2004-45

Final Report

Real-Time Collision Warning and Avoidance at Intersections

Research

Technical Report Documentation Page

1. Report No.	2.	3. Recipients Accession No.
MN/RC 2004-45		

4. Title and Subtitle	5. Report Date
Real-Time Collision Warning and Avoidance at Intersections	November 2004
	6.

7. Author(s)	8. Performing Organization Report No.
Stefan Atev, Osama Masoud, Ravi Janardan, Nikos Papanikolopoulos	

9. Performing Organization Name and Address	10. Project/Task/Work Unit No.
Dept. of Computer Science and Engineering University of Minnesota 4-192 EE/CSci Building 200 Union Street SE Minneapolis, MN 55455	11. Contract (C) or Grant (G) No. (c) 81655 (wo) 1

12. Sponsoring Organization Name and Address	13. Type of Report and Period Covered
Minnesota Department of Transportation Research Services Section 395 John Ireland Boulevard Mail Stop 330 St. Paul, Minnesota 55155	Final Report 14. Sponsoring Agency Code

15. Supplementary Notes
http://www.lrrb.org/PDF/200445.pdf

16. Abstract (Limit: 200 words)

Monitoring traffic intersections in real-time as well as predicting possible collisions is an important first step towards building an early collision warning system. We present the general vision methods used in a system addressing this problem and describe the practical adaptations necessary to achieve real-time performance. A novel method for three-dimensional vehicle size estimation is presented. We also describe a method for target localization in real-world coordinates, which allows for sequential incorporation of measurements from multiple cameras into a single target's state vector. Additionally, a fast implementation of a false-positive reduction method for the foreground pixel masks is developed. Finally, a low-overhead collision prediction algorithm using the time-as-axis paradigm is presented. The proposed system was able to perform in real-time on videos of quarter-VGA (320×240) resolution. The errors in target position and dimension estimates in a test video sequence are quantified.

17. Document Analysis/Descriptors		18. Availability Statement
Tracking Real-time systems Collision prediction	Machine vision Traffic control (transportation)	No restrictions. Document available from: National Technical Information Services, Springfield, Virginia 22161

19. Security Class (this report)	20. Security Class (this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	26	

Real-Time Collision Warning and Avoidance at Intersections
Final Report

Prepared by:
Stefan Atev
Osama Masoud
Ravi Janardan
Nikos Papanikolopoulos

Dept. of Computer Science and Engineering
University of Minnesota

November 2004

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

Acknowledgements

We are grateful to the Minnesota Department of Transportation for supporting this project through Contract No. 81655. We would like to acknowledge Hemanth Arumugam and Harini Veeraraghavan who contributed useful ideas and code in the early stages of the project. We also thank James Aamot, Antonio Fischer, Eil Kwon, Ray Starr and Daryl Taavola from Mn/DOT for their help.

Table of Contents

List of Figures

List of Tables

Executive Summary

Monitoring traffic intersections in real-time as well as predicting possible collisions is an important first step towards building an early collision warning system. We present the general vision methods used in a system addressing this problem and describe the practical adaptations necessary to achieve real-time performance. A novel method for three-dimensional vehicle size estimation is presented. We also describe a method for target localization in real-world coordinates, which allows for sequential incorporation of measurements from multiple cameras into a single target's state vector. Additionally, a fast implementation of a false-positive reduction method for the foreground pixel masks is developed. Finally, a low-overhead collision prediction algorithm using the time-as-axis paradigm is presented. The proposed system was able to perform in real-time on videos of quarter-VGA (320×240) resolution. The errors in target position and dimension estimates in a test video sequence are quantified.

Chapter 1

Introduction

Intelligent Transportation Systems (ITS) are built from technologies such as sensing, control, engineering, and computing to solve transportation-related problems. Such systems have evolved from primarily addressing traffic control problems to current applications that involve better lane design for reduced congestion, developing systems that help reduce traffic-related fatalities, and better vehicle and pedestrian monitoring systems to study flow patterns in various traffic scenarios. Such systems perform well in steady-state traffic situations like those on a freeway, but perform badly when applied to the highly unsteady flow of a busy traffic intersection.

Traffic monitoring applications regularly make use of computer vision principles to model and analyze traffic scenarios. For example, in [7], a contour tracker was used to model each moving vehicle, while the problem of tracking vehicles under challenging conditions in a freeway is addressed by [3] using a feature-based tracking system.

Studies have shown that collisions between vehicles at traffic intersections account for nearly a third of all reported crashes in the United States [5, 11]. This has led to considerable interest at the federal level in developing an intelligent, low-cost system that can detect and prevent potential collisions in real time. This report presents the various components of a vision system that monitors a traffic intersection and uses the tracking results to predict collisions over a short time interval extending into the future. Our goal is to establish the feasibility of this approach; the specific way in which these predictions can be used to prevent possible traffic accidents, however, is not addressed at this time.

The rest of this report is organized as follows: Chapter 2 presents the problem that our system addresses and explains our particular design choices. Chapters 3 and 4 present specific adaptations of low-level and high-level vision algorithms that were implemented to monitor a traffic intersection. The techniques used to predict collisions and measure vehicle dimensions are presented in Chapter 5. Finally, Chapter 6 presents and evaluates our results on a test video sequence.

Chapter 2

Problem Specification and System Overview

The purpose of our system is to monitor a traffic intersection using a live video feed from one or more cameras over extended periods of time. Possible collisions between tracked vehicles must be detected before they occur so that a timely warning may be issued. It is critical that the system runs in real-time and that it adapts to changes in the environment for the duration of the monitoring. In addition, since the system must predict collisions, we need to produce accurate estimates of both the positions and dimensions of vehicles, preferably in real-world units.

Appearance-based trackers, such as the kernel-based mean-shift tracker presented in [4], are good at following moving objects even in the presence of partial occlusions. Unfortunately, such methods usually require a target model to be manually specified and cannot delineate objects accurately. Since we cannot provide target models for the various vehicles that can enter a traffic scene in advance, and because we need the outlines of vehicles to measure their dimensions, we chose to base our target models on connected regions of foreground pixels. Connected regions (blobs) extracted from a foreground mask provide us with good object outlines, allow for automatic target identification, and are well-suited for real-time systems.

