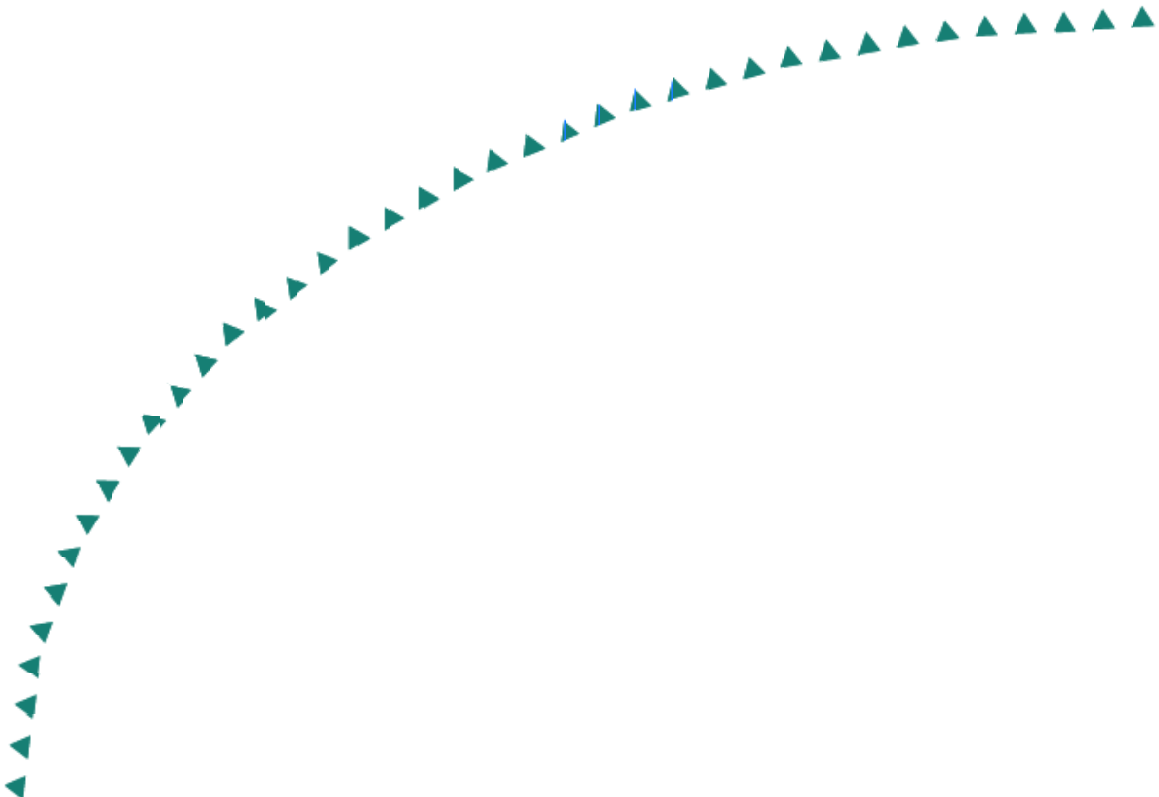
2005-40

Final Report

Development of a Tracking-based Monitoring and Data Collection System



Research

Technical Report Documentation Page

1. Report No.	2.	3. Recipients Accession No.
MN/RC – 2005-40		

4. Title and Subtitle	5. Report Date
Development of a Tracking-based Monitoring and Data Collection System	October 2005
	6.

7. Author(s)	8. Performing Organization Report No.
Harini Veeraraghavan, Stefan Atev, Osama Masoud, Grant Miller Nikos Papanikolopoulos	

9. Performing Organization Name and Address	10. Project/Task/Work Unit No.
University of Minnesota Department of Computer Science and Engineering 200 Union Street SE Minneapolis, MN 55455	11. Contract (C) or Grant (G) No. (c) 81655 (wo) 42

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/200540.pdf

16. Abstract (Limit: 200 words)

This report outlines a series of vision-based algorithms for data collection at traffic intersections. We have purposed an algorithm for obtaining sound spatial resolution, minimizing occlusions through an optimization-based camera-placement algorithm. A camera calibration algorithm, along with the camera calibration guided user interface tool, is introduced. Finally, a computationally simple data collection system using a multiple cue-based tracker is also presented. Extensive experimental analysis of the system was performed using three different traffic intersections.
This report also presents solutions to the problem of reliable target detection and tracking in unconstrained outdoor environments as they pertain to vision-based data collection at traffic intersections.

17. Document Analysis/Descriptors		18.Availability Statement
Algorithm Spatial resolution Data collection	Occlusions Target detection	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	56	

Development of a Tracking-based Monitoring and Data Collection System

Final Report

Prepared by:

Harini Veeraraghavan
Stefan Atev
Osama Masoud
Grant Miller
Nikos Papanikolopoulos

Artificial Intelligence, Robotics and Vision Laboratory
Department of Computer Science and Engineering
University of Minnesota

October 2005

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

Table of Contents

List of Figures

List of Tables

Executive Summary

This report develops vision-based algorithms for data collection at traffic intersections. Some of the problems in obtaining a robust and accurate data collection system arise from the uncontrollability of outdoor environments, apart from the placement of the camera in the scene. The spatial resolution of the targets as well as the extent of occlusions arising in the scene depends on the placement of cameras in the scene. This work tries to address some of these problems, namely; camera-placement, occlusions, and illumination changes by proposing a solution for camera-placement and robust data collection. An optimization-based approach is proposed for optimal camera-placement in a given scene. A multiple cue-based tracker with solution for data association using a joint probabilistic data association filter and a switching Kalman filter for dealing with varying target dynamics is developed. Using the results of the switching Kalman filter along with simple rules, we collect the different statistics in the scene along with the motion of targets in the scene. Extensive experimental evaluation was performed on three different traffic intersections: namely, (1) a T-intersection, (2) a one-way to two-way intersection, and (3) a four-way intersection.

1 INTRODUCTION

The goal of this project is the development of a vision-based system for automatic target tracking and data collection at traffic intersections. While vision-based systems offer a very cost-effective solution for data collection, they also suffer from difficulties in reliable target detection and tracking in unconstrained, outdoor environments such as traffic intersections. This work tries to address some of the ambiguities that plague a vision-based system, specifically, (i) occlusions, (ii) spatial resolution, and (iii) illumination variations. Problems in target segmentation due to occlusions and illumination changes, particularly shadowing are illustrated in Fig. (1.1.a) and Fig. (1.1.b).

(a) Poor placement such as oblique views can result in large occlusions between targets. Large background occlusions, spatial resolution, and illumination changes

(b) Sudden illumination changes due to passing clouds affect segmentation

Fig. 1.1. Sources of poor segmentation. Occlusions and illumination variations are the two largest sources of poor target registration in outdoor traffic scenes.

In this work, we present a systematic approach and solution to address some of these issues through:

i. Camera placement algorithms to maximize the view and spatial resolution, and to minimize the extent of occlusions in the scene, and
ii. A multiple cue-based tracking algorithm with switching Kalman filter formulation to address the ambiguities due to the aforementioned problems illustrated in Fig. 1.1, as well as to deal with the varying target dynamics.

This report is organized as follows: Chapter 2 discusses the camera-placement algorithms, while the camera calibration method, along with the guided user interface for camera calibration, is described in Chapter 3. Chapter 4 discusses the multiple cue-based tracking algorithm along with the results, and discussion of the data-collection algorithms.

2 CAMERA PLACEMENT ALGORITHMS

Occlusions and spatial resolution are two major sources of poor target registration. While occlusions render the target partially or totally invisible to the viewing camera, poor spatial resolution results in difficulty in localizing the target due to small size of the target (which can make it difficult to distinguish a real target from a faulty segmentation resulting from intervening noise). Both of these issues are dependant on the placement of the viewing camera in the scene. This is illustrated in Fig. 2.1.

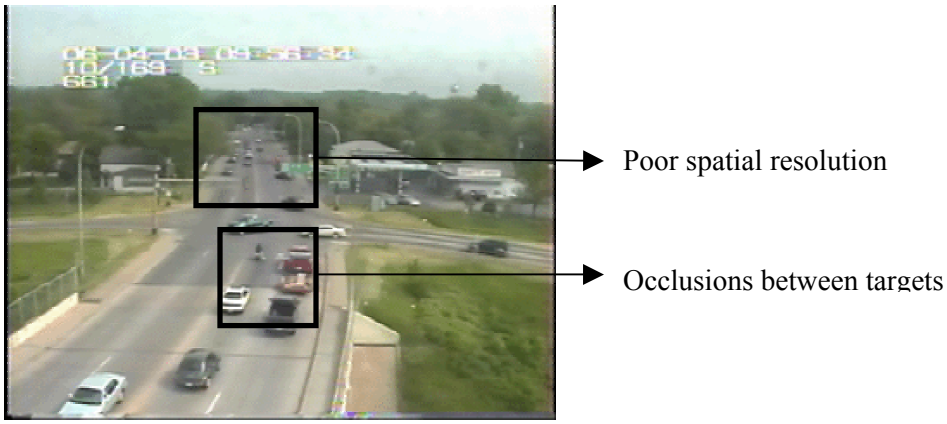
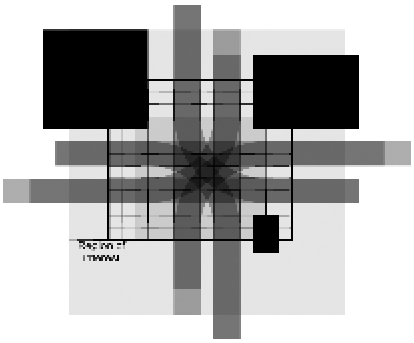


Fig. 2.1 Camera-placement.
Poor camera-placement results in poor spatial resolution and occlusions.

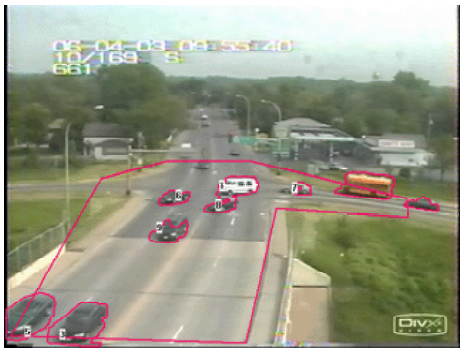
2.1 Camera Placement Algorithm

The camera-placement algorithm uses a set of possible camera locations, a density function that describes the probability of non-occupancy of a given location in the traffic scene, and a set of possible camera locations N, as $S = \{s = s_i \in R^3\}_{i=1}^{N}$, and the density function

$$V(y) \in |0,1|; y \in R^3$$



(a) Top view of intersection



(b) Image with the user specified region of interest

Fig. 2.2. Intersection views.

