system is robust under many difficult traffic situations.

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

The success of this research will be judged based upon our ability to set forth a complete framework that encompasses each of the required components of a visual detection and tracking system. The generalized structure of our framework is illustrated in (see Figure 1). This structure includes the following components.

- *Visual detection module*: This component provides other components with a set of elementary tokens that represent objects that have been recently detected. During each execution of this module, objects that have been previously detected are "redetected." If the visual tracking module is omitted, then this module is applied to every input frame in the image sequence.

- *Analysis of the detection information*: A variety of transient activity can take place on the tokens provided by the visual detection module. An example of such an activity is the identification of surface features on detected objects of interest.
- *Use of the detection information*: There must exist some form of interface between the analyzed object information and the other computer systems involved. In the case of some computer systems, this component can result in a change in the sensed environment (e.g., transportation sensing systems).
- *Visual tracking module*: Rather than performing visual detection on each input frame, we can optionally choose to perform detection only occasionally. In between

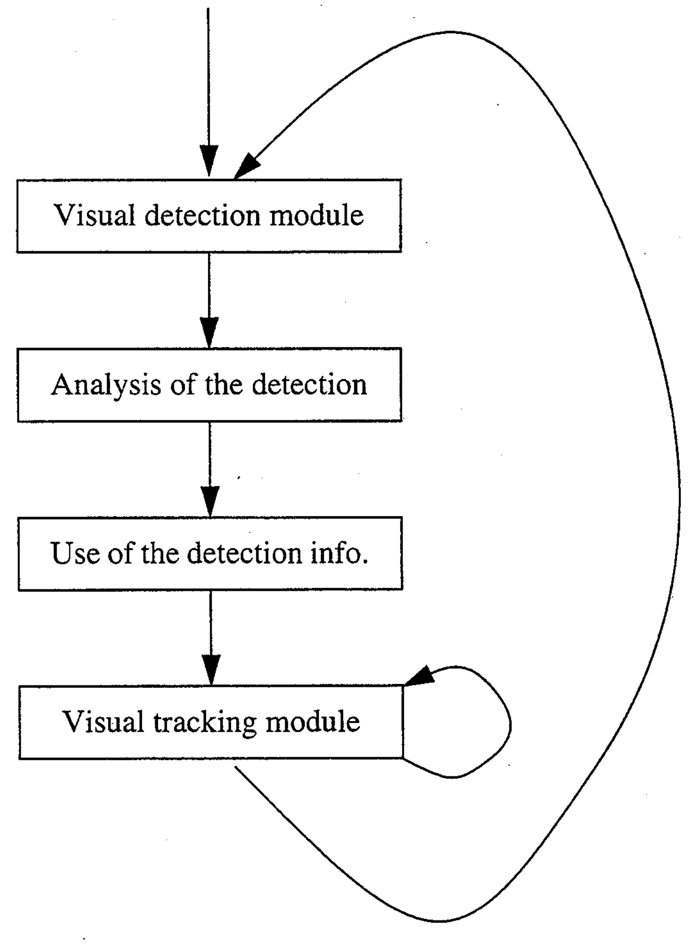


Figure 1: Diagram of Control Flow through the Framework

instances of detection, we maintain knowledge of the objects' situation through some means of visually tracking them. The tracking algorithm is then applied to each of the image frames between the instances of detection.

CHAPTER 2
DETECTION OF OBJECTS OF INTEREST

THEORETICAL BASIS OF THE DETECTION PROBLEM

The tracking of moving objects is an interesting and important problem in computer vision. However, the determination of an object's existence and type is also required if we are to successfully pursue the goal of a more fully automated vision system. By itself, a tracking algorithm makes two limiting assumptions:

- It assumes that the system has a means of knowing when an object appears that is important to a specific application.
- It assumes that the system has a means of knowing the object's location, as well as object attributes that are required by the tracking algorithm (e.g., the location of trackable features within the object).

In the past, computer vision projects have sometimes avoided the detection problem by obtaining this knowledge from a human user with an interface [26]. Other projects have used techniques (e.g., pedestrians with self-illuminating clothing) which make object detection trivial, but they are not suitable for the robust detection of objects in unpredictable, real-world environments. In this chapter, we formulate a detection framework that addresses this problem more realistically. According to the categorization of our system's modules, object detection must precede all other modules. This is because it provides the initial impetus for the vision system. But instead of ceasing activity when an object is detected, this module continues to provide valuable information to the other system components.

The basis of the detection framework is that every image consists of pixels belonging to one of two categories: *figure* or *ground*. Figure pixels are those which are believed to be part of the projection of an object of interest, while ground pixels correspond to that object's surroundings. If we allow the possibility of processing multiple objects, then a pixel is figure if it is part of any of the objects' projections. We formalize the detection problem as the identification and the analysis of figure pixels in each frame of a temporal sequence of images, i.e., the computation of $I_f(x, y, t)$

$$I_f(x, y, t) = \begin{cases} 1 \text{ if } I_c(x, y, t) \in \Upsilon_c(x, y, t) \\ 0 \text{ otherwise,} \end{cases} \tag{1}$$

where $I_c(x, y, t)$ is the current intensity value of the pixel at location (x, y) at time t, and $\Upsilon_c(x, y, t)$ is the set of allowable values for that pixel to be figure at that time.

CONSTRUCTION OF A FIGURE IMAGE

There is a wide variety of techniques that could be used for the identification of whether a pixel is part of the figure or the ground. For example, we could possess a model of the average shape of objects and attempt to fit this model to locations within an image. However, detection algorithms that are too computationally intensive will not be able to perform within the limited time available for real-time processing. In searching for a fast means of estimating the figure/ground state of a pixel, we consider the following heuristic regarding moving objects. These objects tend to be located where pixel intensities have recently experienced significant change, while environmental objects tend to be displayed by pixels whose intensity change is very slight. Thus, a temporal comparison between pixel intensities may yield information about the existence of important objects.

As a base for a temporal comparison, the proposed framework maintains a *ground image* ($I_g(x, y, t)$) that represents the environment's past history. For each pixel in the current image ($I_c(x, y, t)$), a comparison is made to the corresponding pixel in the ground image. If they differ by more than a threshold intensity amount, then the pixel is considered to be part of a binary *figure image* ($I_f(x, y, t)$). In accordance with our heuristic, the set $\Upsilon_c(x, y, t)$ of acceptable intensity values is selected so as to yield the following pixel acceptance test for the construction of the figure image:

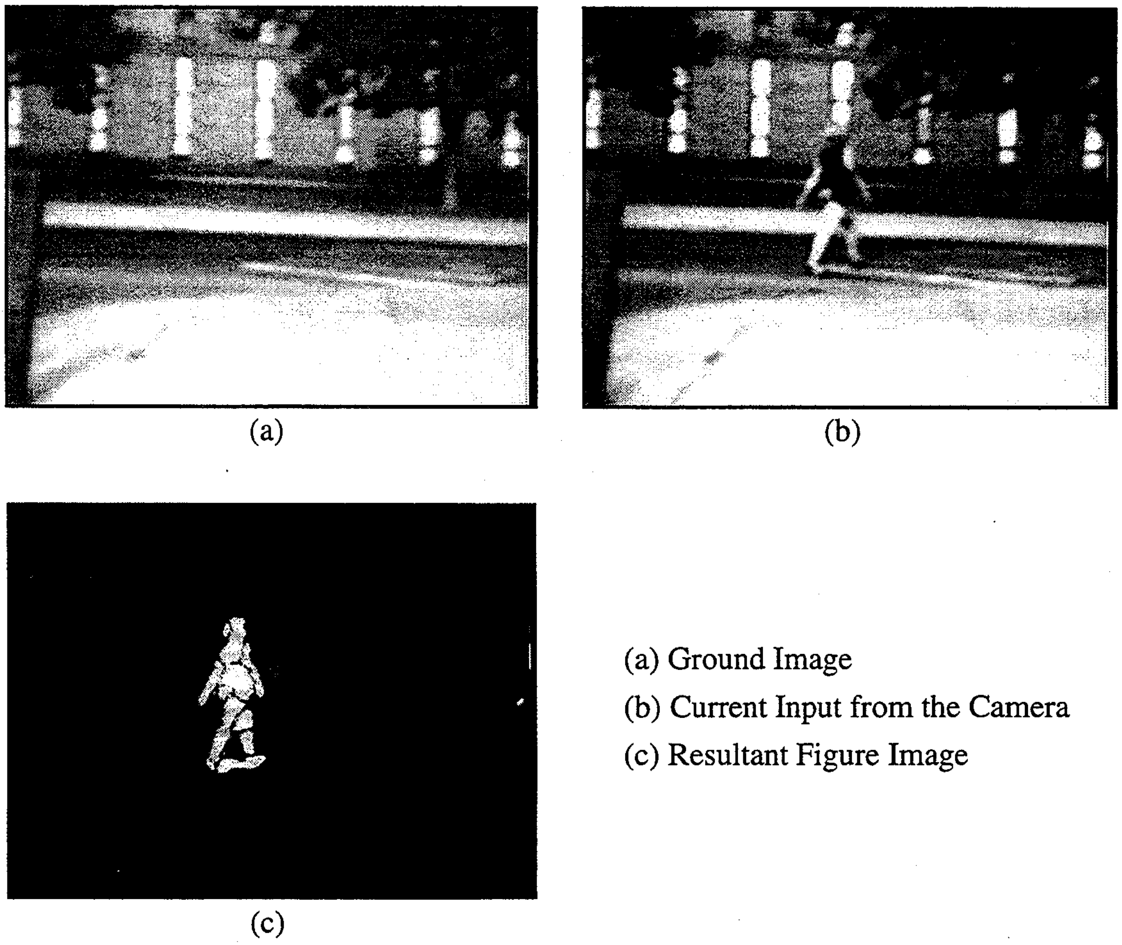
$$I_f(x, y, t) = \begin{cases} 1 \text{ if } |I_c(x, y, t) - I_g(x, y, t)| > c_1 \\ 0 \text{ otherwise.} \end{cases} \tag{2}$$

The choice of threshold value c_1 plays an important role in this process. If this threshold is too small, then portions of an object may blend into the background. If the threshold is too large, then slight changes in the environment will cause "false positive" errors in which the figure image contains many pixels that do not belong to objects of interest. A more sensitive (i.e., smaller) threshold can be used if the images are preprocessed with a low-pass filter (i.e., the image is convolved with an 8 x 8 discrete approximation to a Gaussian kernel). Inherent noise in the vision sensor's signal occasionally causes small, brief variations in an area's intensity values. By spatially averaging pixel intensities, the low-pass filter reduces the difference values that result from this noise. In addition, this filter can be implemented with Datacube pipeline processing elements (the Datacube is part of our hardware environment), making it a more efficient approach than more complicated means of noise reduction.

Figure 2 illustrates the construction of a figure image by a vision system monitoring the traversal of a pedestrian across an intersection. The upper left window shows a ground image that has been stored in memory. The upper right window shows the current image. The lower right window shows how the pedestrian becomes readily apparent, as the figure image is formed by comparing the current and the ground images.

CONSTRUCTION OF A GROUND IMAGE

Initially, the ground image is a copy of the first image of a sequence. However, environmental changes (e.g., lighting changes and shadows) may cause a background pixel's intensity to vary over time. To account for this dynamic aspect of some environments, our system periodically



(a) Ground Image
(b) Current Input from the Camera
(c) Resultant Figure Image

Figure 2: Construction of a Figure Image

updates the ground image with information from the current frame of intensities. Rather than periodically replacing the previous ground image, a new ground image is produced by incorporating new intensity values from the current image according to a time-average:

$$I_g(x, y, t) = \begin{cases} I_c(x, y, t) & \text{if } t = 0 \\ I_g(x, y, t-1) & \text{if } (t \bmod c_3) \neq 0 \\ (1-c_2)I_g(x, y, t-1) + c_2 I_c(x, y, t) & \text{otherwise,} \end{cases} \quad (3)$$

where c_2 is a constant representing the influence of the current image in each ground image update, and c_3 is a constant representing the frequency at which new ground images are produced. Both these parameters are obtained in a heuristic fashion or through the decisions of higher-level processes. For example, [20] describes higher-level processes that establish the values of similar parameters based upon conditions such as the time of day.