In order to classify foreground pixels, we need a background model of the observed scene. Due to the gradual changes in scene appearance over extended periods of time, we cannot use a static background model. Instead, we use an adaptive background model based on the mixtures of Gaussians segmentation method in [13]. The resulting background/foreground classifier adapts well to gradual changes in the monitored outdoor environment and allows for the detection of targets even if they are not moving — a common occurrence at traffic intersections. Occasionally, the background extraction can fail, either because of sudden illumination changes in the scene caused by passing clouds or a camera's gain control circuitry, or due to minor camera shakes resulting from road vibrations or wind load. We present efficient methods for compensating for sudden illumination changes and camera shakes. We also describe a fast implementation of the method presented in [2] for cleaning up the foreground masks.

Vehicles that move throughout the scene will sometimes occlude other moving objects, or be themselves occluded by static objects such as road-sign poles, traffic lights, etc. Such occlusions cause blobs to merge, split into smaller regions, or to disappear completely. These interactions between connected regions can either cause a single target to be visible as more than one blob in the foreground, or cause several targets to be represented as a single blob. The problem can be alleviated by making use of multiple cameras to observe the same intersection; proper camera placement can ensure that vehicles are rarely occluded in the views of all cameras. However, even

when multiple cameras are used, it is advantageous to have some means of handling occlusions in a single view. For that reason, we represent targets as sets of regions and introduce a second-level tracker that is capable of handling blob merges and splits. The second-level tracker also makes use of camera calibration data in order to estimate the position and velocity of vehicles in world-coordinates. The calibration method we use is presented in [10] and allows us to accurately map points from the image-planes of all cameras to positions on a single world-coordinate ground-plane.

It is common practice to use the centroids of connected regions to represent a target's position. Such an approach is sub-optimal, because the position of a centroid relative to the ground-plane depends on the size and orientation of vehicles and also on the particular camera placement. The last fact complicates multi-camera tracking since the centroids tracked in different camera views do not correspond to the same real-world point. We introduce a method that can identify the centers of vehicular bases on the ground plane given the outlines of the vehicles and the camera calibration data. The base centers identified by our method correspond to the same real-world point, which allows for sequential incorporation of the measurements in a target's state vector. The method also produces estimates of the width, length, and height of vehicles which are critical for the collision prediction system.

The last component of our system is the collision prediction module. Given the positions, velocities and dimensions of all targets in the scene, the module reports all target pairs that will collide within a time interval of length L in the immediate future, assuming their velocities stay constant. Optionally, the actual time of impact for each target pair that collides within the specified L time units can be reported. We present an efficient method for predicting pairwise collisions whose run-time performance does not depend on the parameter L. The method is based on the idea of "extruding" the two-dimensional vehicular bases along a time axis to obtain three-dimensional polytopes that can be tested for overlap efficiently by making use of the Separating Axis Theorem [6].

Chapter 3

Low-Level Vision System

In this chapter we elaborate on the methods used for background model maintenance, illumination filtering, camera shake compensation, and noise removal in the foreground mask. Figure 3.1 shows the components of the low-level vision system and the data interactions between them.

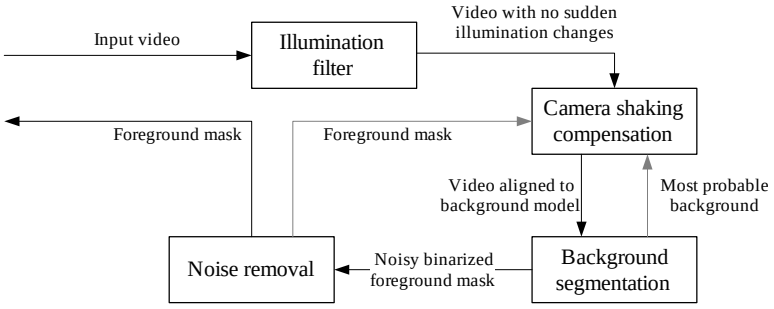


Figure 3.1: Low-level vision system components and data flow for a single frame. Lighter lines indicate the use of data from the processing of the previous frame.

3.1 Illumination Filter and Background Model Maintenance

The illumination filter runs before any other processing takes place. Its purpose is to prevent sudden brightness or contrast changes in its output, while preserving the color information in its input. The filter scales and adds an offset to all pixels in the image. The multiplicative and additive factors used are the same for each pixel component (red, green, and blue). The mean m_{in} and root-mean-square (rms) r_{in} of all pixel components (regardless of channel) are used as measures of brightness and contrast, respectively. The scale factors are chosen so that the mean m_{out} and rms r_{out} of the output image match some specified target values m_{tgt} and r_{tgt}. Finally, we obtain the target values for the next video frame according to the equations:

$$m_{\mathrm{tgt}} \leftarrow (1-\omega)m_{\mathrm{tgt}} + \omega m_{\mathrm{in}} \tag{3.1}$$

$$r_{\mathrm{tgt}} \leftarrow (1-\omega)r_{\mathrm{tgt}} + \omega r_{\mathrm{in}}. \tag{3.2}$$

The factor ω is chosen to be lower than the learning rate used in the background model update equations in order to ensure that the output of the filter never changes faster than the background

model can accommodate the changes.

The background model represents the probability of observing a 3-component background pixel as a mixture of Gaussians:

$$P(\mathbf{c}) = \sum_{i=1}^{n} \pi_i P(\mathbf{c}|M_i) = \sum_{i=1}^{n} \pi_i \frac{e^{-\frac{1}{2}(\mathbf{c}-\mu_i)^T \Sigma_i^{-1}(\mathbf{c}-\mu_i)}}{\sqrt{(2\pi)^3|\Sigma|}}. \tag{3.3}$$

The mixture weights π_i, the mixture means μ_i, and the mixture covariance Σ_i are stored for each pixel and updated at every frame. An input pixel matches a mixture component if its Mahalanobis distance [9] to the component's mean is less than a threshold. The mixture components are arranged in descending order of their $\pi_i/\sqrt{|\Sigma_i|}$ ratios. At every new frame, the index j of the first component (if any) to match the input pixel is recorded. The pixel is labeled as background if $\sum_{i=1}^{j-1} \pi_i \leq T$ for some threshold T.