(a) Top view of an intersection. Regions with higher probability of target occupancy are indicated by darker regions. The region of interest is indicated by the grid, and the pure black regions correspond to artifacts, such as buildings outside the region of interest. (b) Region of interest overlaid on a traffic scene.

The region of interest is also specified as a set of points in the image (which is converted to the corresponding points in the ground plane). This is illustrated in Fig. 2.2.

The goal of the camera-placement algorithm is to maximize the likelihood of the visibility of the points inside the region of interest. It is weighted by a function that penalizes low spatial resolution. Assuming that the visibility of a point is independent from the visibility of other points in the scene, the probability that a point is observable is equal to the product of visibility probabilities along the line of sight from the point to the camera. Except at a small neighborhood, it is reasonable to assume independence and proceed by the described product rule. Calculating products of probabilities over the line of sight is slow and tedious. Since multiplication of probabilities is equivalent to additions of negative log-likelihoods, we define a new density function,

$$O(y) = -\log V(y); y \in R^3$$

Instead of maximizing a product of V values over the line of sight, we minimize the integral of O over the same line. Intuitively, O can be thought of as the "opacity" of a given point. A point inside a static object has zero visibility probability, thus infinite opacity. A location in the scene where no objects ever appear has a visibility probability of 1, hence is completely transparent (zero opacity). An exact specification of the density O is extremely tedious. Rather than specifying the density at each point, the user specifies a set of polyhedra (rectangular bounding boxes in our case), each with its own density. We will denote the set of such polyhedra as,

$$B = \{(b_k, d_k)\}_{k=1}^{O}$$

where b_k are regions of uniform density d_k. Using this formulation, the density $O(x)$ at a point x is expressed as,

$$O(x) = \sum_{k:x \subset b_k} d_k \tag{2.1}$$

As shown in Equation (2.1), the density at a point is then equal to the sum of all densities of polyhedra that contain the given point. Such decomposition is reasonable since it allows for easy specification of buildings and traffic patterns. For example, a building can be represented by a single box of infinite density. A traffic lane with 40% occupancy over time can be represented by a box of density –log(1-0.4) extending over the location of the lane, with a height reflecting the average height of vehicles passing through the lane. More complicated patterns (e.g. a 15% occupancy by cars and 5% occupancy by buses) can be specified by placing several overlapping boxes of different densities.

2.1.1 Spatial Resolution

Measurements obtained from images have a fixed error when processed using the image measurements. However, due to perspective effects, the error in actual world coordinates, depends on the distance of the point from the camera location. In other words, the measurement error at points closer to the camera will be small, while the error at points further from the camera will be large.

The spatial resolution penalty at a point $y_j \in Y$ with respect to a camera location s_k, and camera parameters p_i is defined as,

$$R(y_j, s_i, p_i) = A(y_j, s_i, p_i)^{1/2} \tag{2.2}$$

where $A(y_j, s_i, p_i)$ is the image-space area of the cell to which y_j belongs as viewed from a camera location s_i and pan, tilt, and zoom are specified by the camera parameter p_i.

2.1.2 Objective Function

The optimal camera-placement is obtained by minimizing the objective function over all possible camera configurations. We denote a camera configuration by

$$C = \{(s_k, p_k)\}_{k \in J},$$

where $J \subseteq \{1 \ldots N\}$ is an index set that specifies which camera sites chosen from *S* contain a camera. The parameters p_k denote the particular pan, tilt and zoom arrangement used for the camera located at s_k. The quality of a camera configuration *C* is:

$$t(C) = \sum_{j=1}^{M} min_{k \in J} P(y_j, s_k, p_k) R(y_j, s_k, p_k) \tag{2.3}$$

The resolution penalty, $R(y_j, s_i, p_i)$ weights the occlusion log-likelihood $P(y_j, s_i, p_i)$. The smallest of all the weighted occlusion likelihoods at any given point y_j is the contribution of y_j to the overall occlusion penalty for a given configuration *C*. Given the objective function *t(C)* we can define the optimal camera-placement configuration as:

$$C_{opt} = argmin_C t(C) \tag{2.4}$$

Such a minimization leads to combinatorial explosion. The total number of choices is $\binom{N}{|J|} \wp$, where $\wp$ is the number of discrete possibilities for the choice of an individual camera's parameters p_k. As described, the minimization of the objective function will take an inordinate amount of time. The following section proposes a method for pre-computing some quantities so that the minimization can be performed more efficiently.

2.1.3 Minimization of the Objective Function

The $P(y_j, s_i, p_i)$ terms depend on p_k only insofar as p_k determines the field of view for the camera located at s_k. The resolution penalty $R(y_j, s_i, p_i)$ is defined for points outside the field of view of the camera, but for such points the occlusion penalty is always infinity. Hence, we can rewrite $P(y_j, s_i, p_i)R(y_j, s_i, p_i)$ as:

$$P(y_j, s_i, p_i)R(y_j, s_i, p_i) = P'(y_j, s_i)R'(y_j, s_i, p_i) \quad (2.5)$$

$P'(y_j, s_k)$ is the integral of O over the line $l(y_j, s_k)$ and $R'(y_j, s_k, p_k)$ is the same as $R(y_j, s_k, p_k)$ inside the field of view of the camera at s_k, and infinity otherwise. The point of this transformation is that P' does not depend on a camera's parameters, only on the location. That means that the values $P'(y_j, s_k)$ can be pre-computed.

3 CAMERA CALIBRATION

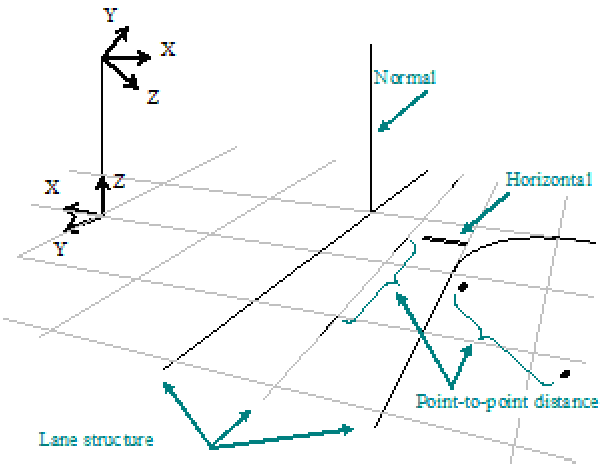


Fig. 3.1. Geometric primitives.

Common traffic scenes geometric primitives; also shown are the camera and ground plane coordinate systems.

3.1 Introduction

Images of natural and man-made environments exhibit certain regularities that are often overlooked. One of these regularities is the presence of geometric entities and constraints that bind them together. Traditionally, the structure-from-motion problem used low-level geometric entities (or features) such as points and lines with hardly any geometric constraints. Although theoretically sound, these methods suffer from two main disadvantages. First, they usually require a large number of features to achieve robustness; and second, because there are no constraints among the features, errors in localizing these features in the image propagate to the structure unnoticed. It is therefore no surprise that primitive-based approaches for reconstruction and camera calibration are on the rise [2], [3], [4], [6], [8], [9], [10], [11], [13], [17]. It is a very effective way to make use of the *a priori* knowledge in natural and man-made scenes. The primitives used can be planes, cubes, prisms, etc. and the relationships can be parallelism, orthogonality, coincidence, angle, distance, and so on.

This report presents a primitive-based approach that targets traffic scenes. Traffic monitoring applications have long been and are still interested in computer vision techniques. Unfortunately, the input data available to these applications comes from cameras that are already mounted in an outdoor setting with little known information about the camera parameters (e.g., height, zoom, tilt, etc.). The recovery of the camera intrinsic and extrinsic parameters is essential to produce measurements needed by these applications (e.g., vehicle locations, speeds, etc.). Accurate camera calibration requires the use of designed patterns to be placed in the field view of the camera.

However, in many cases, such as traffic scenes, this is not practical or even possible since one would need a very large calibration pattern, let alone having to place it on the road.

Depending on the application at hand, primitive-based methods select an appropriate set of relevant primitives [2], [6], [8], [9], [10], [11], [13]. In a similar manner, we select primitives commonly found in a traffic scene. Fig. 3.1 shows a depiction of a typical traffic scene and camera layout. The proposed primitives (lane structure, point-to-point distances, normal, horizontal, and parallel lines) are usually either obvious in the scene, are previously known properties of the scene (e.g., lane width), or, as in the case of point-to-point distances, can be measured. Our method then solves for camera parameters and scene structure by minimizing reprojection errors in the image.

A number of methods [3], [4], [17] have been proposed that addressed the primitive-based structure from motion problem as a theorem-proving and/or constraint-propagation problem. These methods can accept arbitrary geometric constraints involving points, lines, and planes, provided as a grammar. The flexibility in such methods makes them suitable for large size problems such as architectural modeling. However, these methods still need to deal with one or more of a number of issues, such as the guarantee to find a solution, computational cost, and problems arising from the presence of redundant constraints. In our case, the primitives we deal with are well defined and therefore we can choose the parameters optimally.

In [18], an interactive method was proposed to perform traffic scene calibration. Although very intuitive, it relies on the user's visual judgment rather than actual measurements. The contributions of this report are: (i) A method for calibrating traffic scenes from primitives extracted from a single image and multiple images; (ii) An error analysis of the effectiveness of using the proposed primitives by comparing our calibration results to those of a robust calibration method.

3.2 Background: Camera Calibration for Traffic Scenes

Camera calibration involves the recovery of the camera's intrinsic and extrinsic parameters. These parameters combined describe the image point (x, y) where a 3D point $\mathbf{P}$ projects onto the image plane. In a pinhole camera, this process can be expressed as

$$\begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \mathbf{ATP} \tag{3.1}$$

where $\mathbf{T} = [\mathbf{R} \mid \mathbf{t}]$ relates the world coordinate system to that of the camera through a rotation $\mathbf{R}$ and a translation $\mathbf{t}$. The matrix $\mathbf{A}$ describes the camera's intrinsic parameters which in the most general case is given by

$$\mathbf{A} = \begin{bmatrix} \alpha_u & -\alpha_u \cot\theta & u_0 \\ 0 & \dfrac{\alpha_v}{\sin\theta} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

The parameter α_u corresponds to the focal length in pixels (by pixel we mean the pixel width since it could be different from its height). In fact, $\alpha_u = fk_u$, where f is the focal length in camera coordinate system units and k_u is image sensor horizontal resolution given in pixels per unit length. The two terms are not separable and therefore, only their product (α_u) can be recovered. Throughout this report, we will refer to α_u as the focal length. α_v is similar but corresponds to the focal length in terms of pixel heights. It is equal to α_u when the sensor has square pixels. The ratio between the two is known as the *aspect ratio*. The horizontal and vertical axes may not be exactly perpendicular. The parameter θ is the angle between them. The amount by which this angle differs from 90 degrees is called the *skew* angle. The optical axis may not intersect the image plane at the center of the image. The coordinates of this intersection are given by (u_0, v_0) and are referred to as the *principal point*. In addition to these parameters, there are parameters that can be used to model lens distortion.