FIGURE IMAGE SEGMENTATION

Once a figure image has been obtained, we can consider the task of using the information that it provides to identify objects of interest. Unfortunately, figure images tend to contain pixels that belong to a variety of items other than just the objects of interest. For example, one may find that the figure image indicates activity caused by shadows, snowfall, or salt-and-pepper noise. Another problem that appears with the analysis of figure images is that an image may contain multiple objects of interest. In order to better distinguish between several objects, the figure image is partitioned into *segments* that each represents a single object. This figure image segmentation has a beneficial side-effect of identifying and removing many instances of the problem cases described above.

The figure image segmentation is achieved through a single pass of the Sequential Labeling Algorithm described in [14]. Because the algorithm creates segments from only a single pass through the figure image, all statistics that are to be calculated for the segments must be obtained dynamically. The selection of which statistics to calculate depends on the segment analysis (see Section on page 21). In some cases it may suffice to maintain a size (pixel count) and a segment bounding box (see Figure 3).

By following the algorithm of [14] without further modification, there can be as many as several hundred segments generated in a single image, only a few of which describe actual objects of interest. Since the computational performance of object detection relies on keeping the number of considered segments to a minimum, modifications must be made in order to make the process more efficient. After each scanline is processed, an analysis is made of all segments that have been completely processed by previous scanlines. If any of these completed segments are found to have different dimensions than those of a typical object of interest, then they are removed from the segment collection data structure. In practice, this pruning removes a large number of the undesirable sources of figure segments. By continuously minimizing the number of segments in the data structure, this process also improves the speed at which the segmentation can occur.

A common artifact that is observed with figure images is that objects of interest will often be illuminated in a way that causes a curve of background-matching intensities along the interior of their projected area. This *intensity gradient* artifact also occurs at regional borders within an individual object of interest, such as at the neckline formed by the top of an individual's shirt. The result of figure image segmentation when this artifact occurs is a pair of segments whose bound-

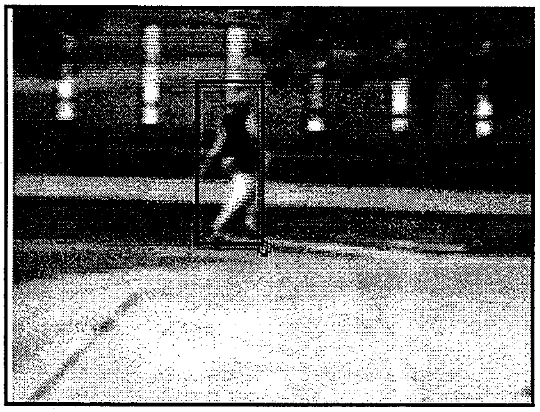


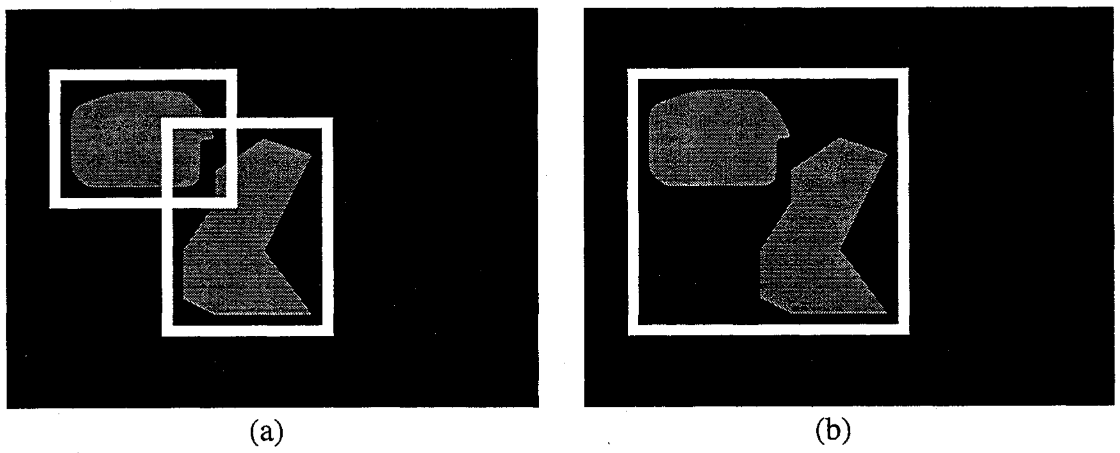
Figure 3: Segmentation Output

ing boxes overlap (see Figure 4). Since these curves are usually not parallel to an axis for their entire length, bounding box pairs caused by this phenomenon almost always overlap. Thus, our system merges overlapping figure segments into a single segment. This merging of figure segments can sometimes have the additional benefit of overcoming the occlusion of an object of interest by a small obstruction (see Figure 5).

SEGMENTATION OPTIMIZATIONS

Segmentation Subsampling

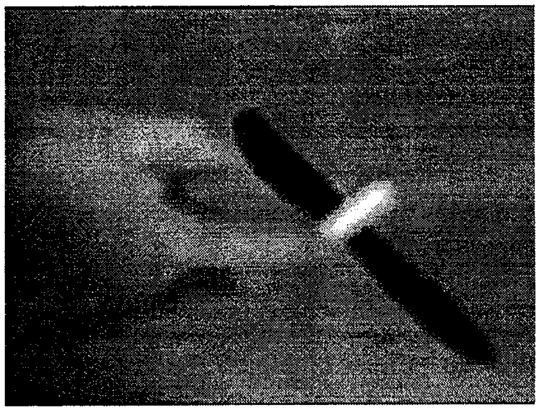
As stated thus far, the figure/ground framework provides a relatively fast form of object detection. However, the detection problem's critical real-time nature is such that we would still like to identify ways in which the detection performance can be further improved. Such an

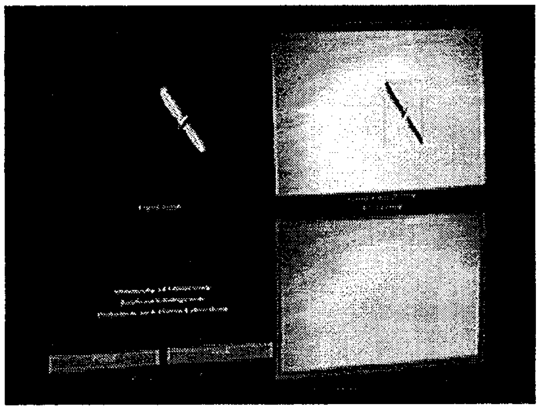


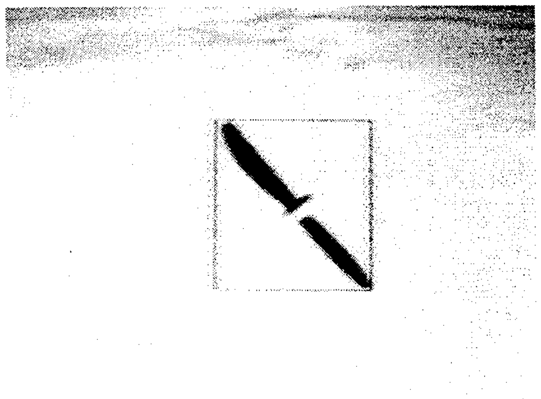
(a) Two intersecting figure segments before they are merged
(b) The figure segment that results from their merging

Figure 4: Merging of Figure Segments

improvement is possible through the use of *subsampling*. Because the segmentation algorithm is currently implemented in software, each figure pixel must be considered sequentially. The subsampling optimization improves performance by considering only every few pixels instead of every individual pixel. We make the assumption that if a pixel has the same figure/ground state as the previously considered pixel (in either dimensions), then all the pixels between these two share that same state. As a result, this optimization causes object detection to fail when the objects approach their minimally acceptable height or width. The failure is due to the fact that they are being pruned away by the segmentation process (see Section on page 13) in situations when the


(a)


(b)


(c)

(a) An occluding object (a strip of paper) is placed over a target (a knife).

(b) Ground, current, and figure images are maintained as before.

(c) The two ends of the knife produce bounding boxes that overlap. As a result, a single figure segment is produced for the target despite the occlusion.

Figure 5: Overcoming an Occlusion through Figure Segment Merging

consideration of each individual pixel would have resulted in an acceptable height or width. Thus, the technique can improve the performance of the detection framework, especially when the quantity of figure pixels is large, but at the expense of lower segmentation resolution.

Segmentation Domains

Another possible optimization involves the use of *segmentation domains*. A domain is an individual, rectangular portion of an image within which the segmentation algorithm is applied. Instead of using a single domain that covers the entire image, time can be saved by the appropriate use of several, smaller domains. The segments obtained from each domain are then combined into a resultant object set.

In our approach, there are two types of domains, *spontaneous* and *continuous*. Spontaneous domains are rectangular areas specified by a person as part of configuring the detection system to a particular location and setting. These domains are placed where the person anticipates that an object will appear for the first time. For example, detection of objects carried along a conveyor belt may work best if the spontaneous domains are placed where the belt's projection reaches the image edges. The spontaneous domains remain static throughout the system execution, and segmentation is constantly being performed in those areas.

Although objects in our conveyor belt example originally appear only at the edges of the image, we still need to be able to detect them as they move throughout other parts of the image. Therefore, continuous domains are used to follow objects' motion at times when they move away from areas indicated when the user selected spontaneous domains. Continuous domains are automatically regenerated after each iteration of detection. A continuous domain is formed as a rectangle centered around the locations of a new figure segment, with slightly more pixels in each dimension. This extra domain area around the sides of segments allows the system to continue to detect objects of interest as they move.

Since spontaneous and continuous domains may intersect, the set of domains is temporarily merged by a method that is similar to figure segment merging. Because the merging only applies for a single iteration of detection, this process results in a behavior of repeated splitting and merging that reflects the state of objects in the image, as illustrated in Figure 6. It does so in a way that accomplishes two goals:

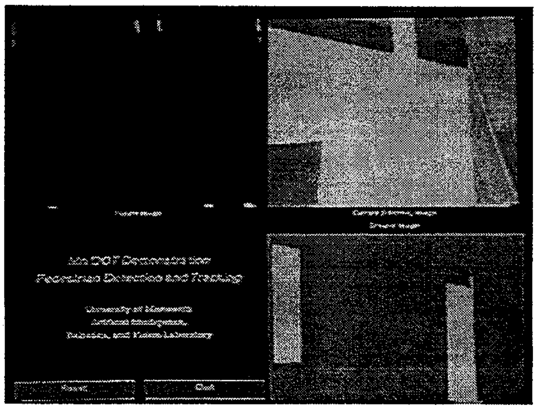
- Objects of interest are always contained within a domain (as long as objects originate in the spontaneous regions).
- Figure segmentation is only applied to areas where it is expected to provide results (i.e., in intrinsically active areas or in areas currently near an object of interest).

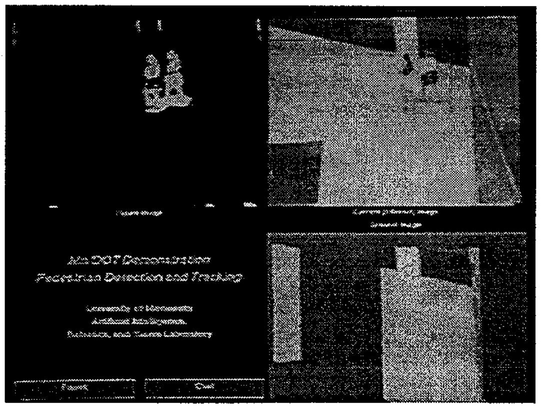
The domains also provide for robust tracking of multiple objects of interest. Even if two objects appear in the same spontaneous domain, they can follow completely different paths through the environment and can still be tracked, since separate continuous domains will be assigned to each one.