If an input pixel $\mathbf{c}$ does not match any mixture component, the least probable one is replaced by a component with mean $\mathbf{c}$ and suitable initial weight and covariance. Mixture components are updated using the following transformations applied sequentially:

$$\pi_i \leftarrow (1-\alpha)\pi_i + \delta_{ij}\alpha \tag{3.4}$$

$$\mu_i \leftarrow (1-\delta_{ij}\beta)\mu_i + \delta_{ij}\beta\mathbf{c} \tag{3.5}$$

$$\Sigma_i \leftarrow (1-\delta_{ij}\beta)\Sigma_i + \delta_{ij}\beta(\mathbf{c}-\mu_i)(\mathbf{c}-\mu_i)^T \tag{3.6}$$

The factor δ_{ij} denotes the Kronecker delta. An account of our choices for the learning rates α and β, as well as all other parameters for the illumination filter and background segmentation can be found in [1]. Theoretical justification of the method is available in [12].

3.2 Camera Shaking Compensation

Camera shaking is problematic for the vision system since the static camera assumption is violated. We cannot afford to match input pixels against a neighborhood of background pixel models since each match involves the calculation of Mahalanobis distances to a number of mixture components. Even a modestly-sized neighborhood increases the computational burden considerably.

The approach we take allows us to compensate for whole-pixel translations of the input relative to the background. Due to the need for good coverage, the cameras observing the traffic are at a significant distance from the road. At such distances, the effects of minute camera pose changes can be approximated by a single translation.

To find the translation that describes a camera pose change most accurately, we match the input video frame against the most probable background image consisting of all the μ_1 mixture component means at each pixel location. The matches are performed at a fixed number of possible horizontal and vertical offsets and the one with the least sum of absolute differences yields the desired translation. The sum of absolute differences measure is used because it is not affected significantly by a small number of outliers. If the background image at a pixel location $\mathbf{p}$ is denoted as $B(\mathbf{p})$, and the input frame is denoted by $I(\mathbf{p})$, we find the optimal translation $\mathbf{q}$ as follows:

$$\mathbf{q} = \underset{\|\mathbf{q}\|_\infty \leq w}{\operatorname{argmin}} \frac{\sum_{\mathbf{p}} V(\mathbf{p})|B(\mathbf{p}) - I(\mathbf{p}+\mathbf{q})|}{\sum_{\mathbf{p}} V(\mathbf{p})}. \tag{3.7}$$

The parameter w limits the search to a small square window. The mask $V(\mathbf{p})$ is zero-valued if the pixel at location $\mathbf{p}$ was a foreground pixel in the previous frame and 1 otherwise. The mask $V(\mathbf{p})$ prevents the motion of objects in the scene from being mistaken for a camera displacement. In practice, when only a small portion of the scene is classified as foreground, the mask $V(\mathbf{p})$ can be ignored since the sum of absolute differences distance measure is robust against the small number of outliers present.

Equation (3.7) does not lead to a very efficient implementation since it requires a total of $(2w+1)^2$ full-image matches to be performed. In order to improve the performance of the camera shaking compensation, we use a hierarchical approach. We build an ℓ-level pyramid of the input image $I(\mathbf{p})$ and the most probable background image $B(\mathbf{p})$, denoted as $I_1(\mathbf{p}) \dots I_\ell(\mathbf{p})$ and $B_1(\mathbf{p}) \dots B_\ell(\mathbf{p})$ in the subsequent discussion. The first level is at full resolution (i.e. $B_1(\mathbf{p}) = B(\mathbf{p})$ and $I_1(\mathbf{p}) = I(\mathbf{p})$), while each subsequent level is obtained by down-sampling the previous one by averaging 2×2 pixel regions. The optimal translation $\mathbf{q}_j$ at pyramid level j is found using the following recurrence:

$$\mathbf{q}_j = \underset{\|\mathbf{q}_j - 2\mathbf{q}_{j+1}\|_\infty \leq 1}{\operatorname{argmin}} \frac{\sum_{\mathbf{p}} V_j(\mathbf{p}) |B_j(\mathbf{p}) - I_j(\mathbf{p} + \mathbf{q}_j)|}{\sum_{\mathbf{p}} V_j(\mathbf{p})}. \tag{3.8}$$

The search starts at the highest pyramid level ℓ (using the definition $\mathbf{q}_{\ell+1} = \mathbf{0}$), and proceeds downwards until we find the full-resolution optimal displacement $\mathbf{q}_1$. At each level, a total of 9 image-to-background matches are performed. The effective range of (3.8) with ℓ levels is the same as that of (3.7) using a window size of $2^\ell - 1$. If the images under consideration have N pixels, the number of absolute differences computed using the hierarchical method is at most $9 \sum_{i=1}^{\ell} \frac{N}{4^{i-1}} \leq 12N$. In contrast, the non-hierarchical method requires a number of absolute differences computed proportional to $(2w+1)^2 N$, which is much larger even for modest values of w. In our implementation we use 5 pyramid levels, which enable us to find translations with horizontal and vertical offsets in the range ± 31 pixels.

Once the appropriate image-to-background translation $\mathbf{q}_1$ is found, we simply shift the input image by $-\mathbf{q}_1$ to align it with the background model.

3.3 Noise Removal

The foreground mask obtained using the described background extraction method usually contains a small amount of noise, which affects the connectedness and outlines of the identified blobs. It is common to post-process such binarized background masks using a sequence of simple morphological operations, such as erosions and dilations. However, determining the appropriate number and size of such operations is hard, and a wrong choice can severely affect the quality of the resulting mask (Figure 3.3). The blob segmentation method in [2] addresses these issues.

The method evaluates the degree to which pixels in the binary mask belong to a connected region. The basic structuring element used for determining the fitness of pixels is shown in Figure 3.2(a). Arrows in the diagram represent logical *AND* operations between the indicated pixels, and the target of the arrows counts the number of operations that yield a true value. The basic elements are combined into a compound structuring element as illustrated in Figure 3.2(b) and the sum of their fitness values is the fitness value for the central pixel. The sum of fitness values over a small neighborhood of each pixel is thresholded to obtain the final binary mask.