In this report, we make a *natural camera* assumption (i.e., zero skew angle and known aspect ratio). It is a matter of practicality to make this assumption since these two parameters rarely differ from zero and one (respectively) anyway. Moreover, of all intrinsic parameters, only the focal length changes during camera operation due to changing zoom. Therefore, other parameters could be calibrated at the laboratory if needed. The principal point is also assumed to be known (the center of the image). It has been shown [14] that the recovery of the principal point is ill-posed especially when the field of view is not wide (which is the case in many traffic scenes).

The geometric primitives that we use in this report have one thing in common: they are related through coincidence or orthogonality relationships to a plane representing the ground (see Fig. 3.1). This is similar to the *ground-plane constraint* (GPC) of [18]. Although roads and

intersections are usually not perfectly planar (e.g., they bulge upward to facilitate drainage), this is still a valid assumption as the deviation from planarity is insignificant (e.g., relative to camera height). We also make an assumption that there is a straight segment of a road in the scene.

We attach a coordinate system to the ground plane whose origin is the point closest to the camera and whose y-axis is parallel to the straight road segment (see Fig. 3.1). The primitives are essentially independent from one another and the only thing that relates them is the ground plane. Therefore, they are independently parameterized with respect to the ground plane coordinate system.

There are four degrees of freedom that relate the camera's coordinate system to the ground plane coordinate system. These may be understood as the camera's height, roll, pitch, and yaw. With the addition of focal length, this makes the total number of parameters to be found equal to five plus any parameters specific to the primitives (described below).

3.3 Geometric Primitives

A. Lane Structure Central to a traffic scene is what we refer to as a lane structure. By lane structure, we mean a set of parallel lines coincident to the ground plane with known distances among them. Given the ground plane coordinate system, we can fully specify a lane structure with exactly one variable: the x-intercept of one of its lines (see Fig. 3.1).

B. Ground Plane Point-to-Point Distances These primitives can be obtained from knowledge about the road structure (e.g., longitudinal lane-marking separation) or by performing field measurements between landmarks on the ground. Another creative way of obtaining these measurements is by identifying the make and model of a vehicle from the traffic video and then looking up that model's wheelbase dimension and assigning it to the line segment in the image connecting the two wheels. The fixed length segment connecting the two points can be fully specified in the ground-plane coordinate system by three parameters: a 2D point (e.g., the midpoint) and an angle (e.g., off the x-axis).

C. Normal, Horizontal, and Parallel Lines These primitives, represented by poles, building corner edges, pedestrian crossings, etc., are all primarily related to a lane structure. Normal lines can be specified by a single 2D point on the ground plane while horizontal (resp. parallel) lines can be specified by a y (resp. x) coordinate.

3.4 Cost Function and Optimization

The cost function is the sum of squared reprojection errors in the image. In the case of point features (such as in point-to-point distances), the meaning is straightforward. However, for line features, one has to be more careful. Many techniques that performed structure-from-motion using line features used one form or another for comparing the model and feature lines [1], [15], [16], [19]. There is no universally agreed upon error function for comparing lines. In our case, we consider the error in a line segment as the error in the two points that specify the line segment. Consequently, the reprojection error for a line segment becomes the square of the two distances corresponding to the orthogonal distances from the end points to the reprojected model line. This is advantageous since it makes it possible to combine the errors from points and lines features together in one cost function. This is also advantageous because the certainty about the location of a line is implicit in the segment length. Therefore, if only a short segment of a line is visible in the image, the user should only specify the endpoints of the visible part and not extrapolate.

The search is done on camera parameters (focal length and extrinsic, a total of five) and model parameters. The camera's rotation is represented in angle-axis form where the axis is represented in spherical coordinates. The model parameters are as follows:

1. Lane structure: one parameter (x-intercept of an arbitrarily selected line).
2. Point-to-point distances: three parameters each, with the 2D point represented in polar coordinates.
3. Normal, horizontal, and parallel lines: no parameters are needed because it is possible to compute a closed form solution in image space.

The cost-function optimization is done iteratively using the Levenberg-Marquardt method.

3.5 Initial Solution

An initial solution close to the global minimum is needed to guarantee convergence of the above optimization. Since not all primitive types need to be specified by the user, the initial solution can be computed in two different ways depending on whether one or two vanishing points can be estimated. The following sections discuss the method for estimating the vanishing points and the computation of the initial solution.

3.5.1 Vanishing Point Estimation

There are many methods for estimating the vanishing point from a set of convergent line segments. Many of these methods use statistical models for errors in the segments [7], [12], [13]. Since the vanishing points we need are used in generating the initial solution, we instead estimate the vanishing point as simply the point with the minimum sum of square distances to all the lines passing through these segments. Let $\mathbf{u}_i$ be a unit normal to the line L_i that passes through segment i's endpoints $\mathbf{a}_i$ and $\mathbf{b}_i$. Given a point $\mathbf{p}$, the orthogonal distance from $\mathbf{p}$ to L_i is $\left|\mathbf{u}_i \cdot (\mathbf{p} - \mathbf{a}_i)\right|$ or $\left|\mathbf{u}_i \cdot \mathbf{p} - \mathbf{u}_i \cdot \mathbf{a}_i\right|$. Therefore, the sum of square distances from point $\mathbf{p}$ to a set of n lines can be written as

$$\sum_{i=1}^{n} (\mathbf{u}_i \cdot \mathbf{p} - \mathbf{u}_i \cdot \mathbf{a}_i)^2 . \qquad (3.3)$$

Minimizing this sum is equivalent to solving the linear system $\mathbf{Ap} = \mathbf{r}$ where $\mathbf{A} = [\mathbf{u}_1 \quad \mathbf{u}_2 \quad \ldots \quad \mathbf{u}_n]^T$ and $\mathbf{r} = [\mathbf{u}_1 \cdot \mathbf{a}_1 \quad \mathbf{u}_2 \cdot \mathbf{a}_2 \quad \ldots \quad \mathbf{u}_n \cdot \mathbf{a}_n]^T$.

3.5.2 Initial Solution Using Two Vanishing Points

If the input primitives include a lane structure and two or more normal lines or two or more horizontal lines, two vanishing points are computed as above. These points are sufficient to compute four of the five camera parameters. The remaining parameter (camera height) can then be computed as a scale factor that makes model distances similar to what they should be. The following describes these steps in detail.

First, we compute the focal length from the two vanishing points. Without loss of generality, let $\mathbf{v}_y$ and $\mathbf{v}_z$ be the two vanishing image points corresponding to the ground's *y*- and *z*-axes. Also, based on our assumptions on the camera intrinsic parameters, let

$$\mathbf{A} = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \alpha & v_0 \\ 0 & 0 & 1 \end{bmatrix} . \qquad (3.4)$$

In the camera coordinate system, $\mathbf{p}_y = \mathbf{A}^{-1}\begin{bmatrix}\mathbf{v}_y^T & 1\end{bmatrix}^T$ and $\mathbf{p}_z = \mathbf{A}^{-1}\begin{bmatrix}\mathbf{v}_z^T & 1\end{bmatrix}^T$ are the corresponding vectors through $\mathbf{v}_y$ and $\mathbf{v}_z$, respectively (i.e., they are parallel to the ground's *y*- and *z*-axes, respectively). Since $\mathbf{p}_y$ and $\mathbf{p}_z$ are necessarily orthogonal, their inner product must be zero:

$$\mathbf{p}_y \cdot \mathbf{p}_z = 0 . \qquad (3.5)$$

This equation has two solutions for the focal length α. The desired solution is the negative one and can be written as:

$$\alpha = -\sqrt{-(\mathbf{v}_y - \mathbf{D})\cdot(\mathbf{v}_z - \mathbf{D})} \qquad (3.6)$$

where $\mathbf{D} = \begin{bmatrix}u_0 & v_0\end{bmatrix}^T$ is the principal point. The quantity under the root is the negative of the inner product of the vectors formed from the principal point to each one of the vanishing points. Note that in order for the quantity under the root to be positive, the angle between the two vectors must be greater than 90 degrees. Next, the rotation matrix can now be formed from $\mathbf{p}_y$, $\mathbf{p}_z$ and $\mathbf{p}_x$ (the latter computed as the cross product of the former two). Finally, the scale (i.e., camera height) is determined as follows. We first assume a scale of one to complete the camera parameters. Primitives that involve distances (e.g., lane structure, point-to-point distances) are then projected from the image to the ground to produce computed distances on the ground plane. Let the original (measured) and the corresponding computed distances be specified as two vectors $\mathbf{m}$ and $\mathbf{c}$, respectively. The scale s is chosen to minimize $\|s\mathbf{c} - \mathbf{m}\|$. This is simply

$$s = \frac{\mathbf{c}\cdot\mathbf{m}}{\|\mathbf{c}\|^2} . \qquad (3.7)$$

3.5.3 Initial Solution Using One Vanishing Point

When there are not two or more normal or horizontal lines, the lane structure will produce one vanishing point. In this case, three camera parameters still need to be determined: the focal length, a rotation about the vanishing direction, and camera height. Fortunately, we can deal with the latter as a last step like we did above. To solve for the former two, we try to match distance ratios between the measured distances with distance ratios between the computed distances. We use the Levenberg-Marquardt optimization method. The residual, which we try to minimize, is completely dependent on the ratios among the scene measurements. We use one measurement, m_0, as a reference and relate all other measurements to it. The residual is computed as

$$r = \sum_{i=1}^{n} \left(\frac{c_i m_0}{c_0 m_i} - 1 \right)^2 \qquad (3.8)$$

where m_i are the measured distances and c_i are the computed distances. We found that this converges rapidly and the choice of the initial values does not affect the convergence but there can be multiple solutions. However, the desired solution can always be found from any of these solutions (e.g., by negating α).

3.6 Multiple Cameras

When images of the scene from multiple cameras are available, two more constraints can be used:

a. Parallelism of lane structures. This constraint can be used if the lane structures in two or more images correspond to the same road.

b. Point correspondences. These may or may not be part of the points used to specify the primitives in the individual images.

We have not used any other correspondences among primitives across cameras (other than the coincidence of the ground plane and the direction of the lane structure). One reason is that imposing correspondence of what seems to be the same primitive may not be a good idea. Consider for example a marker pen in Fig. 3.3. The left edge of the same marker as seen in the two images corresponds to two different lines in space because the marker does not have a zero radius. With the above constraints in place, dealing with multiple cameras is straightforward. If constraint (a) above is not used, the ground planes of two cameras can be related using three parameters: a 2D point on the ground plane and an angle. Otherwise, only a 2D point is needed.