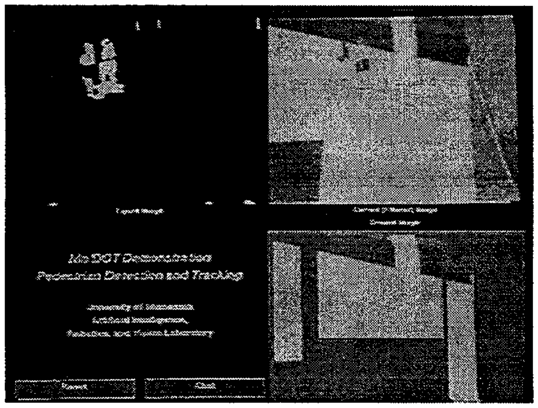
SUMMARY

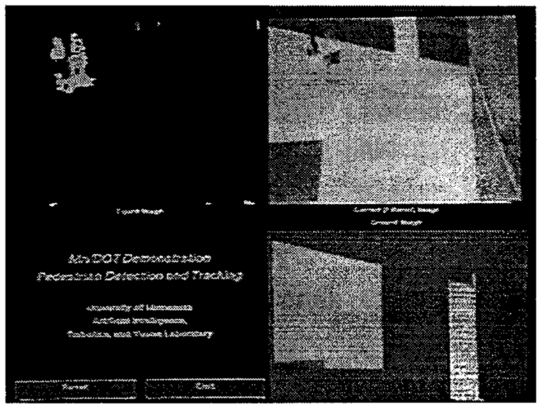
There is a clear need for a detection component in many computer vision applications. Because of the constraints of processing within brief periods of time, the technique used for performing this detection must not only succeed to identify the presence of an object, but it must do so through the use of a straightforward, fast method. In this chapter, we have proposed a figure/ground frame differencing technique in order to satisfy those two goals. Experimentation with this system has shown that both goals have been met:

- The figure segmentation output provides enough information for sufficiently accurate localization of objects of interest.
- Because of special implementation efforts including a modified segmentation algorithm and domain-restricted processing, the system has been made to function sufficiently fast.


(a)


(b)


(c)


(d)

(a) The user initially selects two domains at the ends of a sidewalk.

(b) The continuous domain around the pedestrians is intersected with the spontaneous domain that was already there.

(c) When the pedestrians move away from the spontaneous domain, a continuous domain follows them.

(d) Eventually, the continuous domain merges with another spontaneous domain when the pedestrians reach the other end of the sidewalk.

Figure 6: Segmentation Domains

CHAPTER 3

APPLICATION OF FIGURE SEGMENTS

INTRODUCTION

The methods described thus far form a framework with which a computer vision system can automatically detect moving objects. However, there are additional challenges that an automated detection and tracking system must often consider. One such challenge is object classification. The goal of object classification is to make an intelligent estimation regarding the type of an object. Another ability that we often desire for our vision systems is the selection of appropriate feature windows from within detected objects of interest. These feature windows are often used for other vision algorithms, especially for the tracking of the object. Both of these problems involve the application of figure segments that were previously obtained from the object detection module.

OBJECT CLASSIFICATION

Vision systems may be expected to face a variety of objects of interest. In addition, the system may be expected to function differently based upon the type of object that is present. Therefore, we have extended the abilities of our system to include a mechanism for the classification of detected objects. An obvious application of this ability is in an assembly line where a robot scans a workspace through which several different components pass. In this application, the robot would be given the task of servoing toward only a subset of the parts that it encounters.

We attempt to solve this problem by maintaining a model of the object type. This model consists of reasonable upper and lower bounds for each of the statistics gathered from figure image segmentation. In order to obtain these statistical bounds, training experiments are conducted with each object of interest in their natural setting (in several possible orientations). The orientations are selected so as to produce both minimal and maximal values for each one of the object's statistics. The hope is that a resulting object model will have statistical bounds that are wide enough to encompass any general object motion, yet narrow enough to uniquely identify that object.

FEATURE WINDOW SELECTION

Before addressing the problem of automatically selecting feature windows, it is necessary to understand what qualities we desire the feature window to possess. For our tasks, we would like feature windows to have the following two important qualities:

- The feature window should contain enough pixels that correspond to the object of interest so that the window may be effectively used for tracking. Ideally, each of the feature window's pixels corresponds to the object, causing the window to be invariant with respect to the object's surroundings.
- The pixels within the feature window must possess enough contrast for the window to be tracked.

Alternatively, these concerns could be expressed in the form of the following two questions:

- Which candidate feature windows should we consider?
- What confidence measure should we use to evaluate the usefulness of the candidates?

By computing a bounding box around an object of interest, we have provided a solution for the first of these two concerns. The problem of locating trackable features for an object has been reduced from a search throughout the entire image to a search within the object's bounding box.

The qualification of feature window intensity arrangement with a good confidence measure is critical because visual tracking algorithms may fail due to repeated patterns or areas of uniform intensity in the intensity arrangement. Both situations cause a tracking algorithm to reach ambiguous matches from within its search neighborhood, resulting in an output of spurious displacement measures. In order to avoid this problem, our system considers and evaluates many candidate placements of a feature window. Our system then selects feature window placements that optimize the performance of a tracking algorithm.

Optical Flow

Because the confidence measure that we use for selecting feature windows is based on potential tracking performance, it is necessary to formalize the motion of an object that is computed by the tracking module. The projection of a moving object is modeled through perspective projection with a *camera obscura* (i.e., pinhole camera) of focal length f [26]. This model projects points $\mathbf{p}_s(t) = (x_s(t), y_s(t), z_s(t))$ from the scene (reference frame R_s attached to the camera) to points $\mathbf{p}_i(t) = (x_i(t), y_i(t))$ in an image reference frame R_i according to the equation:

$$\mathbf{p}_i(t) = \left(\frac{f \cdot x_s(t)}{c_4 \cdot z_s(t)} + c_6, \frac{f \cdot y_s(t)}{c_5 \cdot z_s(t)} + c_7\right) . \tag{4}$$

The scale factors c_4 and c_5 account for pixel size and sampling. In this equation, c_6 and c_7 account for any displacement of the image reference frame with respect to the optical axis.

If we suppose that the camera moves with a translational velocity $\mathbf{T}$ and a rotational velocity $\mathbf{R}$ with respect to a static environment, then we can use the following equation to describe the change in object coordinates with respect to the reference frame R_s:

$$\frac{d\mathbf{p}_s}{dt} = -\mathbf{T} - \mathbf{R} \times \mathbf{p}_s . \tag{5}$$

The success of the tracking module will be based upon the ability to correctly locate:

$$\mathbf{p}_i(t+1) = (x_i(t) + \Delta x_i(t), y_i(t) + \Delta y_i(t)) \quad , \tag{6}$$

which corresponds to the point that was previously at $\mathbf{p}_i(t)$. In order to provide enough contrast for tracking, we use a window of intensities $W_{\mathbf{p}}$ around a point $\mathbf{p}$ [34]. We assume that a feature window's intensity values remain relatively constant over the duration of their use.

If we assume that $\mathbf{p}_i(t+1)$ can be found within a neighborhood $N_{\mathbf{p}_i(t)}$ around $\mathbf{p}_i(t)$, then we can locate it with a matching-based technique known as the *Sum-of-Squared Differences* (SSD) [3]. The SSD algorithm selects the displacement:

$$\Delta\mathbf{p}_i = (\Delta x_i(t), \Delta y_i(t)) \in N_{\mathbf{p}_i(t)} \tag{7}$$

that minimizes the error measure:

$$\begin{aligned}\varepsilon = \sum_{i_x, i_y \in W_{\mathbf{p}_i(t)}} & [I_c(x_i(t) + i_x, y_i(t) + i_y, t) - \\ & I_c(x_i(t) + \Delta x_i(t) + i_x, y_i(t) + \Delta y_i(t) + i_y, t+1)]^2.\end{aligned} \tag{8}$$

The assumption that the point can be found within a neighborhood has important implications when one considers issues relevant to visual tracking.

Confidence Measures

Feature selections made from within a bounding box typically lie entirely within the object or contain at least a portion of the object's pixels. *Confidence measures* are used to determine which of these are most suited for tracking. Many different types of confidence measures are obtained through the use of an *auto-correlation* algorithm [3][26]. Auto-correlation assumes the image to be stationary ($I_c(x, y, t+1) = I_c(x, y, t)$) and applies the SSD measure [8] to form a candidate feature window's auto-correlation surface:

$$\varepsilon_a(\Delta x_i(t), \Delta y_i(t)) \qquad \text{where } (\Delta x_i(t), \Delta y_i(t)) \in N_{\mathbf{p}_i(t)} \quad , \tag{9}$$

with a minimum at the origin.

Several possible confidence measures can be applied to the surface in [9] to measure the suitability of a candidate feature window. A particularly robust confidence measure is a two-dimensional parabolic fit. This measure takes advantage of the fact that better candidate feature windows will have auto-correlation surfaces that are paraboloids with steep surfaces on all sides [26]. The parabola $\varepsilon'_a(x_\varepsilon) = c_8 x_\varepsilon^2 + c_9 x_\varepsilon + c_{10}$ is fit to a candidate's auto-correlation surface in each of four predefined orientations. In other words, the parabola parameter x_ε corresponds once to each of:

$$(\Delta x_i(t), \Delta y_i(t)) \in \{(1, 0), (1, 1), (0, 1), (-1, 1)\} \quad . \tag{10}$$

Since better features have steeper auto-correlation sides all around, the confidence measure is defined as the minimum of the four values of the c_8 coefficient. The candidate feature windows with the largest confidence measures are selected.

Gradient Descent Optimization

Computing an auto-correlation SSD surface can be a time-consuming operation. Since the system will be performing this operation for each candidate feature window, we should try to limit the number of candidates that are considered for each object of interest. Although an exhaustive search of each possible feature window placement within an object's bounding box would find the highest confidence value, it may be desirable to sacrifice this guarantee in favor of finding a confidence measure that may not always be ideal, but can generally be found faster. This is done by performing a *gradient descent search* through the space of candidate feature window placements. Beginning at an arbitrary window placement, the search consists of a series of locally optimal *greedy* decisions. Each decision computes the confidence measure of neighboring window placements and selects the best (the maximum). At the selected location, the same decision process is repeated until no neighboring confidence measures are better than the current one.

Imagine this search problem as one where you are located on an uncharted three-dimensional terrain and the goal is to locate the lowest point. If enough surface charting has taken place to indicate a depression, then it would make sense to continue charting in the direction of the depression. However, this algorithm provides no guarantee against instances of ridge, plateau, or foothill problems [5]. The gradient descent optimization is typically able to compensate for this decrease in the perfection of its results by virtue of its speed.

Depending on the number of desired resultant feature window selections, several searches can begin at different locations within the object's bounding box. A natural choice for at least one of these searches is to start from the box's center. By starting the gradient descent searches nearer to the center of the bounding box, the resulting feature window selection is more likely to contain pixels corresponding only to the object of interest.

SUMMARY

In addition to solving the problem of detecting when objects of interest are present, the figure/ground framework provides an computer vision primitive—figure segments—that can be used for further processing. In this chapter, we have seen how figure segments can be used for object classification and feature window selection. However, the application of figure segments that probably holds the most promise is their use for visual tracking. That topic will be addressed as part of the next chapter.

CHAPTER 4
FIGURE SEGMENT CORRESPONDENCES

RELATION TO VISUAL TRACKING

The second major goal of this work is a visual sensing modality capable of tracking the motion of objects in a temporal sequence of images. Following the design of our software system, a visual tracking module can be provided with a set of figure segments that contain several attributes, including a set of good feature windows. However, the detection module provides no correspondence among the figure segments representing an object at different points in time. We define the problem of *object tracking* to be the computation of such a correspondence. The figure segment correspondence problem becomes more complex when we allow the possibility of processing multiple objects of interest at the same time.

The visual tracking module is integrated with the other modules of our computer vision system. The system performs several iterations of the tracking module, after which it returns to repeat any detection and selection steps. Additionally, search-specific optimizations are included in the system to optimize the performance of the correspondence mechanisms used by our experiments.

ADJACENCY

The most straightforward approach to corresponding segments is to establish a correspondence if and only if a segment's bounding box intersects the bounding box of a segment in the previous iteration. This means of computing figure segment correspondence has several advantages over more complicated methods, including:

- The method is straightforward and easy to implement.
- The time required to compute the intersection of bounding boxes is much less that the time required to compute the cross-correlation surfaces required by more complicated

correspondence methods. Since this method is able to more quickly return control back to the detection module, portions of the detection framework (e.g., the size of the continuous segmentation domains) can be made to be more sensitive to time delays between iterations.

- The adjacency method avoids the need to compute feature windows for the objects of interest.
- The method is able to perform its processing based only on the information provided by the detection module, without analyzing any intensity values. Because of this fact, correspondence computation can more elegantly represent a higher level of visual processing. Also, this fact allows the pipeline image processing design to be less complex (a result of the intensity data needing to be routed to processors involved with only the detection module).