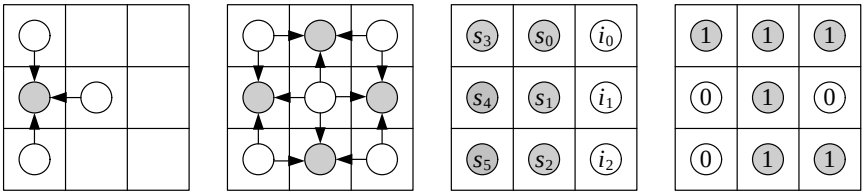


Figure 3.2: a) A single structural element; b) The compound structural element composed of 4 basic elements; c) State is a six-bit binary number $\overline{s_5 s_4 s_3 s_2 s_1 s_0}$ obtained from the first two columns, while the third column specifies the input character as the 3-bit number $\overline{i_2 i_1 i_0}$; d) A sample input with central pixel fitness 5; Before processing the central pixel, the transducer is in state 15; The input character is 5; The transition from state 15 on input 5 produces the desired output value 5 and the next state 30. *Figures a) & b) adapted from [2].*

A direct implementation of the method would require 9 memory accesses, 12 logical *AND* operations and 11 additions per pixel. To reduce the run-time overhead of the method, we based our implementation on a finite-state transducer with 64 states, an 8 character input alphabet and 12 character output alphabet. Figure 3.2(c) shows how the state and input character depend on the values of the pixels in a compound region. At each pixel, we need 3 memory accesses, 2 shifts, and 2 additions to compute the input character. A table lookup given the current state and the input character yields the desired output and next state. We combine the output and next state information in a single integer value (lower 4 bits for the output, next 6 bits for the next state). One iteration through the transducer therefore requires 1 memory lookup, 1 bitwise *AND* operation to extract the output character, and 1 shift to obtain the next state. Using this approach, we achieved a 44% reduction (2.3 ms vs. 4.2 ms on a 1.2GHz Pentium III for 320×240 pixel masks) in the per-frame processing time with respect to an optimized direct implementation of the method.

Figure 3.3: a) Foreground mask before noise removal; b) Noise removal using 1 erosion followed by 1 dilation; c) Using 2 erosions and 2 dilations; d) Using 1 erosion and 3 dilations; e) Structural denoising using a 3×3 neighborhood and fitness threshold 30.

Chapter 4

Target Identification and Tracking

The output of the low-level vision system is a foreground mask on which we perform connected region extraction. The target tracking proceeds in two stages — image-space region tracking and ground-plane tracking of sets of connected regions. We will briefly discuss the camera calibration data available to the ground-plane tracker in the following section and then describe the two tracking stages in more detail. Figure 4.1 illustrates the components of the tracking system.

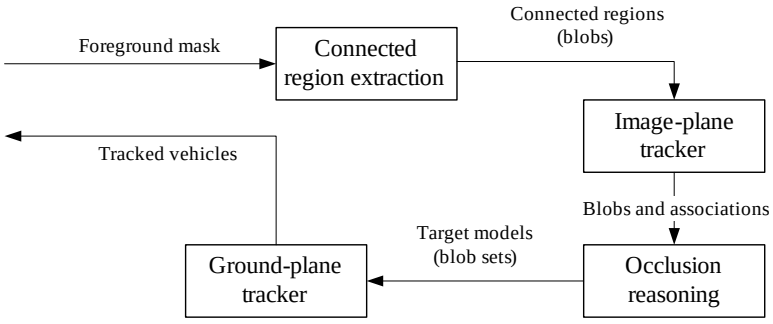


Figure 4.1: Tracking system components and data flow. The interaction between the ground-plane tracker and the collision prediction module is not indicated for conciseness.

4.1 Scene Calibration

Camera calibration is performed for a given traffic intersection using the approach outlined by [10]. Geometric primitives on the scene such as sets of parallel lines, vertical lines, and perpendicular lines are identified on a still-image view from each camera. Correspondences between points in the views of different cameras and some real-world distances between points in the same camera view are also specified. The camera calibration routine then generates the following data for each camera: a 3×3 intrinsic matrix A, a world–to–camera coordinate transformation represented by the 4×4 matrix $T_{\text{wc}} = \{t_{ij}\}$, an image-plane–to–ground-plane perspective mapping represented by the 3×3 matrix $G = \{g_{ij}\}$, and the camera's location $\mathbf{c} = [c_1, c_2, c_3]^T$ in real-world coordinates. The calibration data allows us to preform a number of computations that are relevant to the ground-plane tracker and the size estimation component of the collision prediction module.

A point (u, v) in the image-plane of a camera can be mapped to a point (x, y) on the ground plane using the following relation:

$$z[x, y, 1]^T = G[u, v, 1]^T. \tag{4.1}$$

The scalar z in (4.1) represents the perspective division. The calibration data also allows us to find the image-plane coordinates of a vanishing point $\mathbf{w} \in R^2$ in the direction of any world-coordinate unit-vector $\mathbf{d} \in R^3$ as follows:

$$F = AI_{3\times4}T_{\text{wc}} \tag{4.2}$$
$$\mathbf{r} = F\mathbf{d} \tag{4.3}$$
$$\mathbf{w} = [r_1/r_3, r_2/r_3]^T. \tag{4.4}$$

To find the length of a line segment perpedicular to the ground-plane, we need to determine the real-world height of its floating end-point $\mathbf{x}$ given the image coordinates of its projection $\mathbf{y} = [y_1, y_2]^T$ on the ground-plane (the other end-point). The height h is given by:

$$z = \frac{(\mathbf{x} - \mathbf{y}) \cdot ([f_{13}, f_{23}]^T - f_{33}\mathbf{y})}{f_{13}^2 + f_{23}^2 + (\mathbf{x} \cdot \mathbf{y})f_{33}^2 - f_{33}[f_{11}, f_{23}]^T \cdot (\mathbf{x} + \mathbf{y})}$$
$$h = z/(g_{31}y_1 + g_{32}y_2 + g_{33}). \tag{4.5}$$

The scalars f_{ij} are the components of the matrix F in (4.2).

To compute the ground-plane projection $\mathbf{y}$ of a point with whose height h and image coordinates $\mathbf{x}$ are known, we first find the ground-plane point $\mathbf{z}$ which corresponds to $\mathbf{x}$ in the image-plane (using (4.1) with $\mathbf{x}$ and $\mathbf{z}$). The projection of $\mathbf{x}$ on the ground-plane is then computed as:

$$\mathbf{y} = \mathbf{c} + (1 - h/c_3)(\mathbf{z} - \mathbf{c}). \tag{4.6}$$

4.2 Image-Space Tracker

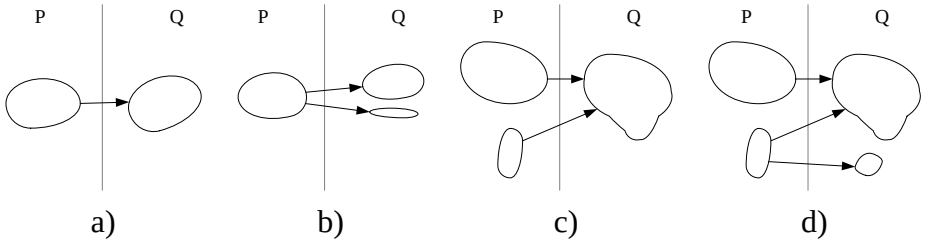


Figure 4.2: Possible blob interactions between the current frame and the previous frame. a) one-to-one; b) simple split; c) simple merge; d) split-merge conflict.