The optimization for multiple cameras is done as a final step after each camera is optimized independently. During this final step, the parameters optimized are the parameters for all cameras, the primitives in each image, and the parameters relating ground planes described above. An initial alignment of ground planes is done using one point (or two points if constraint (a) is not used) arbitrarily chosen from point correspondences. The cost function is the same as before but now also includes reprojection errors from point correspondences. So if a point $\mathbf{p}_A$ in camera A's image corresponds to a point $\mathbf{p}_B$ in cameras B's image, $\mathbf{p}_A$ is projected to the ground plane and

reprojected onto B's image where the distance to $\mathbf{p}_B$ can be computed. The same is also done in reverse.

3.7 Results

Results from actual outdoor and indoor scenes are presented. In order to evaluate the quality of the calibration parameters produced by our method, we constructed a mini-road scene, which is a scaled down version of a typical road scene in all its aspects (e.g., lane widths, lane marking lengths, etc.). The scale is approximately 1:78. We also used two cameras A and B, as shown in Fig. 3.3. Marker pens standing on their flat ends were used to represent vertical poles in the scene. The cameras are standard CCD with 6mm lens giving them a horizontal field of view of about 60 degrees. The images are captured at a 640x480 resolution. Scaling down allows us to evaluate the accuracy of calibration method. We used the robust method of Jean-Yves Bouget [5]. For calibration, several images of a specific pattern are collected, in our case, nine. One of the images had the pattern carefully placed and aligned with the road for relating the coordinate system of the road with the pattern in order to minimize the reprojection errors.

In [5], the user has the flexibility to choose which intrinsic parameters to optimize. We chose to estimate the focal length (two parameters, assuming unknown aspect-ratio), the principal point (two parameters), and lens distortion (four parameters, a fourth order radial distortion model with a tangential component). The availability of many calibration images enables us to do this. The cameras are then simultaneously calibrated to refine all parameters.

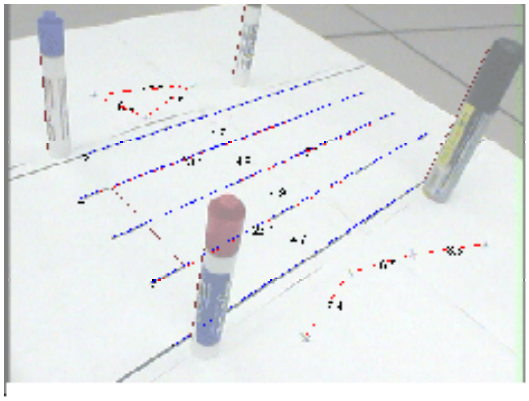


Fig. 3.2 Specification of primitives

The RMS reprojection error was on the order of approximately 0.3 pixels for both cameras. We also repeated the process but this time with the restriction on the intrinsic parameters that our method uses (i.e., a constant aspect ratio, a known principal point (image center) and no distortion model). This was done to give an idea of the expected lowest error when using a method that enforces these restrictions like ours. The results are shown in Table 3.1. Using an elaborate intrinsic model has an advantage but the restricted model is still acceptable with errors being less than one pixel.

To model this scene, we used a five-line lane structure, nine point-to-point distances, and four normals in each of the cameras. Camera B had an additional horizontal line. These primitives are shown graphically (on a shaded background for clarity) in Fig. 3.2 for camera B. After generating the initial solution, the optimization was performed on each camera individually. Then the two cameras were optimized simultaneously using eight correspondence points and a parallelism constraint on the two lane structures. Fig. 3.3 shows the results at the end of the optimization procedure. Quantitative results are shown in Table 3.2. Values under "model" correspond to the RMS reprojection error resulting from projecting the geometric primitives to the image and computing the distances to the corresponding features.

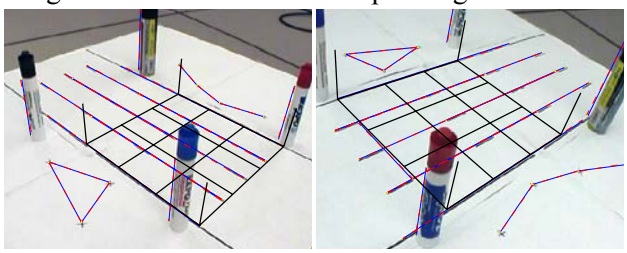


Fig. 3.3. Calibration for cameras A and B after cross-camera optimization

Notice that when calibrating multiple views simultaneously, the model error is higher than when using a single image. This is due to over-fitting noisy or otherwise insufficient features in the single image case. The combined pattern error, however, is decreased after simultaneous optimization, indicating an improvement over single-image optimization. This error is still three times larger than the best achievable but it is very small considering that a single pair of images was used to obtain it. As for the model error value of 1.25 pixels, it corresponds to approximately 10cm in the scaled-up version of this road at a point on the road near the center of the image. This is very acceptable in most traffic applications.

Camera A	*Camera B*	*Combined*
Unconstrained intrinsic model		
0.27	0.31	0.29
Restricted intrinsic model		
0.86	0.92	0.89

TABLE 3.1 RMS reprojection errors using all patterns (in pixels)

Camera A		*Camera B*		*Combined*	
Model	*Pattern*	*Model*	*Pattern*	*Model*	*Pattern*
Primitives: single image					
0.56 [a]	1.67	1.08 [a]	3.6	0.87 [a]	2.83
Primitives: stereo					
0.98	2.91	1.47	2.17	1.25	2.56

TABLE 3.2 RMS reprojection errors using primitives (in pixels)

a. do not include point correspondence errors.

Results from an actual pair of images of a traffic scene is now presented. Two images of the same traffic scene were captured by two different cameras A and B at 320x240 resolutions. The primitives used were a three-line lane structure and two normals. In addition, Camera A had two horizontal lines while camera B had one horizontal line and one point-to-point distance. The two images from the scene and these measurements are shown graphically in Fig. 3.4(a-d). Notice that the marked line segment corresponding to the middle line of camera B's lane structure is short. This is intentional since this was the only part that is clearly visible in the image and it is better not to extrapolate. The initial solution (Fig. 3.4(e-f)) is further improved after image-based optimization (Fig. 3.4(g-h)) but it still has problems as can be observed by noticing how parallelism between the overlaid grid and the shadow of the pole on the road progresses. The simultaneous optimization step uses nine point correspondences and the results from that look further improved (Fig. 3.4(i-j)). The RMS reprojection error is 2.0 pixels. This corresponds to approximately a 40cm and a 20cm distance on the road around the center of the images of camera A and B, respectively. From our experience, selecting more primitives and more accurate distances can further reduce this error.

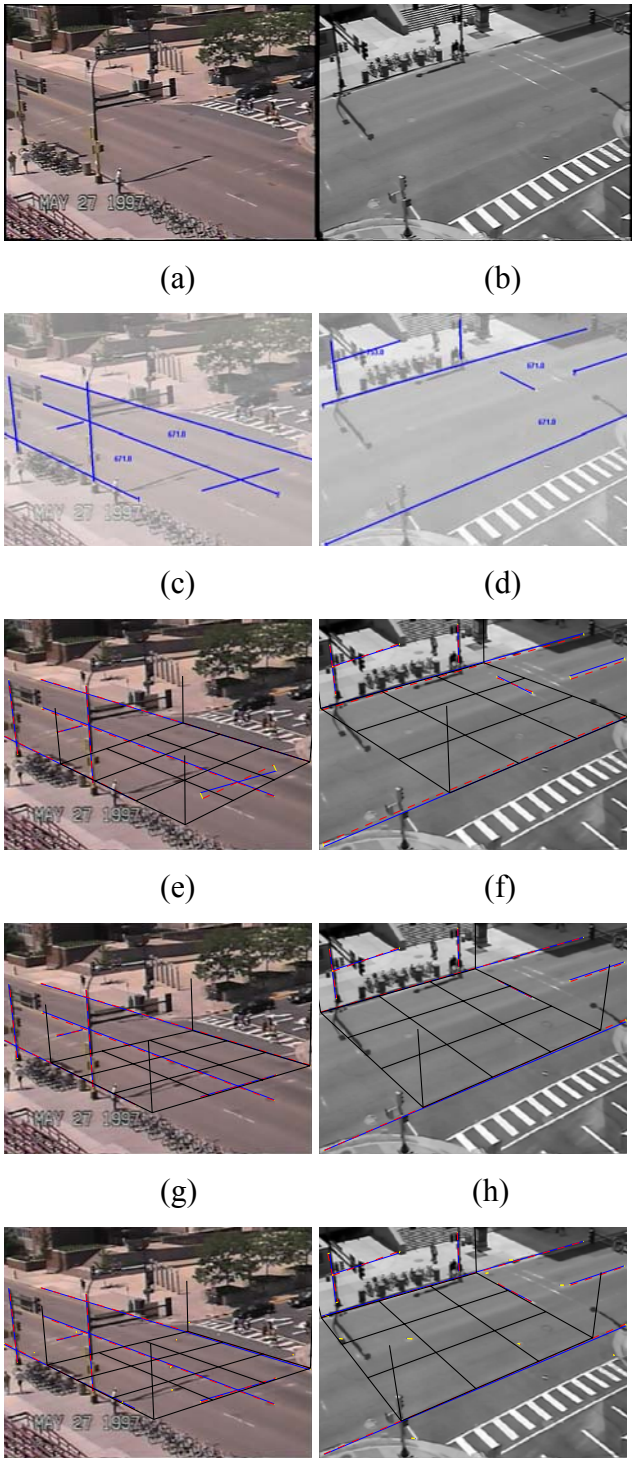


Fig. 3.4. Calibration for real traffic scenes.

3.8 Camera Calibration Guided User Interface (GUI)

We also developed a front-end user interface for calibrating any scene that needs to be used for analysis. A windowed interface as shown in Fig. 3.5 is used for this purpose. The interface allows the user to input any image (currently only .jpg and .png files are supported) and input measurements from the scene based on the primitive method as discussed in the preceding sections. The results of calibration are stored in a project file with an *.fml* file extension which should be used for running the analysis on a movie corresponding to the calibrated scene. The movie that needs to be analyzed can also be input in the GUI.