However, the adjacency method does suffer from one major drawback; it imposes a limit on the total amount of time that an iteration of the system can take. If the combination of detection and tracking operations takes too long, then a moving object may be able to move to a location where its bounding box does not intersect the bounding box from the previous iteration. In such a scenario, the tracking system fails. Following are alternative methods for computing figure segment correspondences, both avoiding this problem.

OBJECT CROSS-CORRELATION

We have experimented with an alternative method of figure segment correspondence that tracks a subsampled representation of an object's matrix of intensity values. In this method, figure segments are visually tracked by performing several iterations of *object cross-correlation.* Each cross-correlation iteration yields a new location for the object of interest by computing an estimate of the object's displacement from its previous location (its initial location is provided by the detection module). After performing a predetermined number of iterations, the detection module is called to produce a new set of figure segments. If a detected segment is found to intersect one that has been tracked, then it is assumed to correspond to the same object of interest. By using this method, the identity of an object can be maintained across several iterations of the vision system, even in situations where detected figure segments do not intersect with each other.

During each individual iteration of object cross-correlation, figure segment displacements are computed through the use of optical flow. Unlike the derivation of auto-correlation in [9], the intensity values for $I_c(x, y, t)$ are not made equal to the current image $I_c(x, y, t+1)$. Instead, they are obtained from an earlier image frame that was stored in memory. During each iteration, the SSD error measure of [8] is computed for displacements from the neighborhood $N_{\mathbf{p}_i(t)}$ to form a cross-correlation surface:

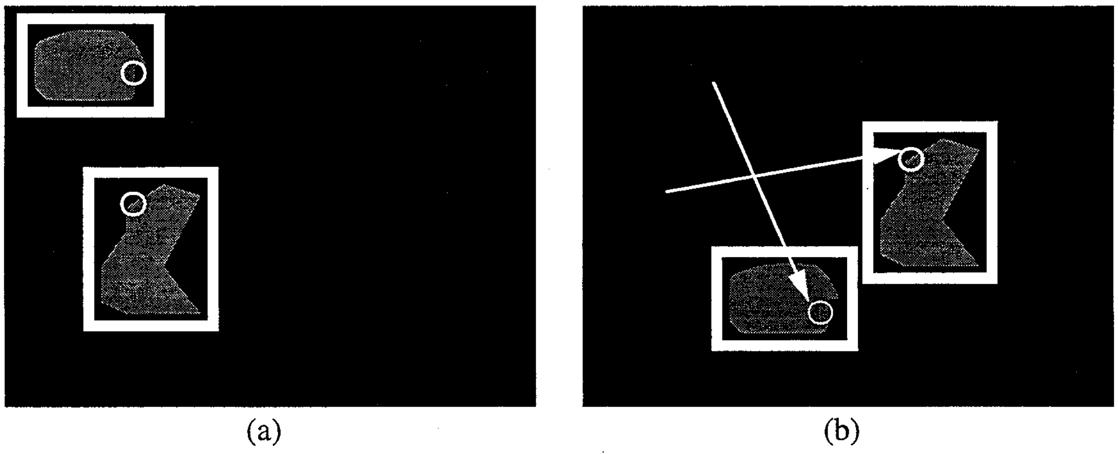
$$\varepsilon_c(\Delta x_i(t), \Delta y_i(t)) \qquad \text{where } (\Delta x_i(t), \Delta y_i(t)) \in N_{\mathbf{p}_i(t)} \quad , \tag{11}$$

where the window $W_{\mathbf{p}_i(t)}$ used to calculate the SSD measure is the feature segment's bounding box. The displacement that produces the minimum ε_c value is then taken to be the displacement of the figure segment.

FEATURE WINDOW CROSS-CORRELATION

There is an additional way in which a cross-correlation technique can be used to correspond figure segments for the visual tracking of objects. Instead of taking the window $W_{\mathbf{p}_i(t)}$ to be a figure segment's bounding box, it can be a small, recognizable portion of that object (i.e., a *feature window*). Again, the tracking module performs several iterations of SSD-produced displacement measurement, after which the detection module locates new figure segments. Object identity is maintained across iterations of the vision system in a similar manner as before. If a newly detected segment is found to contain a tracked feature window, then the segment is assumed to correspond to the object for which the feature window was originally selected. The accuracy of this correspondence mechanism can be increased by relying on multiple feature windows.

The speed at which the tracking module can operate is dependent on the size of the SSD windows that are used. By reducing this size, the feature window cross-correlation method has the potential of operating faster than the object cross-correlation method. However, this improvement comes at the cost of having to locate feature windows that are suitable for tracking. Figure 7 illustrates how feature window cross-correlation provides a method of successfully corresponding figure segments in a situation where the adjacency method would have failed.



(a) The detection module produces two figure segments. A feature window is found for each.

(b) After tracking the feature windows, the current window locations can be matched to newly detected figure segments.

Figure 7: Feature Window Method of Correspondence

CROSS-CORRELATION OPTIMIZATIONS

We expressed previously the importance of the efficiency of a SSD algorithm used for real-time systems. This importance applies for cross-correlation methods just as it did for the auto-correlation. In this section, we present several means of reducing the time that is spent computing SSD measurements for visual tracking systems. Each of these optimizations has been shown through experimentation to improve the computational performance time of the vision system.

SSD Window Subsampling

It was mentioned previously that the speed of the figure segmentation algorithm is increased by subsampling the pixels. In a similar manner, subsampling can increase the speed of the cross-correlation correspondence methods. Subsampling can be applied in two ways:

- When searching the neighborhood $N_{\mathbf{p}_i(t)}$ for displacements that minimize the SSD measure of [8], we consider only a subsampled number of displacements. This reduces the number of SSD calculations that are performed.
- When performing the SSD summation [8] over the window $W_{\mathbf{p}_i(t)}$, we consider only a subsampled number of the i_x and i_y terms. This reduces the time that each SSD calculation takes.

Pyramiding Levels

At first glance, the search—either exhaustive or subsampled—of the neighborhood $N_{\mathbf{p}_i(t)}$ for an minimal SSD value is sufficient to visually track a particular object or feature. But there is an additional problem that must be considered if we are to use either of the two cross-correlation methods. If the motion of an object of interest is rapid enough to cause its projection to move beyond $N_{\mathbf{p}_i(t)}$ before the tracking algorithm is able to complete its search, then the tracking algorithm will fail. However, if we try to solve the previous problem by increasing the size of $N_{\mathbf{p}_i(t)}$, then the search time increases, possibly making the problem worse.

The alternative option that we use in our system is to increase the size of $N_{\mathbf{p}_i(t)}$ proportionally to an increase in the neighborhood search subsampling. For example, if the lowest subsampling level computes an SSD measure for every displacement in a 9×9 neighborhood, then the second level would compute measures for every other displacement in a 18×18 neighborhood, and the third level would compute measures for every third position in a 27×27 neighborhood. Together, these form a sequence of *pyramiding levels* (see Figure 8).

Although the use of pyramiding allows the user to obtain both goals of capturing a large displacement and computing it quickly, it does so at a cost of reduced precision. The pyramiding level can be made dynamic, so that the figure segment correspondence algorithm uses a pyramid-

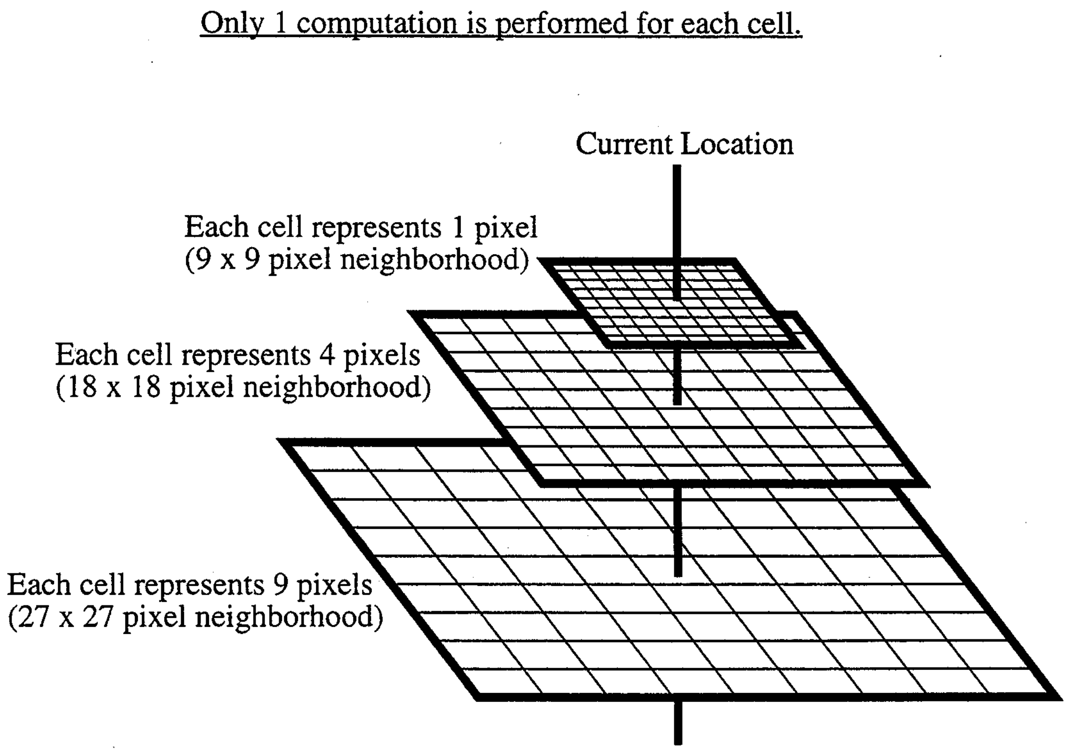

Only 1 computation is performed for each cell.
Current Location
Each cell represents 1 pixel (9 x 9 pixel neighborhood)
Each cell represents 4 pixels (18 x 18 pixel neighborhood)
Each cell represents 9 pixels (27 x 27 pixel neighborhood)

Figure 8: Sample Pyramiding Levels

ing level that is based upon the magnitude of the figure segment's previous displacement. Precision is kept by reducing the pyramiding level in response to small a displacement, but the level is increased if larger displacements indicate a need for a larger search neighborhood [29].

Gradient Descent, Revisited

We described previously an application of subsampling that computed the SSD measure of [8] for a restricted number of candidate displacements from the neighborhood $N_{\mathbf{p}_i(t)}$. However, subsampling uses a methodical, non-intelligent means of selecting candidate displacements (e.g., every third displacement from the neighborhood). An alternative method of selecting candidate displacements is to use a gradient descent search strategy like the one that was used for feature window selection (see Section on page 25). Beginning at a degenerate displacement, the search repeatedly makes greedy decisions until no neighboring displacement is better than the current one. Each decision evaluates Figure 8 for neighboring displacements and selects the best (the minimum).

Spiraling

In the previous section, we described an improvement on the way in which subsampling reduces the number of SSD calculations that are made. This improvement was obtained by using intensity information contained within the image itself. We can make a similar improvement on the other subsampling technique that we mentioned before, namely performing a SSD summation with a reduced number of the i_x and i_y terms from the window $W_{\mathbf{p}_i(t)}$.

Reducing the number of summation terms by the subsampling method does not take into account the fact that previous SSD measures have been computed in all but the first case. Because the cross-correlation correspondence algorithm is searching for the minimum SSD measure, our system can stop accumulating summation terms if the current summation value exceeds a previously computed SSD measure. There is a heuristic which says the following: i_x and i_y indices

HIGH-LEVEL ANALYSIS

Through the use of figure segment correspondences, we have advanced the capabilities of our computer vision system from the detection of an object's presence to the tracking of the object as it moves through the camera's field of view. This provides a basis from which a variety of analysis tasks are possible. Some tasks that have potential application involve the identification of object characteristics such as the size or class of a component in an assembly process. This information could be useful for a system where an individual robot scans an assembly line through which several different components pass. The robot would be given the task of servoing toward only a subset of the parts that it encounters. Another high-level application could include the identification of motion characteristics such as the time that it takes an object to move across a monitored area.

that are closer to the center of the window $W_{\mathbf{p}_i(t)}$ tend to increase the summation result more than those on the periphery [29]. Since we will need to add fewer terms if the terms that we choose to add are larger, we use a *spiral* pattern of iterating through i_x and i_y (see Figure 9).