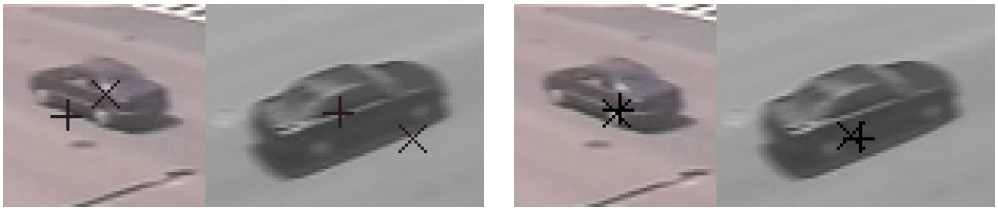
The image-space region tracker computes the fractional overlap between regions in the current frame, denoted $Q = \{(\mathbf{q}_j, \mathbf{v}_j)\}_{j=1}^{n_q}$, and the predicted position of blobs from the previous frame, denoted as $P = \{(\mathbf{p}_i, \mathbf{u}_i)\}_{i=1}^{n_p}$. The predicted position of a blob $P_i \in P$ is obtained by adding its velocity $\mathbf{u}_i$ to its position $\mathbf{p}_i$.

If the overlap is above a threshold (50%), the respective blobs are associated and the velocity $\mathbf{v}_j$ of the current frame blob $Q_j \in Q$ located at $\mathbf{q}_j$ is estimated for use in the next frame. Figure 4.2 shows the possible overlap configurations that can occur from one frame to the next one; a particular configuration is indicative of an occlusion event that must be handled as a separate case when estimating the image-plane velocity of connected regions.

Since we only need to predict the future positions of blobs in the image for one frame into the future, we make use of a simple method for estimating the image-space velocities: If a blob P_i is only related to one blob Q_j (Figure 4.2(a)), $\mathbf{v}_j$ is estimated as $0.5\mathbf{u}_i + 0.5(\mathbf{q}_j - \mathbf{p}_i)$. In the case of a split of the blob P_i into the blobs $Q_{j_1}, Q_{j_2}, \ldots, Q_{j_m}$ (Figure 4.2(b)), we set each of the velocities $\mathbf{v}_{j_1}, \mathbf{v}_{j_2}, \ldots, \mathbf{v}_{j_m}$ to $\mathbf{u}_i$. Newly appearing blobs are assumed to have zero velocity, while the velocity of a blob which has more than one predecessor (Figures 4.2(c) and 4.2(d)) is computed by taking the average velocity of all predecessors, weighted by their areas.

4.3 Ground-Plane Tracker

The ground-plane target tracker makes use of the blob associations computed by the image-space tracker. Each target model consists of a per-camera collection of blobs and camera-independent information about the ground-plane position of targets' bases, velocity, and dimensions. Position measurements from each camera are sequentially incorporated into the state vector by means of an Extended Kalman Filter, while size measurements are combined using a recursive Least Squares Estimator for each dimension independently. Figure 4.3 shows two possible approaches for vehicle position estimation — by tracking the centroids of targets, or by tracking the centers of the vehicles' bases. As indicated in Figure 4.3(a), the centroid is not a good tracking feature for multiple-camera configurations since it does not correspond to the same real-world point on the vehicle.



a) b)

Figure 4.3: Two views of a vehicle tracked by 2 cameras *independently*. Features recovered from camera 1 are indicated by a diagonal cross, while features from camera 2 are indicated by a plus sign. Each view shows the corresponding estimate from the other camera superimposed. a) Position recovered from blob centroid; b) Position estimated by finding the vehicle's base.

Tracking the bases of vehicles simplifies the problem of establishing target correspondence across views from different cameras. We restrict our further discussion of the ground-plane tracker to the single-camera case since the extension to multiple cameras is not overly complicated.

The blob collections of existing targets are updated by replacing each element with its associated regions from Q. The additional requirement that the blobs from Q are close to the target is imposed in order to keep the models tightly clustered. Targets are allowed to share blobs (e.g.,

during dynamic occlusions), so a usage counter is kept for each blob. The next phase of the tracker attempts to incorporate unused regions into a target's collection if its base center is contained by the contour of blob. This step allows targets to re-acquire regions lost due to temporary static occlusions. New targets are identified next, by selecting single unused regions that have been tracked successfully for a number of frames.

Finally, we proceed to update the state information of targets. The position and velocity of each one is first predicted. The predicted values will not be corrected for targets that share some of their blobs with others, since in that case the blob contours do not reflect a single object's outline. For the remaining targets, measurements are performed using the methods described in the following chapter.

Chapter 5

Collision Prediction

The collision prediction module is responsible for measuring the base center and dimensions of a target given its outline and for predicting potential collisions between vehicles. The specific way in which we use object outlines and calibration data for the first task is presented, followed by a description of the collision detection test.

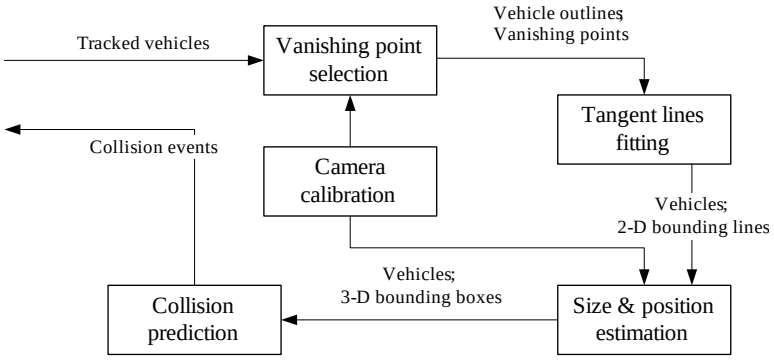


Figure 5.1: Collision prediction system components and data flow. Size and position estimates are provided to the ground-plane tracking component of the tracking module.