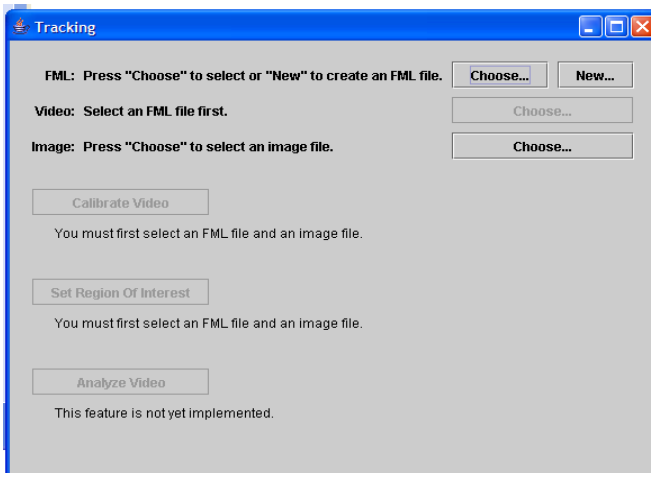


Fig. 3.5. GUI for calibration

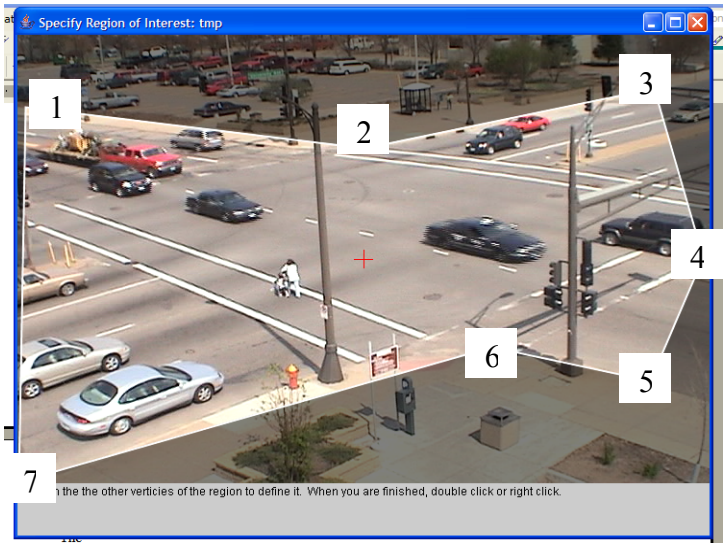


Fig. 3.6. Setting the region of interest. The brightened regions correspond to the region of interest.

The calibration GUI takes three different inputs: (i) the movie that needs to be analyzed (which is stored in the *.fml* file, (ii) a region of interest, and (iii) the image from the scene that needs to be calibrated.

Fig. 3.7(a) Setting the parallel lines. The parallel lines are indicated along with the distance between them by the connecting lines. The distance between parallel lines or lanes can be altered by the user.

Fig. 3.7(b) Perpendiculars; the perpendiculars are indicated by the lines on the poles.

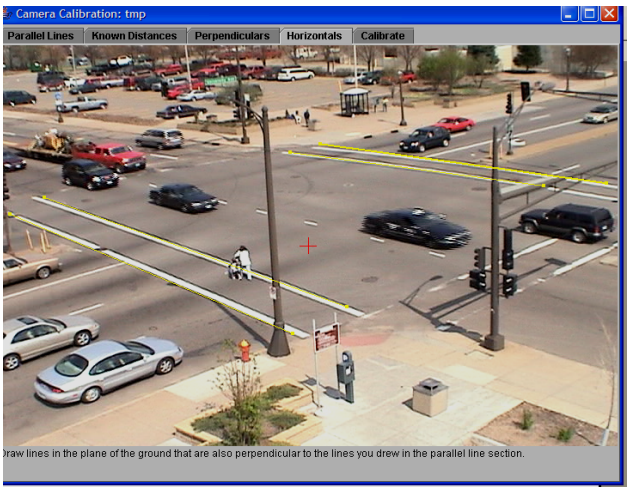


Fig. 3.7(c) Horizontals; the horizontal lines are indicated.

Fig. 3.7(d) Known measurements; known measurements with their corresponding true distances (in ft) are indicated as shown

In case of absence of a specific region of interest specified by the user, the system uses the entire image as the region of interest. The region of interest is set by clicking on successive portions of the image (in regular order as a polygon) as shown in the Fig. 3.6. After setting the region of interest, the user may also optionally alter the shape of the region.

For calibration, the user inputs measurements from the scene which consist of: (i) parallel lines (or) lanes, (ii) perpendiculars (any perpendicular entity in the scene such as traffic poles), (iii) horizontal lines (or) any horizontal markings flat on the road, and (iv) known distances, which generally consist of measured distances taken on the road whose corresponding positions are indicated in the image, as shown in Fig. 3.7(a), Fig. 3.7(b), Fig. 3.8(c), Fig. 3.9(d).

4 MULTIPLE CUE-BASED TRACKING AND DATA COLLECTION

4.1 Introduction

While a vision-based system has the advantages resulting from economy and the corpus of information made available from a single source, it also suffers from several shortcomings especially in unconstrained environments such as outdoor traffic intersections. The accuracy of the tracking system depends on the accuracy with which target positions can be recovered. This is affected adversely by sources such as varying illumination, resulting in shadowing, background clutter and high traffic density manifesting as occlusions, and the motion of vehicles such as turning which results in pose variations with respect to the camera. For obtaining any reliable information from an automated data collection system, the effect of the above-mentioned sources on the target tracking system must be minimal.

Most methods for outdoor tracking try to use a single cue that can provide good measurements under certain conditions in the environment. Examples of methods employed in outdoor scenes include, [22], [23], [24], [25], and [26]. However, single cue-based trackers have the problem that they are reliable only as long as the assumptions they are supposed to work under remain true. As soon as the scene changes in ways that violate these assumptions, the cue fails to provide meaningful measurements about the targets of interest. Multiple cue-based methods, such as [27], [28], overcome this problem due to the fact that different cues fail at different conditions thereby increasing the operating range of the system.

In this project, we present a method for obtaining a reasonably accurate tracking system by making use of multiple cues such as blobs obtained through an adaptive background subtraction method, based on [29], and the color of the tracked targets (vehicles). Both blobs and color are used for obtaining a target's position in each frame. Blobs are tracked from frame to frame using a blob tracking method and the color of the targets are used for localizing them in subsequent frames using a mean shift tracking procedure based on [23]. The cues are then fused sequentially using an extended Kalman filter.

First-order motion models, such as a constant velocity motion model, are used widely for tracking vehicles in traffic scenes. While these models are robust in comparison to higher order motion models such as constant acceleration models, they are accurate only as long as the vehicles whose motion they model follow more or less constant velocity motion paths. This is not true for turning or stopping vehicles that are fairly common in traffic intersection scenes. Hence, we use a

switching Kalman filter framework, which provides estimates of a vehicle's position, velocity and acceleration by a weighted combination of three different filter models, namely, a constant position, constant velocity, and constant acceleration model.

This chapter is organized as follows: Details of the individual tracking modalities, namely, the blob, and mean shift tracker are in Section 4.2. The brief theory of the switching Kalman filter is in Section 4.3. Algorithms for collecting the different statistics, such as, vehicle trajectories, mean speeds, accelerations, and other statistics such as the number of left-turning, right-turning, stalled and over-speeding vehicles are discussed in Section 4.4. Section 4.5 presents the results of the data collection algorithm for three different traffic intersections, and the discussion of the results is in Section 4.6.

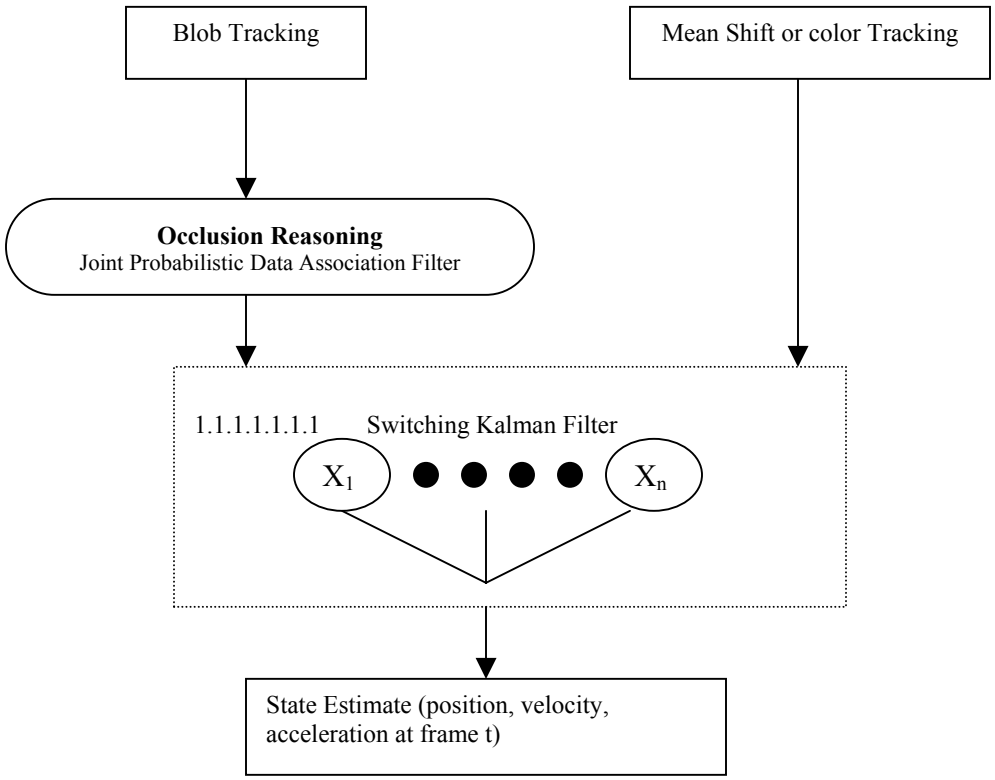


Fig. 4.1. Tracking approach. Position measurements from blob and mean shift tracking are combined using a switching Kalman filter to generate the position, velocity and acceleration estimates of the target at frame t.

4.2 Tracking Methodology

Scene interpretation in natural environments using a single visual cue is difficult owing to the increased ambiguity presented by the scene. Adaptive integration of multiple sources of information has two advantages, namely, easier ambiguity resolution, and efficient operation of the system under a wider range of environmental conditions. While the introduction of additional components for scene interpretation increases the complexity and computational requirements of the system, the robust tracking solution yielded by such a system offsets the inadequate and limited information provided by single cue trackers.

The cues used for tracking consist of the target color distribution (represented by its color histogram), and the foreground motion blobs (obtained through adaptive background segmentation). Both the cues return the position of targets, namely, vehicles, which are integrated sequentially in a Kalman filter framework. Details of blob tracking, color based target localization and the cue integration method are discussed in the following sections. The tracking method is as described in Fig. 4.1.