There is another heuristic which states that an object without any motion bias will have an average displacement of $(0, 0)$. Since the spiral summation optimization works best when we find a minimum SSD value early in the search process, we can gain an additional benefit by beginning cross-correlation searches at the origin. Furthermore, if we choose to perform an exhaustive search of the neighborhood $N_{\mathbf{p}_i(t)}$ (for the sake of accuracy), it will clearly be advantageous to perform the search in a spiral pattern, as shown in the figure.

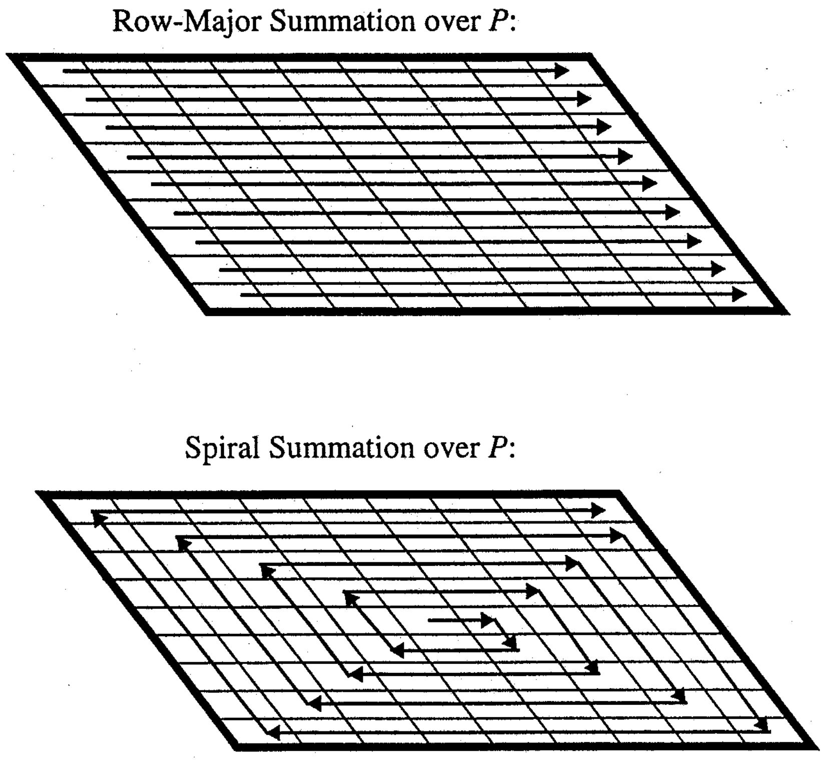


Figure 9: Row-Major and Spiral Patterns of Summation

CHAPTER 5
KALMAN FILTERING

MOTIVATION BEHIND THE USE OF KALMAN FILTERING

The pedestrian tracking system is based on connected region segmentation of the ``difference'' image followed by a phase of region merging. In the absence of noise and false targets, the method successfully extracts pedestrians. However, it is rarely the case that noise and false targets are absent. Shadows for example constitute a major source of error. When the pedestrian approaches a wall, the shadow cast on the wall may result in a large blob surrounding the pedestrian in the ``difference'' image. Region merging would result in the extraction of a bounding box surrounding the whole blob and the exact location of the pedestrian is lost. Another problem occurs when two pedestrians walking towards each other meet. Again, the extracted bounding box will surround both pedestrians. We tried to use Kalman filtering in order to address some of these problems.

KALMAN FILTER

We used a constant speed model as our pedestrian model. In this model, we assume that pedestrians walk with a constant velocity. Only the horizontal component equations will be presented here. The vertical component is treated in the exact same manner. In effect, we will be using two independent filters for each pedestrian, one for the horizontal component and one for the vertical component. In the following equations, t represents time.

The system equation is given by

$$\mathbf{x}_{t+1} = \mathbf{F}\mathbf{x}_t + \mathbf{w}_t \tag{12}$$

where $\mathbf{x} = \begin{bmatrix} x \\ v \end{bmatrix}$ is the state vector consisting of the location, x , and velocity, v, $\mathbf{F}$ is the transition matrix, $\begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$, and $\mathbf{w} = \begin{bmatrix} 0 \\ w \end{bmatrix}$ is a Gaussian system noise. The measurement model equation is given by

$$z_t = \mathbf{H}\mathbf{x}_t + u_t \tag{13}$$

where z is the measured location, $\mathbf{H} = \begin{bmatrix} 1 & 0 \end{bmatrix}$ is the measurement matrix, and u is a Gaussian measurement error. We denote the variance in system error by σ_h^2 for the horizontal filter and σ_v^2 for the vertical filter. Measurement error variances for the horizontal and vertical filters are denoted by δ_h^2 and δ_v^2, respectively.

The horizontal Kalman filter equations for the prediction and correction phases become:

- Predictions:
  $\hat{\mathbf{x}}_{t+1} = \mathbf{F}\mathbf{x}_t$

  $\hat{\mathbf{P}}_{t+1} = \mathbf{F}\mathbf{P}_t\mathbf{F}^T + \begin{bmatrix} 0 & 0 \\ 0 & \sigma_h^2 \end{bmatrix}$

(14)

- Correction:
  $\mathbf{K}_{t+1} = \hat{\mathbf{P}}_{t+1}\mathbf{H}^T(\mathbf{H}\hat{\mathbf{P}}_{t+1}\mathbf{H}^T + \delta_h^2)^{-1}$
  $\mathbf{x}_{t+1} = \hat{\mathbf{x}}_{t+1} + \mathbf{K}_{t+1}(z_{t+1} - \mathbf{H}\hat{\mathbf{x}}_{t+1})$
  $\mathbf{P}_{t+1} = (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})\hat{\mathbf{P}}_{t+1}$

In the above equations, $\hat{\mathbf{x}}$ and $\mathbf{x}$ are the predicted and estimated system states, respectively. $\hat{\mathbf{P}}$ and $\mathbf{P}$ are the predicted and estimated system covariance matrices, respectively. In our system, we used the values $\{1, 1, 5, 5\}$ for $\left\{\sigma_h^2, \sigma_v^2, \delta_h^2, \delta_v^2\right\}$ which have a bias towards the history of state since we assume that measurements can be highly noisy.

In the figures at the end of the chapter, one may evaluate the performance of the Kalman filters (the different curves correspond to different values of the filter parameters). The smoothing effect is apparent.

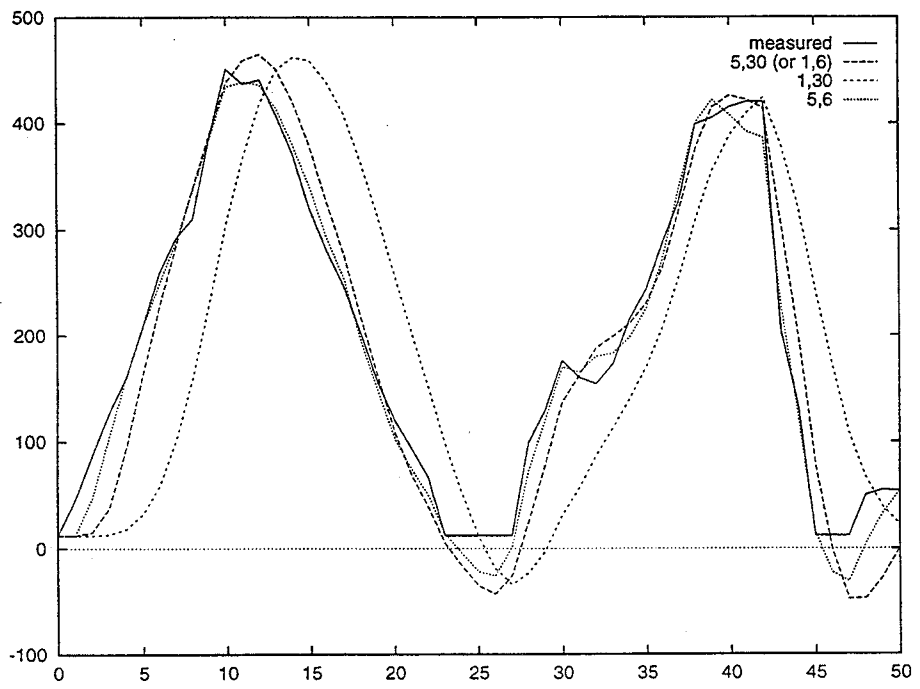


Figure 10: Horizontal Measurements and Their Filtering Results

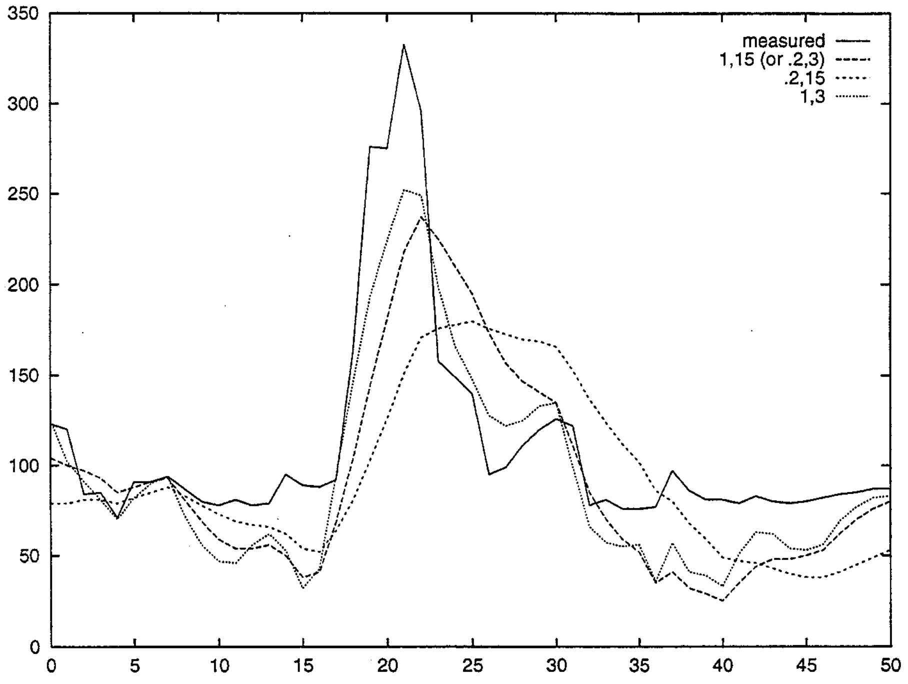


Figure 11: Vertical Measurements and Their Filtering Results

CHAPTER 6

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 7
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. Because of the large number of times that the summation in [8] is computed, 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 12, 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

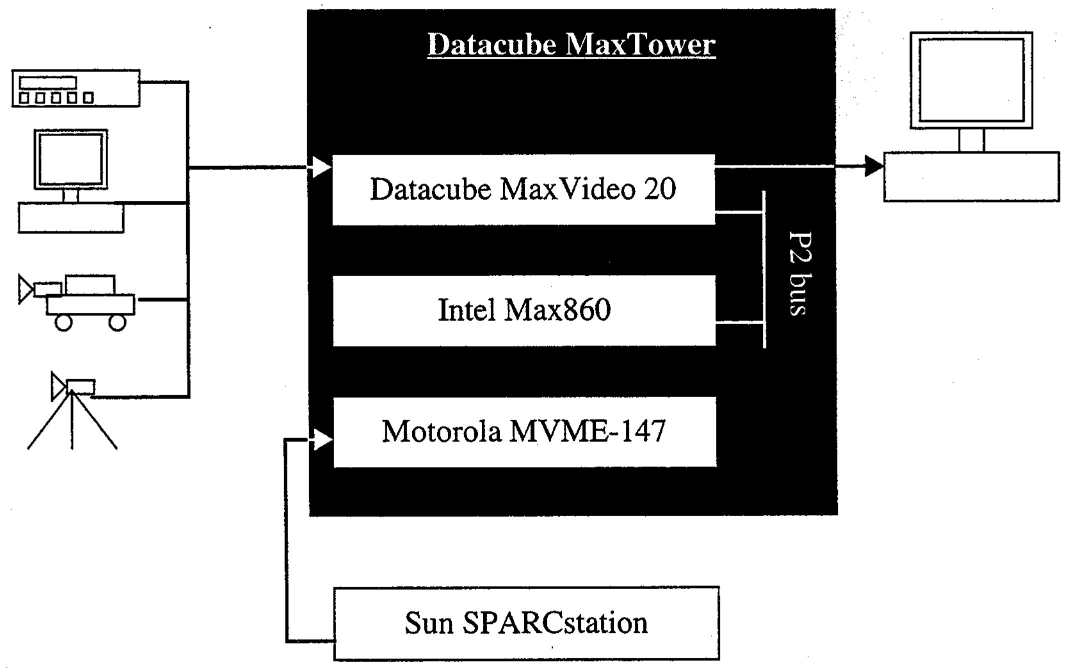


Figure 12: The Minnesota Vision Processing System

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 Imageflow 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.