5.1 Dimension and Position Measurement

To measure a target's position and dimensions, we fit a three-dimensional box to its outline. The outline is the union of all contour points of regions in the target's blob collection. Assuming for the moment that we know the box's position, dimension, and orientation, we can extend its edges into lines. Those lines intersect at three distinct vanishing points in the image plane (we also handle lines parallel to the image plane as a special case.) The importance of this result is that we can reverse the process — starting with the three vanishing points and the object's outline, we can find some of the edges of the box. The vanishing points in a given direction $\mathbf{d} \in R^3$ is obtained using (4.4). The relevant directions are found by making two assumptions: that the target's orientation coincides with its direction of motion and that the bases of the targets are parallel to the ground-plane. The first assumption determines the vanishing points $\mathbf{w}_x$ and $\mathbf{w}_y$ in directions parallel to the

ground-plane: the direction of motion $\mathbf{d}_x$ and perpendicular direction $\mathbf{d}_y$. The second assumption fixes the third vanishing point $\mathbf{w}_z$ in the direction $\mathbf{d}_z = [0, 0, 1]^T$.

For each vanishing point, we find the two tangent lines to the convex hull of the target's outline. If a vanishing point is inside the outline's hull, it will be ignored in subsequent steps since all box edges that vanish to that point would be contained in the outline and thus irrecoverable. There is no need to compute the convex hull, since its vertices are a subset of the outline's vertices. The tangent lines to the hull are the lines through a vanishing point and an outline point that form the least and greatest angles relative to a fixed axis. Figure 5.2 shows a sample target, the relevant vanishing points, and the directions that determine them, as well as the tangent lines identified by our algorithm.

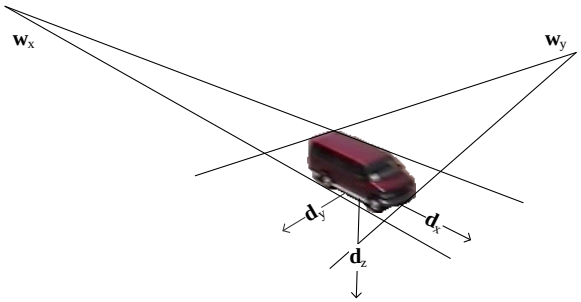


Figure 5.2: Tangent lines to a target's outline from the $\mathbf{w}_x$ and $\mathbf{w}_y$ vanishing points. The directions $\mathbf{d}_x$, $\mathbf{d}_y$, and $\mathbf{d}_z$ are indicated as well.

The end-points of the bounding box edges are found by intersecting the tangent lines. The three cases that we need to consider are shown in Figure 5.3. In the last case we cannot determine the length of two of the edges. This case does not occur if the camera is placed sufficiently high above the road plane, however, even with proper camera placement some of the dimensions may be unavailable due to partial visibility (we consider an edge partially visible if one of its end-points is within a small distance of the image borders.) Partially visible edges will not be used to produce dimension measurements, but will still be used for position measurements since the tracker requires them for every frame.

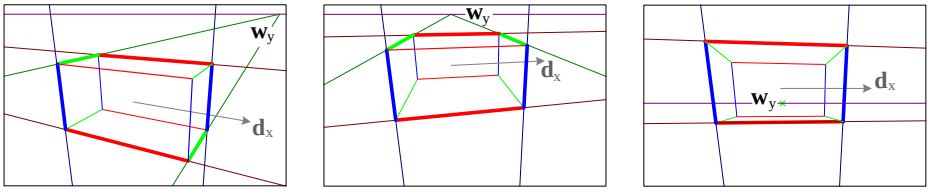


Figure 5.3: The three major cases for the tangent line intersections. Thicker line segments indicate edges whose length can be retrieved. The location of the $\mathbf{w}_y$ vanishing point and the direction of motion $\mathbf{d}_x$ are indicated. In a) and b) all three dimensions can be recovered, while in c) only two are recoverable.

The center of a target's base is obtained by taking the mid-point of a line segment connecting two edge end-points on opposite corners of the box. The real-world lengths of edges in the ground plane can be identified by using (4.1) on both end-points. The length of vertical edges can be

determined using (4.5). The recovered heights are necessary in order to determine the lengths of edges for which both end-points are vertically displaced from the ground-plane, using (4.6).

5.2 Collision Prediction

The problem considered by our collision prediction system is to identify pairs of vehicles that would to collide if they maintain their current velocities. We are only interested in imminent collisions, so we test for collisions only L time-units into the future (we used $L = 30$ frames for our tests). Vehicle heights are irrelevant for the collision test since we track them in the ground-plane. Hence, we only test the bases of vehicles for collision.

The number of vehicles that can be present at a traffic intersection is fairly limited, so the lower time complexity of advanced collision detection methods, such as those outlined in [8], does not translate to better run-time performance because of the overhead imposed by pre-processing steps and the use of advanced data structures. Thus, we opted to test all vehicle pairs in a scene for possible collision over a time interval and focused on maximizing the performance of the individual tests.

Our method is based on the idea "extruding" the bases of vehicles along a time axis. The extrusion of a rectangle moving from a point $\mathbf{p}_0$ to the point $\mathbf{p}_L$ over L time units is a parallelepiped like the one shown in Figure 5.4(a). A collision occurs if and only if the parallelepipeds representing two vehicles overlap. Figure 5.4(b) illustrates this and introduces our notation for the following discussion.

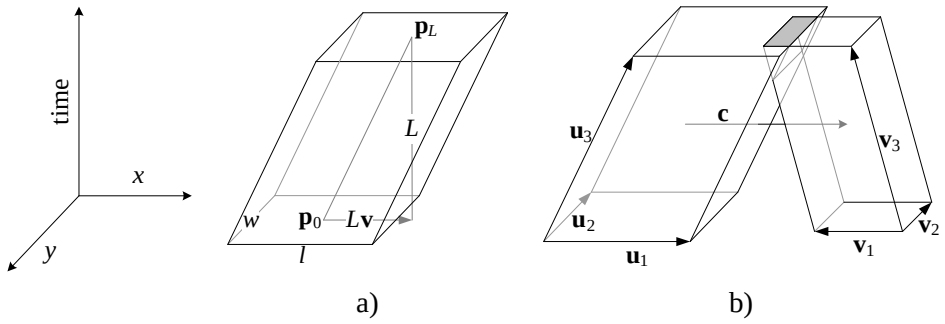


Figure 5.4: Rectangles extruded in time. a) A $w \times l$ rectangle at position $\mathbf{p}_0$, moving for L time units with velocity $\mathbf{v}$ (reaching point $\mathbf{p}_L$); b) Two overlapping parallelepipeds with labeled edges; the vector $\mathbf{c}$ connects the centroids of the polytopes.