4.2.1 Blob Tracking

Blob (segmented foreground region) tracking is used for the purpose of target detection and tracking. The details of the segmentation and tracking method are outlined in our previous work [31]. While computationally trivial, blob tracking based methods pose a problem due to reduced information content, and the resulting ambiguity in data association as well as false target detection. The reduced information content results from the abstraction of information solely as foreground or background. This is illustrated in Fig. 1.1(b). We try to address the data association ambiguity based on a joint probabilistic data association method as described in [20]. In order to reduce the computational complexity of the method due to complete target-data association, measurements are gated based on the associations between blobs in two consecutive frames.

The blob tracking method is a lower level method that uses the blobs detected in the current frame to detect any associated blobs from the previous frames. Association is defined as proximity in the spatial location of the blobs. The results of the tracking are then used in the joint probabilistic data association filter, which computes the relative association of each blob to each vehicle. In essence, a vehicle's measurement consists of a weighted combination of blob measurements, weighted by their proximity to the vehicle's predicted position and position error covariance (predicted using the Kalman filter based on previous position estimates).

4.2.2 Color: Mean Shift Tracking

A target's color is modeled as a histogram across three channels (normalized R, G, and B). The tracking method is based on the mean shift tracking approach proposed by Comaniciu et al. [23]. The target model is represented by *m* histogram bins, which are normalized by re-scaling the rows and columns of the target's bounding box (computed around its blob) to eliminate the influence of dimensions. Given the target model p_j, the tracking method consists of searching for the closest target model around the predicted target position. The similarity of a model around a new position and the target model is computed based on the Bhattacharya coefficient, which can be expressed as,

$$g.\rho\left[p(x), q\right] = \sigma_{j=0}^{m-1}\sqrt{p_j(x)q_j} \tag{4.1}$$

where p_j and q_j correspond to the target candidate and the target model respectively.

As the model is initialized automatically using the axis-aligned bounding boxes of the vehicle candidate blobs, errors can be introduced in target localization due to errors created by poor representation based on axis-aligned boxes. For achieving better localization, we use a simple heuristic wherein, only the region inside half the dimension of the bounding box around the center is used for the target model formulation. Furthermore, the target histogram is weighted relative to the background such that, the portions of histogram distinct from the background are weighted higher than the portions similar to the background color distribution.

4.2.3 Cue Integration

Different methods have been proposed in previous literature for combining multiple cues. Examples include democratic integration [35] where the cues are weighted equally and combined. However, this is not a realistic way of combining cues in outdoor scenes due to different reliabilities of cues. [34] discusses a method for integration of cues weighted by their reliability. This approach ties in with the Kalman filter where cues are integrated into the filter weighted by their measurement error covariance, which is inversely proportional to the confidence of a measurement.

The standard method for updating *n* measurements available from *n* different sensors in a Kalman filter consists of updating all the measurements in batch mode. This procedure makes the Kalman filter update step very computationally expensive. However, as long as the measurements are

uncorrelated, the measurements can be updated sequentially as shown in [21], thereby making the procedure more computationally simple. Given that blob-based position measurements and the color tracking-based position measurements are uncorrelated to each other, we can combine the two measurements sequentially in two update steps.

4.3 Switching Kalman filters

Kalman filters are widely used estimators for systems that obey a linear model and are Gaussian, that is, the noises entering the system are Gaussian. In the case of the given scenario, namely, vehicle tracking in traffic intersections, non-linearities are introduced due to the mapping of the measurements from image (or) pixel space to the scene coordinates due to the non-linear transformation. In these cases, locally linearized filter approximations such as extended Kalman filters, Gaussian sum filters, and iterated Kalman filters are frequently used. However, these approximations work only so long as the probability density function *pdf* of the system is Gaussian. In the case of vehicles in traffic intersections, vehicles exhibit different motion patterns due to stopping, acceleration or deceleration during turning, uniform motion when moving on a straight path etc. Hence, using a single model, such as a constant velocity model for example, to describe the motion of a vehicle throughout its trajectory can be inappropriate. This, for instance, can yield inaccurate results for estimates of velocity, acceleration, and other factors as well lead to track divergence.

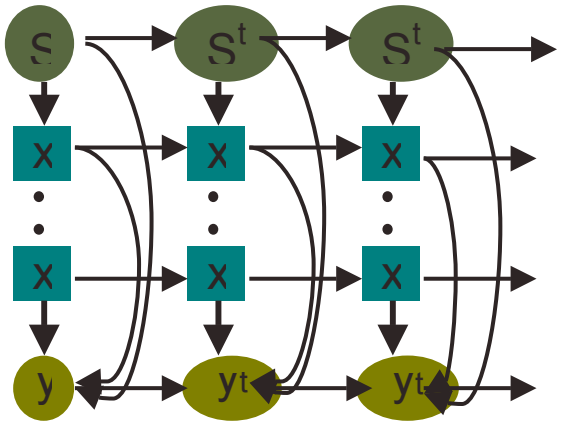


Fig. 4.2. Switching Kalman filter.

Switching Kalman filters mitigate this problem by allowing the state space to be a mixture of Gaussians. These are essentially switching state space models, maintaining a discrete set of (hidden) states and switch between or take a linear combination of these states. These can be described as illustrated in Fig. 4.2.

As depicted in Fig. 4.2, a state space model or switching Kalman filter, maintains a set of states $X_1, \ldots, X_N$ at a given time t. A switch variable S^t is used for switching between different states thereby allowing for the option of modeling different operating conditions of the system. The switch variable contains the probability of each model. Thus at any given time t, the state estimate, is computed as a combination of all the states weighted by their probability indicated by the switch variable S^t. The switch parameter is computed as

$$\begin{aligned} E_t^{t-1,t}(i,j) &= P(S_{t-1}=i, S_t=j \mid y_{1:t}) \\ &= \frac{\mathrm{L_t}(i,j)\Phi(i,j)E_{t-1}^{t-1}(i)}{\sum_i \sum_j L_t(i,j)\Phi(i,j)E_{t-1}^{t-1}(i)} \end{aligned} \tag{4.2}$$

$$E_t^t(j) = \sum_j E_{t-1}^{t-1}(i,j) \tag{4.3}$$

$$\begin{aligned} W^{i|j} &= P(S_{t-1}=i \mid S_t=j, 1:y_t) \\ &= \frac{E_t^{t-1,t}(i,j)}{E_t^t(j)} \end{aligned} \tag{4.4}$$

where i, j corresponds to filter models. The term $L_t(i,j)$ corresponds to the innovation likelihood computed as the probability of the filter residual (discrepancy between a filter's prediction and observed measurement), given the innovation covariance (computed using the filter's predicted state covariance and the measurement error covariance). $\Phi(i,j)$ corresponds to the conditional probability of the switch variable being *j* at time *t*, given that the switch variable at time *t – 1* is *i* given measurements from *1: t –1* and expressed as,

$$\begin{aligned} \Phi(i,j) &= \mathrm{P}(S_t=j \mid S_{t-1}=i, 1:y_{t-1}) \\ &= \frac{P(1:y_{t-1} \mid S_t=j, S_{t-1}=i)P(S_t=j \mid S_{t-1}=i)}{P(y_{1:t-1} \mid S_{t-1}=i)} \\ &= \frac{L_{t-1}^{(i,j)}P_t^{t,t-1}}{E_{t-1}^{t-1}(i)}. \end{aligned} \tag{4.5}$$

The probability $P(S_t=j \mid S_{t-1}=i)$ corresponds to the switching transition probability. This probability is assumed to be a constant for any two states and is pre-computed offline.

4.3.1 Switching Filter Models

Vehicles are described using three different motion models, namely, a constant position, constant velocity, and constant acceleration motion model.

Constant position: $[x\ y]$ where x and y correspond to the position of a target in the scene coordinates. This model assumes that the position of the target is stationary and the position estimates are corrupted by a small Gaussian noise. This model is true for stationary targets such as vehicles stopping at intersections.

Constant velocity: $[x\ y\ \dot{x}\ \dot{y}]$ where x and y correspond to the position of a target in the scene, while $\dot{x}$ and $\dot{y}$ correspond to velocities in x and y directions. This model assumes that the targets are moving with a uniform speed. This is generally true for vehicles moving without stopping or turning on straight stretches of road.

Constant acceleration: $[x\ y\ \dot{x}\ \dot{y}\ \ddot{x}\ \ddot{y}]$ where x and y correspond to the position of a target in the scene, while $\dot{x}$ and $\dot{y}$ correspond to velocities in x and y directions, and $\ddot{x}$ and $\ddot{y}$ correspond to the accelerations. This model allows the targets to accelerate and decelerate. This behavior is exhibited by vehicles coming to a stop, moving from a stopped position, or while turning.

For switching, the state transition probabilities were pre-computed using trial and error. The state transition matrix is given by

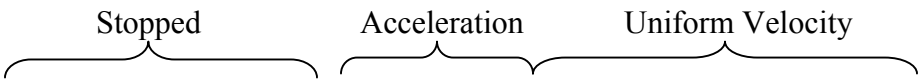
$$A = \begin{bmatrix} .65 & .13 & .22 \\ .09 & .65 & .26 \\ .17 & .18 & .65 \end{bmatrix}.$$

4.3.2 Vehicle Mode Detection

The following modes are assumed to characterize the motion of a vehicle in the given image sequences:

- Uniform motion or moving with a constant velocity
- Accelerating or decelerating motion
- Slow moving or stalled
- Turning right, and
- Turning left.

Modes such as slow moving, uniform velocity or acceleration help to glean interesting information about the nature of traffic in the given image sequence. For instance, vehicles operating in the slow-moving motion mode can be used as an indicator for congestion in the scene. The motion modes such as slow moving or stopped, uniform velocity and acceleration or deceleration mode are recovered based on the filter probabilities. In other words, the mode corresponds to the filter with the highest probability at a given time. For instance, if the constant position filter has a highest probability (given by the switch variable) at a given time t, the motion at time t is marked as "stopped." Since the motion mode derived at just one time instant can be too noisy to be used as an estimate for a vehicle's motion over a certain length of time, we collect samples of motion modes at time instants separated by a certain time interval. The motion modes are then collected as runs, from which the state of the target can be easily inferred. An example run for a vehicle can look like:



SSSSSSSSSSLLSSRSSSSSAAAAAAAAAAUUUUUUUUAAUUUUAUAUUU

S – stopped or slow moving, L – left turn, R – right turn, A – accelerating/decelerating, U – uniform velocity.

As shown in the above example run, the state of the motion can be inferred as passing from stopped (long periods of S), followed by acceleration (long consecutive A strings), followed by a constant velocity (long U strings) motion. Using runs to analyze the motion provides a simple scheme for representing the motion and the state can easily be inferred by looking for a certain length of a specific motion type over a certain window.