CHAPTER 8

APPLICATION OF THE TECHNIQUES TO PEDESTRIAN CONTROL PROBLEMS

GOALS OF INTELLIGENT TRANSPORTATION SYSTEMS

Transportation systems have been subjected to considerable increases in complexity and congestion during the past two decades. As a direct result of these conditions, both American and European transportation systems have experienced a decrease in each of the following areas:

- safety,
- efficiency,
- and convenience.

In response, government agencies have been actively searching for ways in which modern technology can be used to improve travel conditions with respect to each of these topics. As a group, these concerns illustrate the problem area of an *intelligent transportation system* (ITS).

There are two modes in which the visual detection framework can be used to further these ITS goals. First, each of the three transportation conditions can be improved by providing the human operator of a vehicle with a computerized advisory system that supplements his or her own sensory information about the vehicle's environment. Second, systems that control the flow of traffic can be improved by making them more responsive to dynamic traffic conditions.

Both of these transportation topics rely upon the use of sensory devices—especially cameras—to dynamically provide the identity and location of traffic objects. The different objects that an ITS may need to consider are varied, including vehicles, traffic signs, obstacles, and pedestrians. It is reasonable to expect that a complete ITS would require the ability to automatically detect each of these types of traffic objects. Therefore, an ITS is a likely application of the object detection framework described in this report.

In order for an ITS to operate with minimal human intervention, it would also require the ability to track the traffic objects that it detects. The proposed visual tracking components provide this flexibility, without relying upon accurate measures of target object parameters or constraining the target's motion. Relatively high-speed targets are tracked under varying conditions without explicit target models. In addition, the identification of surface features on obstacles or the knowledge of approximate obstacle shape (i.e., the shape of a pedestrian's body) may further improve the usefulness of an ITS object detection scheme.

Next, we describe the automatic monitoring of pedestrians in typical urban settings.

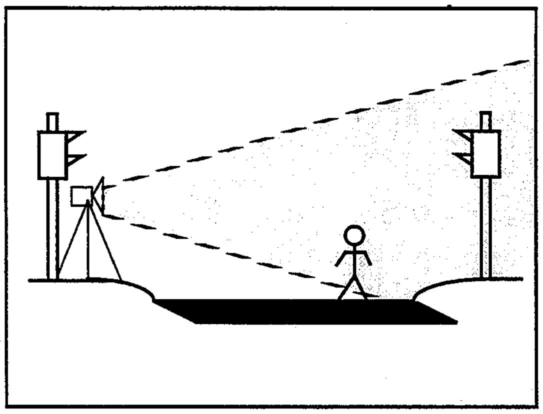
PEDESTRIAN MONITORING

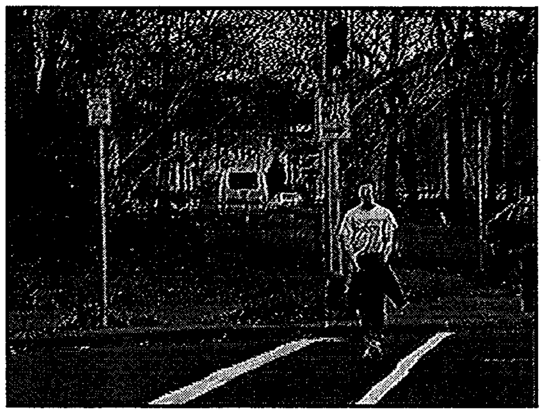
Pedestrians constitute one class of traffic objects that are of extreme importance to many ITS applications. Considering the potential for injury when pedestrians and vehicles come into contact, the ability to detect and track pedestrians is likely to be an component of any ITS.

For our *pedestrian monitoring* experiments, we capture images from a real intersection using a camcorder with a field of view encompassing the breadth of a street. The camera is statically mounted on or adjacent to a solid base, such as the utility pole supporting an intersection's crosswalk signals (see Figure 13). The experiments consist of pedestrians walking with a typical gait, either towards the camera or frontal-parallel to it. Later, the recorded image sequences can be input into the MVPS using a video cassette recorder.

The elements of the detection framework are successfully implemented. For a test image sequence of several minutes of pedestrian traffic, the system successfully detects every pedestrian entering in the camera's field of view. The framework is shown to be able to track multiple objects without a significant increase in computation. Figure 14 shows a series of illustrations that demonstrate how two bounding boxes produced by the segmentation of figure images can simultaneously track two pedestrians. The interaction between spontaneous and continuous domains substantially improves the speed of the detection phase. In Figure 15, the dynamic nature of the domains is illustrated.

We also demonstrate that our framework successfully performs segment correspondences under normal environmental situations. The cross-correlation algorithm is used to track a feature window on the waist of a pedestrian while he crosses an intersection. Six sample frames from this sequence are shown in Figure 16. In spite of the degraded contrast in the imagery due to an overcast sky, pedestrian tracking is not lost during the entire crossing. Cross-correlation search with the classical algorithm averages 136 milliseconds per tracking iteration over 5000 frames under a variety of feature window motion. By incorporating the subsampling and spiraling optimizations


(a)


(b)

(a) Camcorder placement diagram
(b) Sample image acquired by the pedestrian monitoring ITS

Figure 13: Pedestrian Monitoring

described previously, search with the same image sequence averages 8 milliseconds per iteration [29]. If we instead use the subsampling and gradient descent optimizations, search performance is

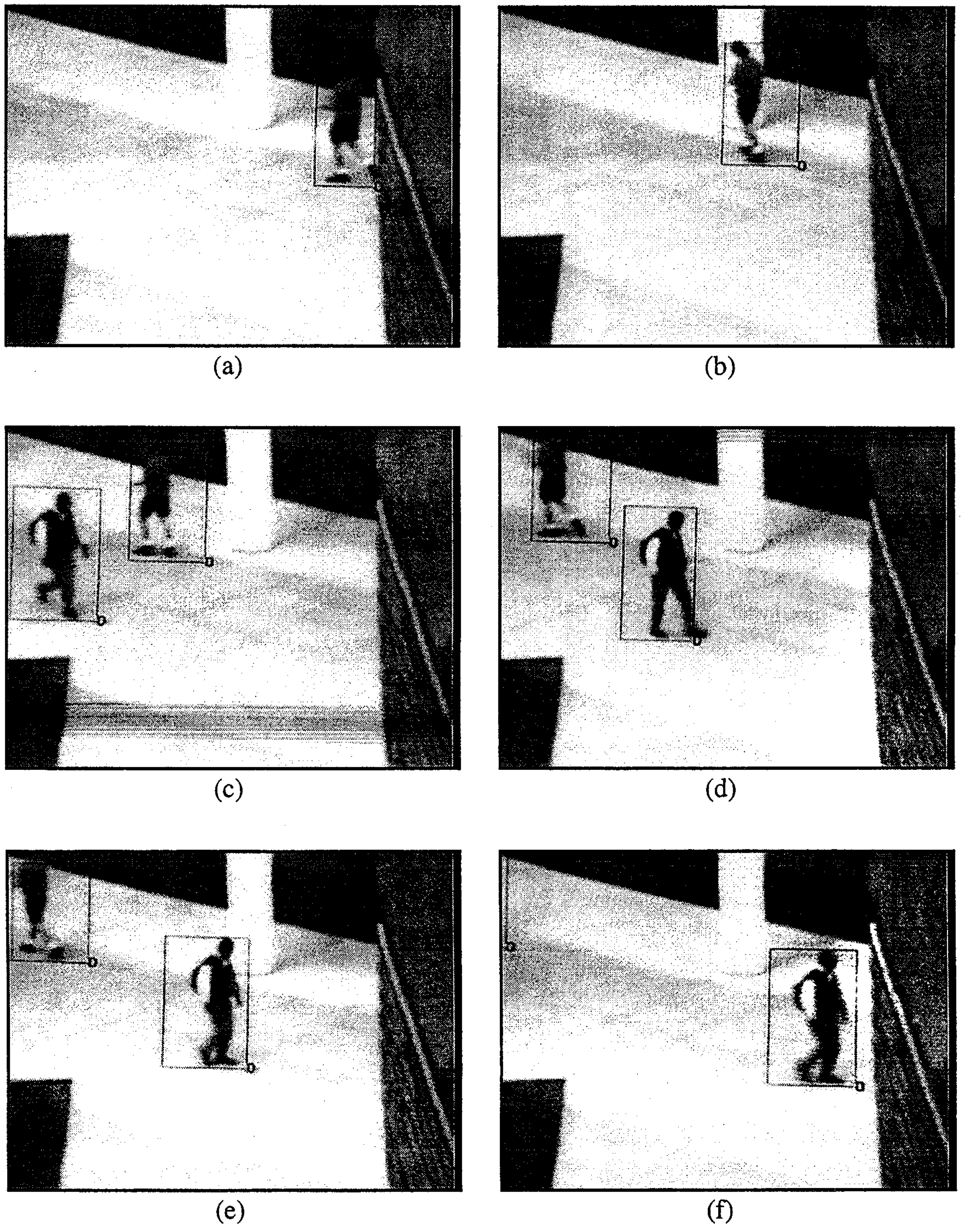
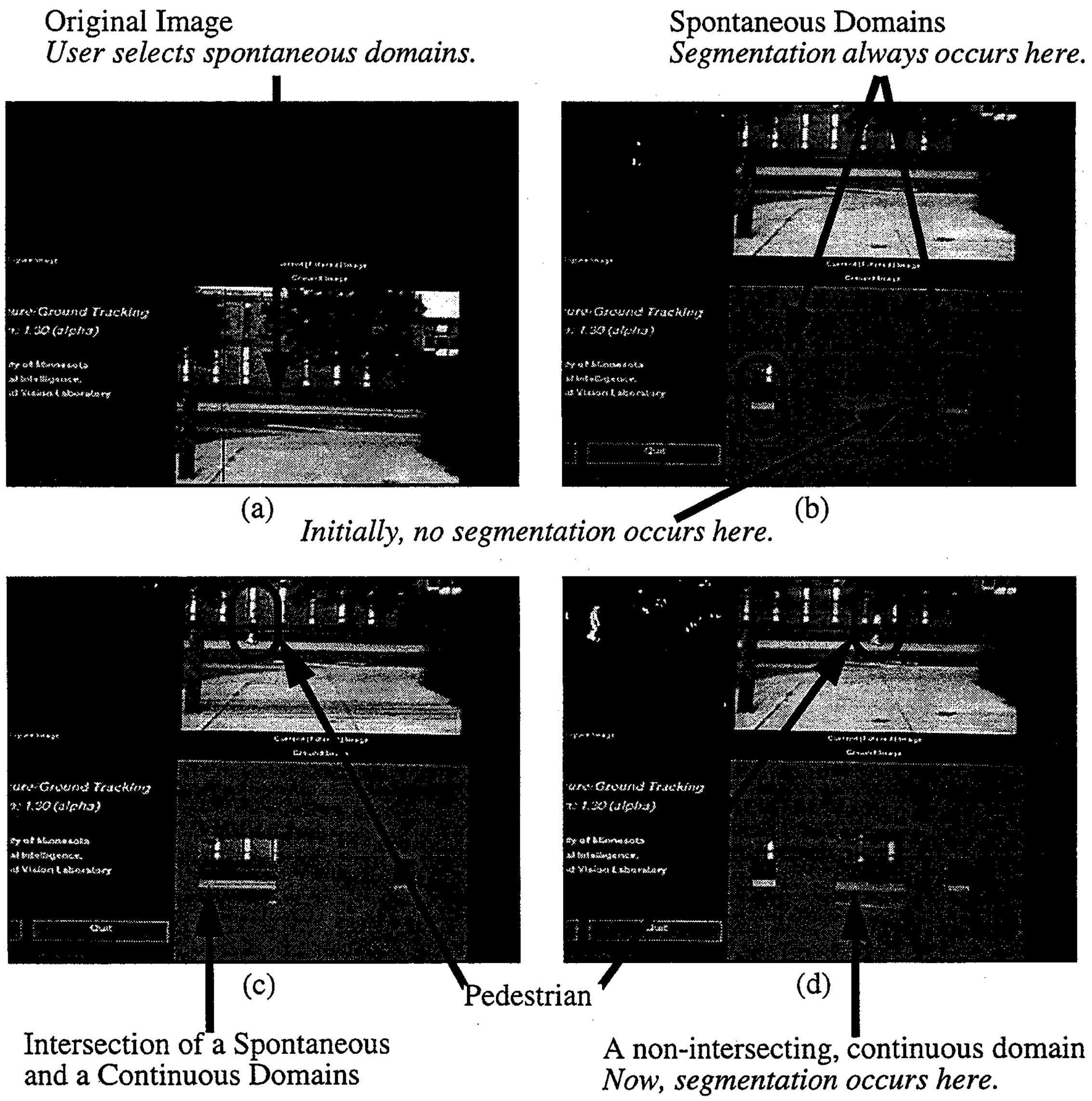


Figure 14: Pedestrian Monitoring using the Detection Framework

similarly improved.