Two convex polytopes are disjoint if there exists an axis on which their projections are disjoint. The Separating Axis Theorem [6] establishes that it is sufficient to test for overlap on axes that are perpendicular to a face from either polytope or perpendicular to edges from both polytopes. The 15 axes we need to consider are defined by the pairwise cross products of the vectors $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{u}_3$, $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ in Figure 5.4(b). The 6 pairwise cross-products of the vectors $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{v}_1$ and $\mathbf{v}_2$ are parallel to the time axis, which eliminates them from further consideration since objects always overlap on the time axis. An axis $\mathbf{a}$ chosen from one of the remaining 9 axes separates the two

polytopes if the following condition holds:

$$2|\mathbf{c}\cdot\mathbf{a}| \;>\; |\mathbf{u}_1\cdot\mathbf{a}| + |\mathbf{u}_2\cdot\mathbf{a}| + |\mathbf{u}_3\cdot\mathbf{a}| + |\mathbf{v}_1\cdot\mathbf{a}| + |\mathbf{v}_2\cdot\mathbf{a}| + |\mathbf{v}_3\cdot\mathbf{a}|. \tag{5.1}$$

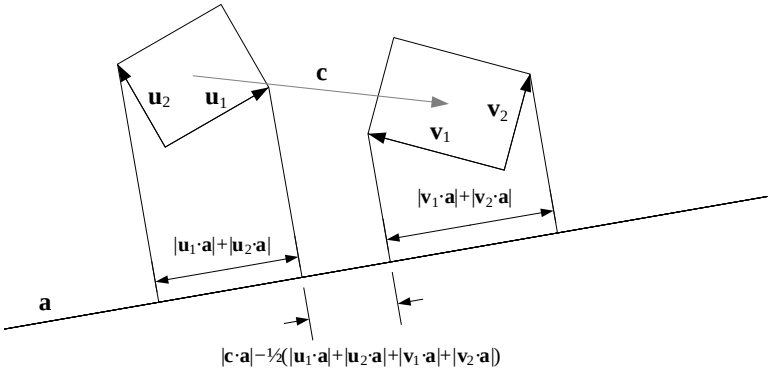


Figure 5.5: An example illustarting (5.1). In this particular case, the objects are separated by the axis.

The vector $\mathbf{c}$ in (5.1) connects the centers of the two polytopes, as shown in Figure 5.4(b). The constant factor of 2 in front of the left hand side of (5.1) accounts for the fact that each polytope's projection on the axis $\mathbf{a}$ is symmetric around the projection of its center. Figure 5.5 illustrates (5.1) for an equivalent two-dimensional example. As soon as an axis is found to separate the two polytopes, we can be sure that no collision between the respective vehicles is possible.

	$\cdot\mathbf{u}_1$	$\cdot\mathbf{u}_2$	$\cdot\mathbf{v}_1$	$\cdot\mathbf{v}_2$	$\cdot\mathbf{u}_3$	$\cdot\mathbf{v}_3$	$\cdot\mathbf{c}$
$(\mathbf{u}_1\times\mathbf{u}_3)$	0	A	B	C	0	D	W
$(\mathbf{u}_2\times\mathbf{u}_3)$	A	0	E	F	0	G	X
$(\mathbf{v}_1\times\mathbf{v}_3)$	B	E	0	I	H	0	Y
$(\mathbf{v}_2\times\mathbf{v}_3)$	C	F	I	0	J	0	Z
$(\mathbf{u}_1\times\mathbf{v}_3)$	0	A	B	C	D	0	W
$(\mathbf{u}_2\times\mathbf{v}_3)$	A	0	E	F	G	0	X
$(\mathbf{v}_1\times\mathbf{u}_3)$	B	E	0	I	0	H	Y
$(\mathbf{v}_2\times\mathbf{u}_3)$	C	F	I	0	0	J	Z
$(\mathbf{u}_3\times\mathbf{v}_3)$	D	G	H	J	0	0	U

Table 5.1: All possible values for individual terms in (5.1). Identical letters indicate identical values for the absolute value of the dot product between the vectors indicated in the first column and the vectors indicated in the first row. For example, from the table it can be seen that $|(\mathbf{u}_1\times\mathbf{u}_3)\cdot\mathbf{u}_2| = |(\mathbf{u}_2\times\mathbf{v}_3)\cdot\mathbf{u}_1|$ (both represented by an 'A' in the appropriate table cells).

It also turns out that terms in (5.1) that involve the same three vectors produce the same values, as shown in Table 5.1. This lets us eliminate 4 more axes from consideration and allows us to re-use some of the intermediate results (the terms indicated by individual letters in Table 5.1) of one axis overlap test in subsequent tests. Our final implementation of the test requires a total of 44 multiplications, 35 additions/substractions, and 5 comparisons. The resulting method for detecting possible collisions over a time interval has an almost negligible impact on our system's running time and requires no extra storage or complicated data structures.

Chapter 6

Results

Our system performs in real-time on current generation hardware (34.6 fps on a 2.66GHz Pentium IV system.) The output of the system is shown in Figure 6.1 — target bounding boxes are overlaid on the input video to allow for visual inspection of the tracking results. The foreground mask for each frame is also shown in order to highlight the behavior of the system with regard to dynamic and static occlusions. No collision warning is issued for the two occluding vehicles, since their bounding boxes are separated.

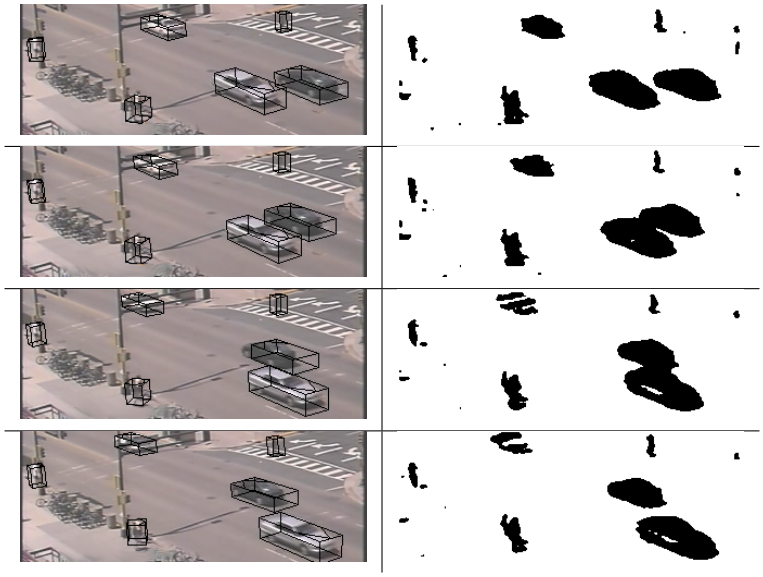


Figure 6.1: A tracking sequence at the intersection of Washington Ave. SE and Union St. SE on the University of Minnesota – Twin Cities campus. Two regions are merged in the second and third frame (dynamic occlusion). Another region is split in the third frame (static occlusion). Video of the output is available at http://www.cs.umn.edu/research/airvl/its/.