The reason for using samples separated by a time window is twofold. Firstly, the local motion between two consecutive time intervals, say, t and $t + 1$, is not going to give any useful information about the global motion of the target and is also prone to be noisy. For instance, if the measurement at time t was corrupted due to an occlusion, the chances that the measurement at time $t + 1$, for the same target to be corrupted are higher resulting in noisy estimates as opposed to the case of sampling the motion separated by a time window say, $t + w$. Secondly, it is not possible to recover any useful turn direction information from two very closely spaced samples as the position vectors are very closely spaced to each other thereby providing no information about the turn direction. Details of the turn-detection algorithm are discussed in the following section.

4.3.3 Turn Detection

The basic method for detection of the turning direction consists of computing the resultant vector connecting the positions sampled at three different time intervals, t, $t + w$, $t + 2w$. The resultant vector connecting the position at time t and $t + 2w$, gives the direction of motion of the vehicle.

From the angle of the resultant vector, the direction of motion can be obtained. The angle of the resultant vector has a specific relation to the turn direction as shown in Fig. 4.3. As shown in Fig. 4.3, a motion is classified as left, right, or straight based on the following relation:

$$\begin{cases} if \ \ |a| > ta \wedge |a| < \pi/2, \ \text{then left} \\ if \ \ |a| > \pi/2 \wedge |a| < \pi - ta, \ \text{then right} \\ \qquad\qquad \text{else, straight} \end{cases} \tag{6}$$

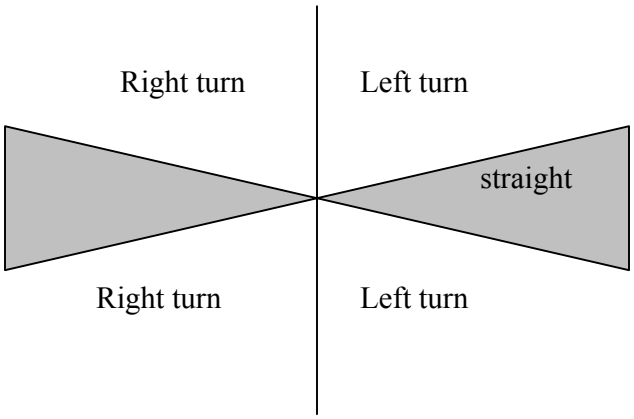


Fig. 4.3. Direction of motion related to angle of resultant vector.

4.4 Trajectory Classification

The result of applying the switching Kalman filter to the trajectory involves the smoothing of the trajectory. Every vehicle's extracted trajectory is classified into one of the twelve directions, namely, moving south to north, north to south, west to east, east to west, as well as turning left in south to west, west to north, north to east, east to south, and right turning in south to east, east to north, north to west, and west to south.

For this purpose, we make use of a clustering algorithm based on Zelnik-Manor and Perona's clustering algorithm [36]. The input consists of a collection of trajectories extracted from the scene over a certain period of time. The result of the algorithm is in grouping closely related trajectories in a single cluster. Fig. 4.4 shows the result of grouping a collection of trajectories into different groups which are then classified into one of the twelve directions.

The main limitation of this approach is that the robustness of the classifier is directly related to the number of trajectories used for clustering. Hence, greater the number of trajectories used for classification, the better the accuracy. The problem with using a small number of trajectories is that clusters with large variances can be produced, as a result of which trajectories moving in unrelated directions can be grouped together.

Classification of the trajectories is done per cluster. This is based on the assumption that all the trajectories in a cluster will be moving in the same direction. For classification, a linear function (line) is fit to the set of trajectories and using the angle and direction of motion computed from the trajectories is used to classify the motion. This approach differs from the string-based approach described in the previous section in that, the former is based on producing classifications based on grouping similar trajectories, while the latter is based on computing classification based on local motions inferred from a single trajectory.

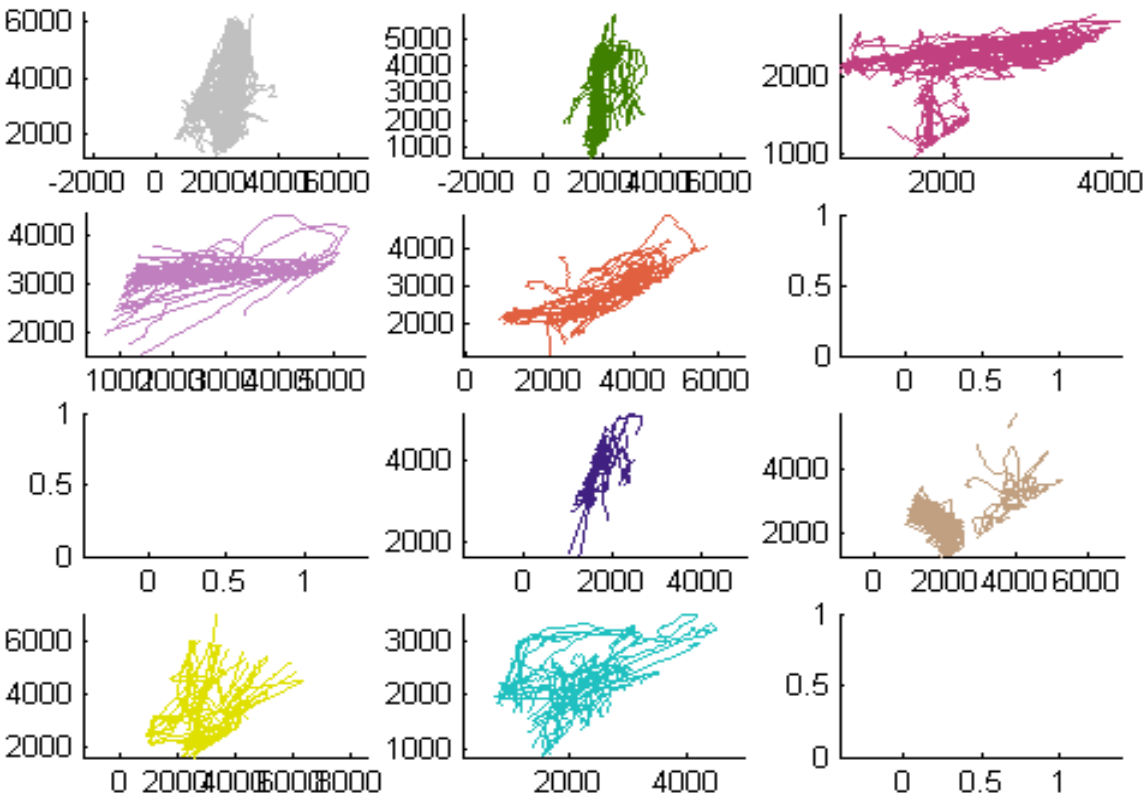


Fig. 4.4. Results of clustering and trajectory classification. For plots showing the trajectories, the units are in the scene coordinates expressed in cms in both x and y axes.

4.5 Results

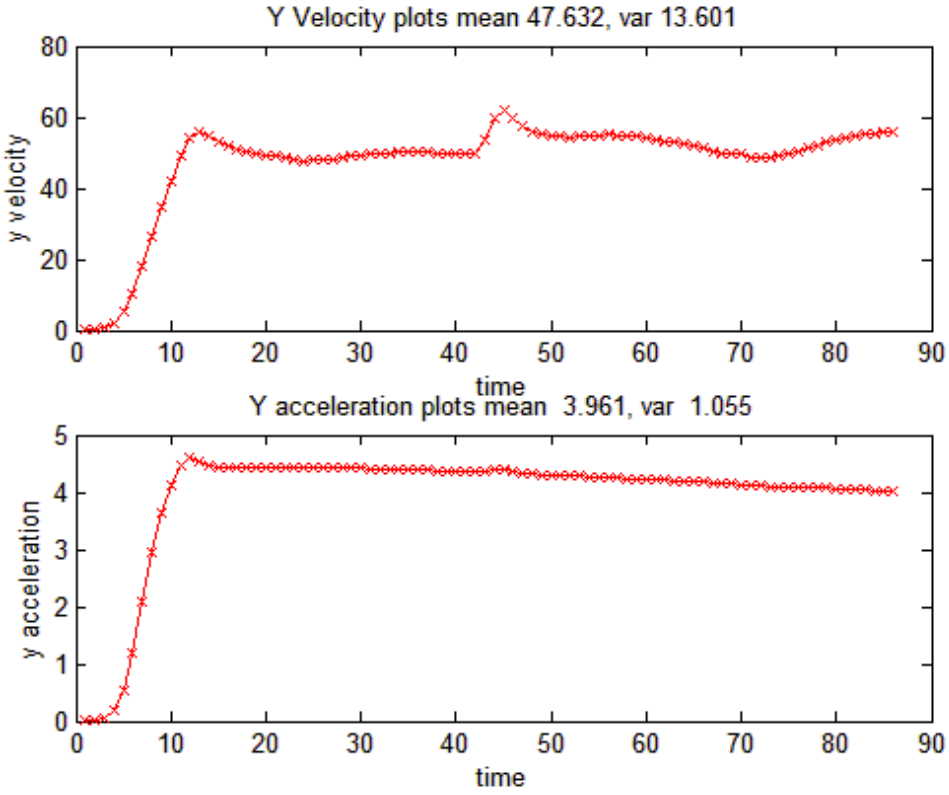


Fig. 4.5 Velocity and acceleration plots of a straight moving target. Only the **y** component of velocity is only plotted. The velocity is expressed in cm/frame, while acceleration is in cm/frame^2.

The input to the system consists of image measurements obtained using the blob and mean shift tracker. Each target consists of individual tracker, which estimates the position, velocity and acceleration of the target in the scene coordinates. The result of the switching Kalman filter is also used to obtain the mode (or) behavior of the targets in each frame. These modes are stored as runs as described in Section 4.3.2. Three different intersections, namely (i) a T-intersection (Washington and Union street near the EE/CS building at the University of Minnesota), (ii) four-intersection (with vehicles moving from a one-way street into a two-way street), I10/I169S intersection and, (iii) a four-way intersection (University and Rice Street at St. Paul) were used for the experiments. The intersections used for the experiments are shown in Fig.4.6. The experiments were conducted on recorded video sequences of about 25minutes each in length.

Fig. 4.5 shows the variation of the y velocities for a vehicle moving on a straight path without stopping and undergoing no occlusion. The velocities are components of the vehicle's speed in x and y Cartesian coordinates. As the vehicle moves in y direction, the component in the y direction is more

significant compared to the x component and hence, the x component of the velocity is omitted. The variation in the acceleration for the uniformly moving target is due to the noise in the measurements.