Original Image
User selects spontaneous domains.
Spontaneous Domains
Segmentation always occurs here.
(a)
Initially, no segmentation occurs here.
(b)
(c)
Pedestrian
(d)
Intersection of a Spontaneous and a Continuous Domains
A non-intersecting, continuous domain
Now, segmentation occurs here.

(a) First, the user selects two small domains at the ends of a sidewalk.

(b) Once the selection is made, only a small area of the intersection's projection is considered for segmentation of the figure image.

(c) The segmentation has identified a pedestrian who has approached from the left. The continuous domain around the discovered pedestrian is intersected with the spontaneous domain that was already there.

(d) The pedestrian has moved away from the spontaneous domain, causing there to be three separate areas of figure segmentation.

Figure 15: Pedestrian Monitoring using Segmentation Domains

The ability to perform motion analysis is demonstrated by adding a module which allows a user to specify the extent of a represented intersection (when the system is configured to a partic-

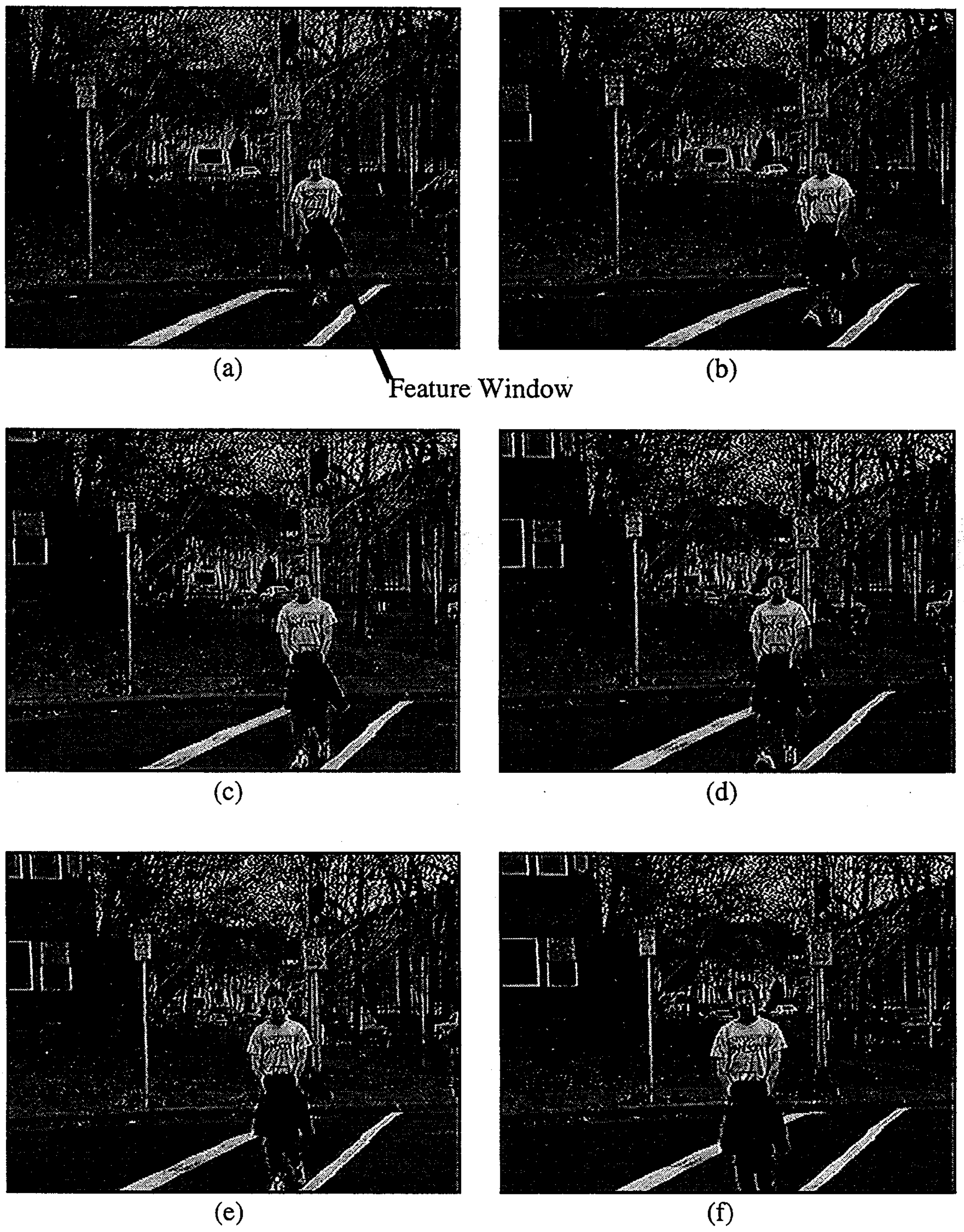


Figure 16: Pedestrian Monitoring using Figure Segment Correspondences

ular location). While the system in running, the positions of detected figure segments are monitored with respect to the specified intersection boundaries. If an object is detected to traverse the intersection from one edge to the opposite edge, the time that it took to do so is reported. Such tracking has direct application to intelligent signal control.

SUMMARY

The conditions of current transportation systems demonstrate a need for an ITS that possesses both visual detection and tracking abilities. Through our experimentation, we show that the identity and location of pedestrians can be provided through an application of the figure/ground framework. Through the automatic monitoring of pedestrians, we see how this framework provides the necessary information regardless of the size, speed, or number of target objects that the system is faced with.

CHAPTER 9

DESCRIPTION AND COMPARISON OF THE DIFFERENT SENSORS USED IN THE PEDESTRIAN CONTROL PROBLEM

THE ULTRASONIC SENSOR

The purpose of the TC-30 ultrasonic sensor is to detect obstacles within a certain distance ahead of the sensor. This can be used to feed information into a device like a traffic controller, a counting device, or an automatic gate.

The way this sensor works is by sending a sound pulse and detecting the resulting echo. If the echo is detected within a certain amount of time corresponding to the desired distance, the sensor gives a positive response. The process is repeated six times per second.

The distance within which obstacles are sought can be set to anything in the range of 24 feet. The detection area is the area inside a cone whose diameter is 4 feet at the distance of 24 feet.

THE MICROWAVE SENSOR

The TC-20 is a motion sensor: it detects motion of objects rather than the objects themselves. Moreover, it detects motion only if the direction of motion is towards the sensor.

The TC-20 works based on the same principle used in police radar. A microwave signal is transmitted and the received signal is checked for a Doppler shift which will occur only if the object that reflected the signal was moving.

The range of this sensor is larger than that of the TC-30 and has a larger cone diameter. The range can be set to anywhere between 5 and 110 feet; and the diameter is 7.4 feet at the distance of 24 feet.

VISION SENSORS

Vision sensors (cameras) provide a much larger information volume. A typical CCD camera will provide an array (called image) of 640 by 480 cells corresponding to the locations of the view captured by the camera. Each cell contains three values corresponding to the red, green, and blue components; and each value is in the range of 0 to 255 depending on the intensity of the corresponding color.

The range of a vision sensor is unlimited. Basically, it can see whatever a human eye can see. The width (cone diameter) can be controlled using lens of different focal lengths.

COMPARISON OF DIFFERENT TYPES OF SENSORS

For the pedestrian control at intersections project, we used all the above sensors to explore their usefulness. Figure 15 shows a picture of the stand on which the three sensors are mounted. They were connected to the Datacube as shown in Figure 16.

The **ultrasonic sensor** was able to accurately detect the presence of any object in the viewing area. This sensor does not give any distance information. It only gives an on/off response which makes it impossible to infer the distance to the object. This means that we cannot actually track the object using this sensor alone. One major problem with using this sensor at an intersection is that it may pick up the front of a vehicle if the vehicle was visible in the crosswalk. This means that the system may falsely assume that a pedestrian is present in the crosswalk and keep walk signal on for a long time.

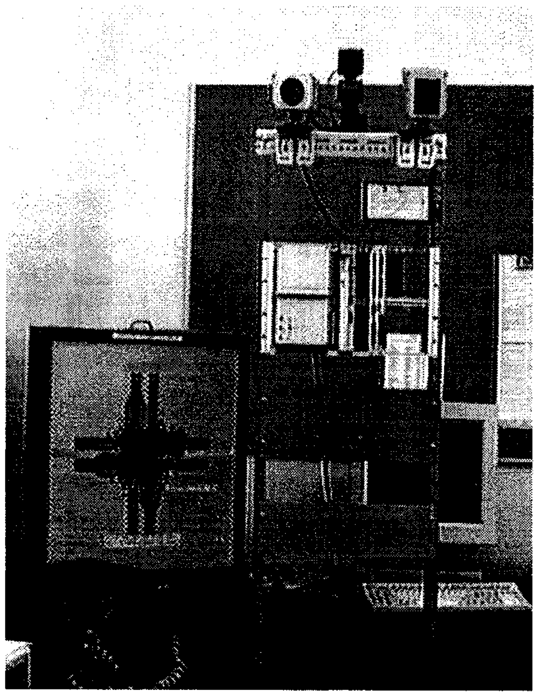


Figure 17: An Ultrasonic Sensor, a Microwave Sensor and
a Camera Mounted on an Equipment Rack

The **microwave sensor** on the other hand detects motion in the direction towards the sensor. This would require mounting one sensor on each side of the crosswalk.

Each sensor monitors the flow of pedestrians in one direction. Using microwave sensors will overcome the problem of having parts of vehicles present in the crosswalk because they would be stationary and thus will not be detected. However, a different problem is introduced: a pedestrian who is momentarily stationary or moving too slowly may not be picked up by the sensor causing the control system to assume that the crosswalk is clear and to trigger the green traffic light.

Even a combination of the two sensors still does not resolve the problem. There is no way to distinguish between a stationary pedestrian and a vehicle blocking the crosswalk because the two sensors will give the same responses in both cases: the ultrasonic sensor will give an "on" response and the microwave sensor will give an "off" response.

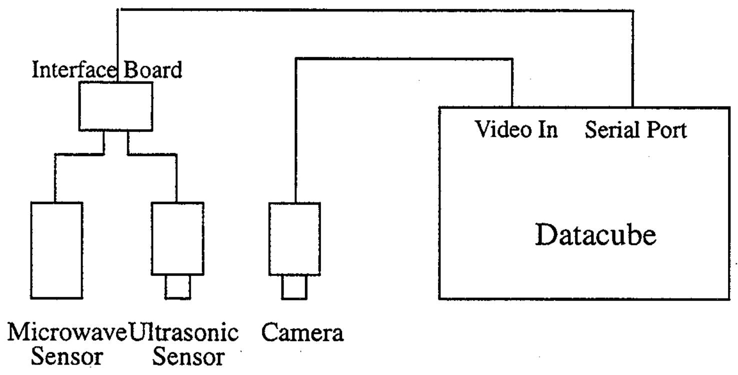


Figure 18: Different Sensors and their Connections to the Datacube System

Unlike the above two sensors which give an on/off response, **cameras** give a completely different and rich form of information as explained previously. Because of this rich image information, the use of such a vision sensor is not limited to a small number of applications. There are many ways in which this information can be processed and interpreted to extract different kinds of information. For example, even though an image itself does not contain any depth or motion information, successive images can be used to extract such information. Another kind of information which cannot be retrieved by other types of sensors is the one that is relevant to object recognition. For example, objects can be classified as vehicles or pedestrians by performing a detailed recognition procedure on the image.