We manually counted the number of cars in 86 regularly spaced video frames. Of the 273 vehicles found, 231 (85%) were correctly identified by our system, 19 (7%) were not identified at all, and 23 (8%) were identified, but merged with other targets in the scene. We then manually fit bounding boxes around 15 different vehicles for several consecutive frames, which gave us 292 individual boxes. The root-mean squared error in the estimated quantities, as well as the average

	Position	Length	Width	Height
a)	72 cm	32 cm / 6%	82 cm / 33%	22 cm / 14%
b)	70 cm	34 cm / 7%	33 cm / 14%	23 cm / 16%

Table 6.1: Root-mean-squared errors are reported in centimeters, while average relative errors are reported as percentages. a) Results from all 15 vehicles; b) Excluding a turning vehicle whose long shadow caused severe errors in the width estimate (18 of 292 samples removed).

relative error for the three reported vehicle dimensions are indicated in Table 6.1.

The indicated accuracy of our system is significant, given the resolution at which the measurements were taken, and considering that the reprojection error of our camera calibration was as large as 40 cm in the central regions of the image.

The correctness of the collision prediction algorithm was evaluated using simulated data, since a statistically significant sample of vehicular collisions cannot be readily obtained — in the test video sequences available to us, no collisions between vehicles occurred. In the absence of severe tracking failures, our system did not issue erroneous collision warnings, despite the common occurrence of dynamic and static occlusions.

Chapter 7

Conclusions

7.1 Summary

We presented a vision-based system for monitoring traffic intersections that issues warnings about imminent collisions. The sensors used comprise of one or more fixed video cameras, calibrated in advance of the system's operation. Instabilities in the input video due to camera shaking, sudden illumination changes, and weather conditions (e.g. light rain) are handled automatically by the low-level vision components of the system. Accurate velocity, position, and size information in real-world units is obtained for each vehicle in the scene by the high-level vision components of the system. The collision prediction module makes use of this data to predict vehicle trajectories and report collisions. The data generated by the system is also suitable for vehicle classification purposes and for categorization of the severity of collisions. The system is robust to temporary static and dynamic occlusions, and is capable of handling the stop-and-go situations that occur in the setting of a traffic intersection. The proposed algorithms were implemented on general-purpose computer system without specialized vision-processing hardware. We were able to achieve real-time performance (greater than 30 Hz sampling rate) on videos of sufficient resolution.

7.2 Future Work

The presence of shadows cast from moving objects poses an interesting challenge for future research. Such shadows exacerbate occlusion problems and degrade the quality of the target outlines recovered by connected region extraction. The detection and elimination of shadows is an area of active research, but it remains to be seen if a method capable of meeting the stringent real-time performance and quality requirements of our system will be found.

Another area of interest for further research is the extension of the camera calibration techniques we use to allow for tracking of objects on non-planar road surfaces. This amounts to developing supervised algorithms for the recovery of road surfaces using common geometric primitives visible in a still image of a traffic intersection.

Finally, we would like to investigate the benefit of using more descriptive vehicle motion models in order to improve the position and velocity estimates of tracked vehicles. The noise in image-space measurements precludes the use of high-order filters, but a mixture of several high-bias filters combined using a Switching Kalman Filter may further improve the quality of tracking results.

Bibliography

[1] S. Atev, O. Masoud, and N. Papanikolopoulos, "Practical Mixtures of Gaussians with Brightness Monitoring." *Proc. IEEE Conf. Intelligent Transportation Systems (ITSC 2004),* 2004.

[2] A. Bevilacqua, "Effective Object Segmentation in a Traffic Monitoring Application," *3rd IAPR Indian Conf. Computer Vision, Graphics and Image Processing,* 2002.

[3] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik, "A Real-Time Computer Vision System for Vehicle Tracking and Traffic Surveillance," *J. Transportation Research: Part C,* vol. 6, no. 4, pp. 271–288, 1998.

[4] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based Object Tracking," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 25, no. 5, pp. 564–577, May 2003.

[5] B. Ferlis, "Analysis of Infrastructure-based System Concepts–Intersection Collision Avoidance problem area," *Unpublished FHWA document.* 1999.

[6] S. Gottschalk, M.C. Lin, and D. Manocha, "OBB-Tree: A Hierarchical Structure for Rapid Interference Detection," *Computer Graphics,* vol. 30, pp. 171–180, 1996.

[7] D. Koller, et al, "Towards Robust Automatic Traffic Scene Analysis in Real-time," *Proc. 12th Int'l Conf. Pattern Recognition (ICPR 1994),* pp. 126–131, 1994.

[8] M. Lin and S. Gottschalk, "Collision Detection between Geometric Models: A Survey," *Proc. IMA Conf. Mathematics of Surfaces,* 1998.

[9] P. Mahalanobis, "On the Generalized Distance in Statistics," *Proc. National Inst. Sci. (India).* vol. 12, pp. 49–55, 1936.

[10] O. Masoud and N. Papanikolopoulos, "Using Geometric Primitives to Calibrate Traffic Scenes," *To appear in Proc. IEEE Int'l Conf. Intelligent Robots and Systems (IROS 2004),* 2004.

[11] T. Penney, "Intersection Collision Warning System," *Pub. No. FHWA-RD-99-103.* 1999.

[12] W. Power and J. Schoones, "Understanding Background Mixture Models for Foreground Segmentation," *Imaging and Vision Computing New Zealand (IVCNZ 2002),* 2002.

[13] C. Stauffer and W. Grimson, "Adaptive Background Mixture Models for Real Time Tracking," *Computer Vision and Pattern Recognition (CVPR 1999),* 1999.