For the purpose of the pedestrian control project, the use of a camera has resolved the problems mentioned above associated with the other sensors. A pedestrian is be tracked by comparing successive images and by comparing the current image to the background image. The tracker in a way locks on the pedestrian so even if the pedestrian becomes stationary, the tracker will still be able to determine that the pedestrian is present. A pedestrian model is used where the width-to-height ratio of the pedestrian is required to be in a certain range. Any object which does not satisfy this requirement is assumed to be something else. So if a vehicle comes in the way, it is distinguished from a pedestrian because it does not fit in the pedestrian model. This form of object recognition is simple yet can effectively distinguish pedestrians from vehicles.

SUMMARY

We found that the camera sensor allows us to perform a more accurate detection of pedestrian and to more effectively avoid false signals than either the ultrasonic or the microwave sensors. In addition, there are certain cases where it is impossible to determine the situation using either of the latter two sensors. We therefore feel that cameras are well suited for the purpose of this project. The other sensors can be of use in different applications like counting vehicles in the case of the ultrasonic sensor or warning systems in the case of microwave sensors.

Combining all three sensors together may seem advantageous in the sense of providing more robustness. However, we believe that we can do without the ultrasonic and microwave sensors and still obtain the same degree of robustness with cameras alone.

CHAPTER 10

GENERAL TESTING OF THE SYSTEM AND PERFORMANCE UNDER DIVERSE WEATHER 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 19.

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 20(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 20(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.

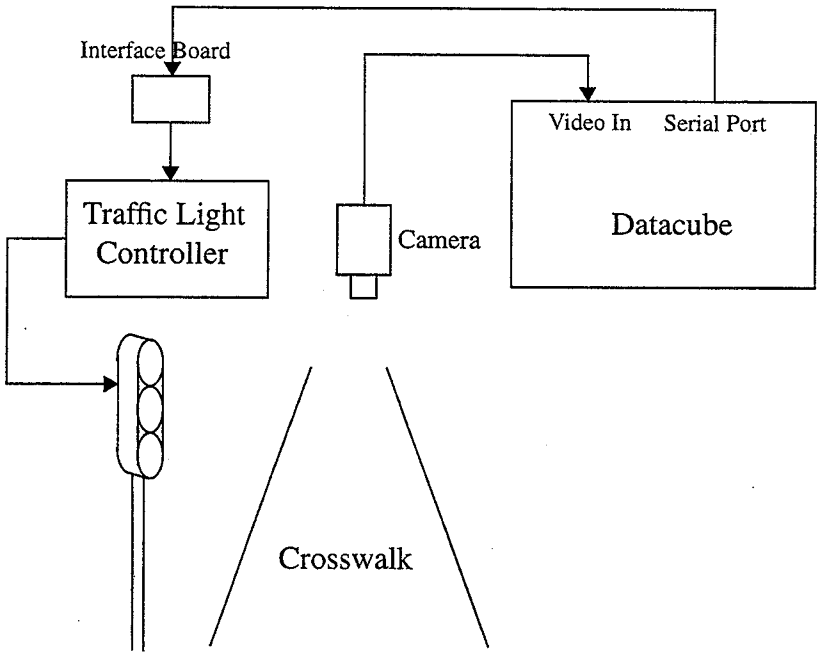


Figure 19: Pedestrian Tracking System
Components and Connections

SMALL OBJECTS

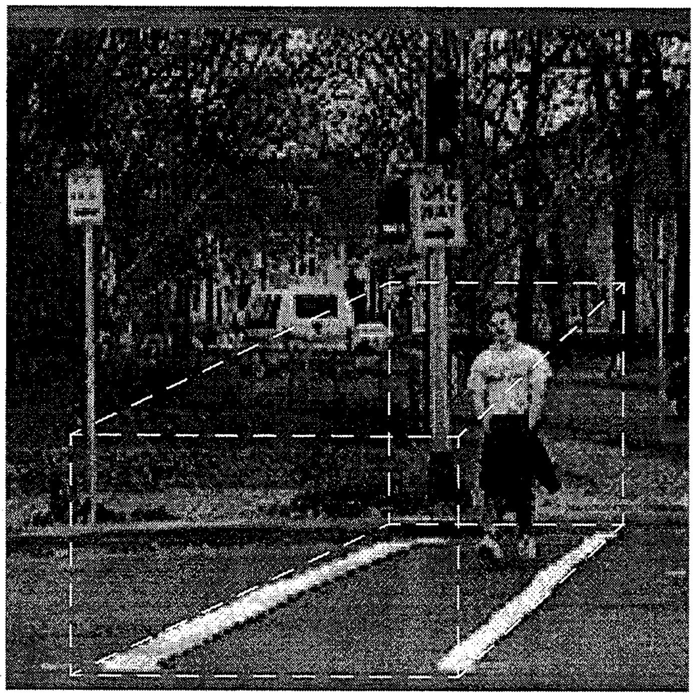
Our system is motion sensitive and therefore will pick up the motion of tree leaves due to wind. Such objects should be regarded as false. Although the interesting region is usually lower than trees, it is still possible that tree leaves may show up in the interesting region. Therefore, this case had 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

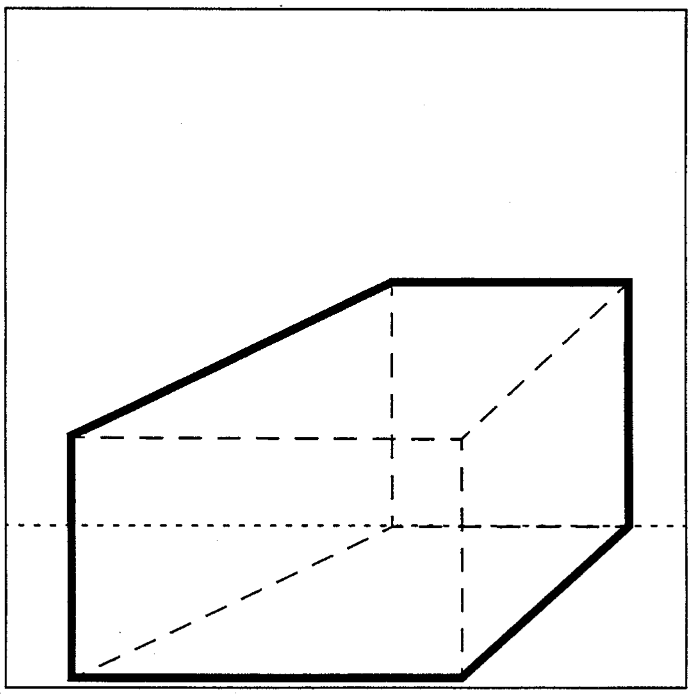
Although vehicles are not supposed to pass the crosswalk lines, they sometimes do. If we do not handle this case, the system may think that a pedestrian is positioned in the crosswalk and make the undesired response of keeping the walk signal on. Another place where vehicles may show up is in a region 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 20: 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 21 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 intersections 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 and in particular the robustness of the system to a variety of weather conditions.

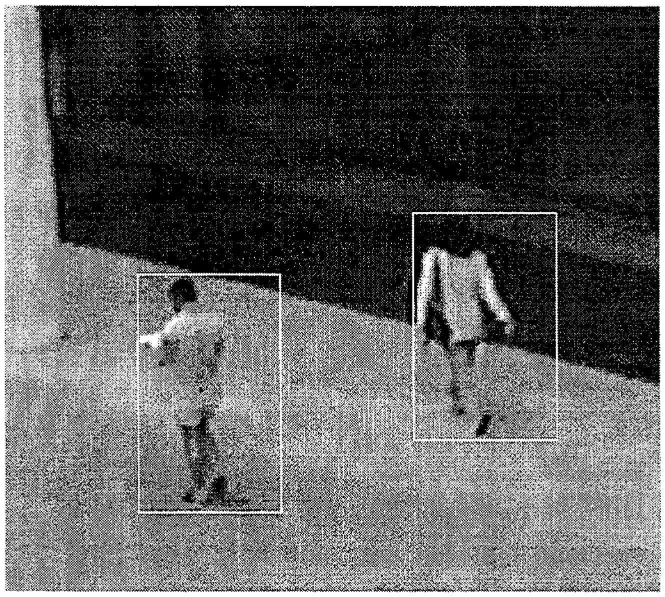


Figure 21: Shadows Become Part of the Detected Pedestrian

CHAPTER 11
CONCLUSIONS

SUMMARY OF THE REPORT

In this report, we propose robust vision-based detection and tracking techniques for pedestrian control at intersections. The report discusses how the figure/ground framework of [29] can be used to measure the trajectories of multiple objects in a temporal sequence of images. Furthermore, it is shown that results of the detection component may be used as a means for categorizing the objects in a computer's environment, allowing for the system to take action that is specific to a particular object. Experimental results are presented with a variety of objects, and the plausibility of systems that follow this framework is demonstrated. The report also compares the vision sensors with other sensors (with respect to the pedestrian control project) and presents several robustness related studies. Finally, it studies the potential use of active deformable models as an effective pedestrian tracking module.

CONTRIBUTIONS

The contributions of this work can be summarized as follows:

- We create a framework for the automatic visual detection of objects of interest. In particular, we call this the figure/ground framework.
- We develop a system which is able to perform detection according to this framework.
- We codify algorithms within this development that are fast and robust enough to be applied to the real-world vision tasks.
- We describe a scheme for using the information provided by the figure/ground framework for selectively detecting and tracking a subset of the objects encountered.

- We implement an algorithm for the automatic selection of feature windows, based upon the results provided by the visual detection module.
- We perform a theoretical study of the interaction between the detection algorithms and the tracking algorithms.
- We introduce the active deformable models in the pedestrian tracking problem.
- We test the methods extensively and evaluate other relevant sensors with respect to their applicability to the pedestrian control project.

We believe that these contributions demonstrate the potential of the figure/ground framework.

FUTURE WORK

Our research can be expanded in several different directions. The current level of technology has several limitations. Future work should address the following drawbacks of our framework:

- The frame-differencing algorithm is sensitive to large camera jitter.
- Our techniques depend on environmental conditions.
- Knowledge of figure segments is not used in our creation of ground images.
- The ground image creation process is universality applied to an entire image frame.
- Our techniques use only greyscale information from the imagery.
- The figure segments are crude compared to the output form of other methods.
- Currently, some portions of the detection framework assume that the camera is static and cannot be relied upon if a fovea transition occurs (e.g., segmentation domains and segment correspondences).

Unexpected motion of the sensor can create a large quantity of undesired figure pixels along the borders of background objects. If we properly tune the system's expectation of object size, then most of these large blobs of figure segments caused by camera motion can be ignored. However, two problems still remain. First, performance suffers due to the larger number of figure pixels that are being processed during the motion. Second, objects that may otherwise have been detected may intersect with figure segments caused by the motion, effectively rendering them undetectable.

Another concern that demands further research on the figure/ground framework is its dependence on environmental conditions. Some conditions that would disturb humans, such as heavy snowfall, would be ignored by the system (because each flake would be removed by the segmentation size threshold). However, the automated system is hindered by some environmental effects that do not disturb humans. A relatively frequent example is cloud motion. Moving clouds can cause large, undesired regions in the figure image due to sharp, fast illumination changes. Currently, such regions can only be dealt with in the same manner as those resulting from camera motion.

Another issue for further improvement involves the update of the ground image through the time-averaging process. Segmentation information might be used to increase the quality and the computational performance of the averaging. For example, information about the previous figure images can be used to limit the scope of the averaging to those pixels that lie outside the figure image.

Occasionally an object of interest will occlude something in the background in such a way that corresponding colors for the two objects have hues that are clearly different to the human observer, but whose intensity values are similar. Performing the differencing portion of obtaining a figure image with more than just greyscale information may provide better results, but it would add computational and hardware costs.

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.

Office of Research Administration
200 Ford Building, 117 University Avenue, Mail Stop 330
Saint Paul, Minnesota 55155

(612) 282-2274