(a) Washington and Union Street (b) I10/I169S (c) University and Rice

Fig. 4.6. The intersections used in the experiments

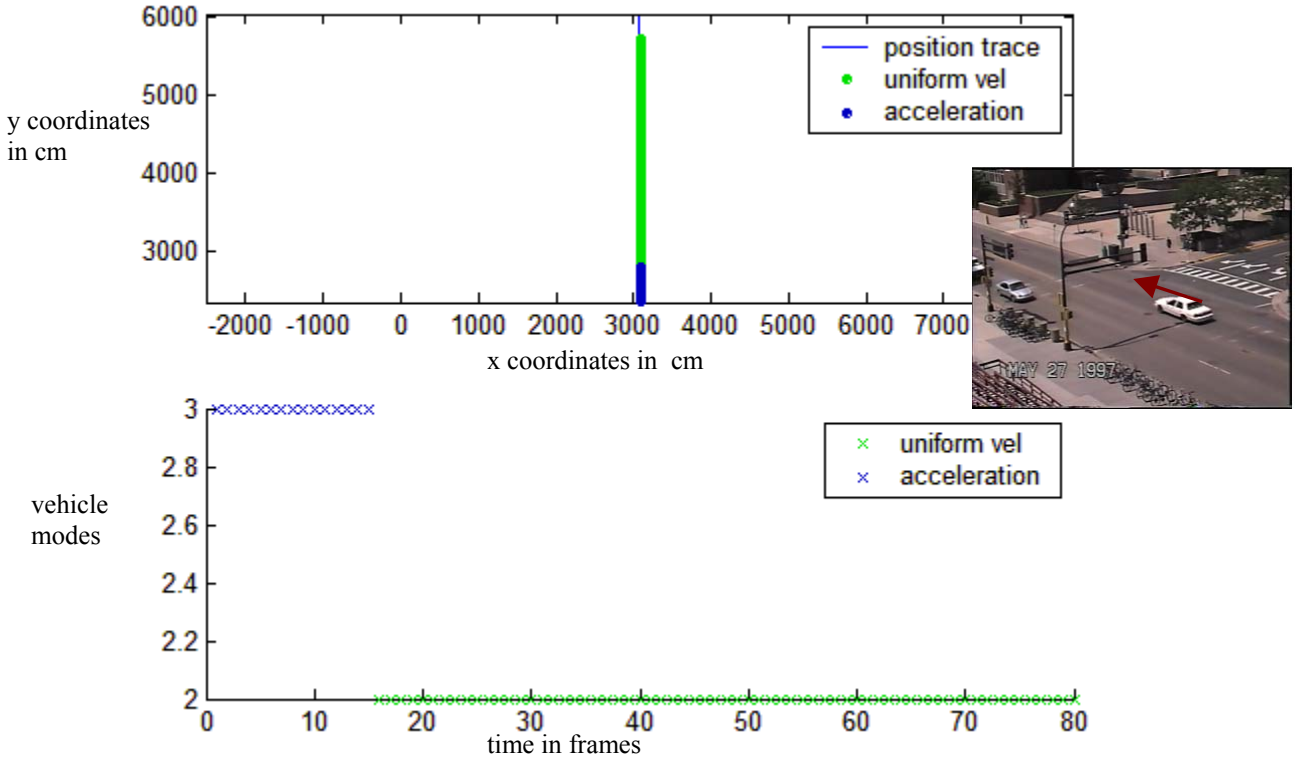


Fig. 4.7. Trajectory and categorization of a straight moving target into motion modes. The intersection used for testing is shown on the side. The x and y axes correspond to the projected x and y axes in the world coordinates. The position of the vehicle is expressed in cms.

Fig. 4.7 shows the trajectory with the recognized vehicle motion modes. The initial acceleration observed is due to inaccuracies in the initial state estimate obtained from the estimator, as the vehicle enters the scene. Fig. 4.8 shows the trajectory of a straight moving vehicle undergoing

occlusions. The initially detected right-turn and slow motion is due to the vehicle being occluded by another vehicle when it entered the scene. The vehicle is occluded further along its trajectory by other vehicles and by a background occlusion as a result of which its trajectory looks jerky as opposed to straight.

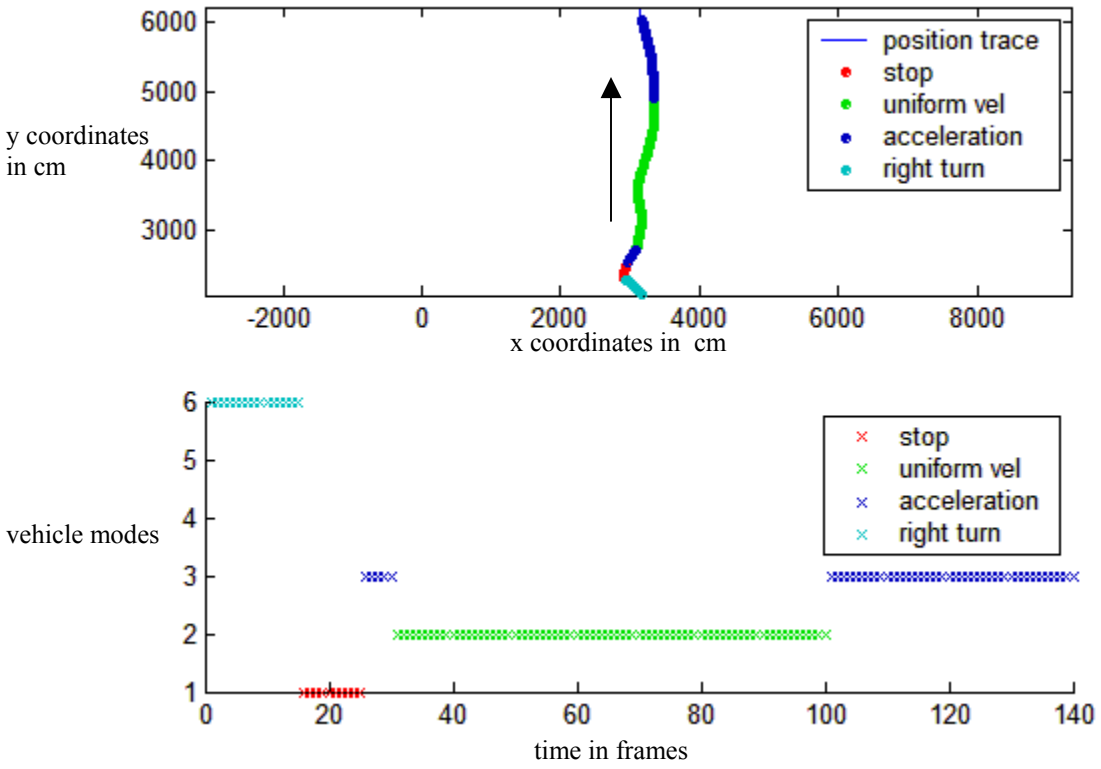


Fig. 4.8. Trajectory of an occluded vehicle with motion modes. The intersection used is the same as shown in Fig. 4.7. The vehicle's position is expressed in cm on world x, y coordinates.

The system performs fairly accurately in tracking and vehicle mode detection. The percentage of miss-tracked vehicles is 8.8% over all the intersections, and about 10.2% of the vehicles were not detected, resulting in an overall accuracy of about 81% for tracking. The accuracy of tracking is limited by the extent of occlusions in the scene as well as effects such as sudden illumination changes such as shadowing due to passing clouds, which adversely affect accurate target localization. Most of the tracking failures also occurred due to false targets (parts of background detected as foreground due to illumination changes, pedestrians or groups of pedestrians identified as vehicles). Persistent or multiple occlusions especially occurring early in the tracking (soon after target initialization) also resulted in track loss. A track loss is defined as the tracker jumping to a different target, or diverging from the correct target's trajectory.

The percentage of falsely identified vehicle modes was 2.9% for the intersection (i) & (ii), and about 21% for the intersection (iii). Note that incorrect detection of vehicle modes merely means some portion of a vehicle's recovered trajectory was bad although the vehicle itself might have been tracked successfully. This is primarily the result of occlusions during which time a vehicle's trajectory is not necessarily accurate.

Poor segmentation, as well as the sampling window size used for mode detection affects the accuracy of the mode detection. For instance, the majority of the false mode detection in the third intersection resulted from slowly turning vehicles whose mode was recognized by the system as slow moving while the left or right turn was missed completely owing to small displacements. Further, any errors in the calibration can also affect the mode detection due to incorrect position estimates in the scene coordinates.

While the mean shift tracking algorithm helps to localize stopped targets, even when they are invisible to the blob tracking algorithm (due to the stopped target being modeled as part of the background), the algorithm performs poorly when the target's color distribution is not very distinct from the background, resulting in track loss in some cases. Also, multiple stopped vehicles with more or less similar color result in poor target localization due to the mean shift tracker. In the intersection (ii), shown in Fig. 4.6 (b), only the vehicles that stopped alone or spaced at significant distance from each other were tracked even while stopping and picked up correctly once they started moving from the stopped position. All other vehicles were miss-tracked or the trajectory was lost.

Also, for targets farther away from the camera, the lower resolution of the target, introduces significant errors in tracking. One way to reduce the problem of reduced resolution would be to use multiple cameras in the scene. While this adds to the computational requirements of the system, more accurate and reliable tracking performance can be obtained by the using multiple cameras due to better resolution and easier occlusion resolution.

Another means of improvement would be to make use of more additional cues to prevent track drift especially when the vehicles are stopped. For example, features such as edges and corners which are visible for most vehicles would be very good cues for pinpointing the target location in case of failure of the color-based mean shift tracking algorithm and blob tracking.

Fig. 4.9, and Fig. 4.10 show trajectory with mode categorization for a lane changing, left and right turning vehicles. Fig. 4.11, shows the trajectory of a right turning vehicle that made a slow turn due to waiting for crossing pedestrians. This is shown by the stopped events detected for the vehicle. Fig. 4.12a shows snapshots of the tracking sequence in the intersection Fig. 4.5c, and the Fig. 4.12b shows the trajectory of a vehicle numbered 1 shown in Fig. 4.12a with the detected modes.

Most of the misclassifications in the detected modes resulted in the right-and left-turn motions. This was mostly for vehicles that were turning slowly or in the presence of large occlusions.

Fig. 4.13a, Fig. 4.13b, Fig. 4.14a and Fig. 4.14b show the mean vehicle x, y velocities, and mean x, y accelerations for vehicles in the two test intersections a, b shown in Fig. 4.6 for straight moving vehicles. Fig. 4.15a, Fig. 4.15b, Fig. 4.16a, and Fig. 4.16b show the mean vehicle x, y velocities, and mean x, y accelerations for left turning vehicles in the two intersections. Fig. 4.17a, Fig. 4.17b, Fig. 4.18a, and Fig. 4.18b show the mean x, y, velocities, and mean x, y accelerations for right turning vehicles in the two intersections.

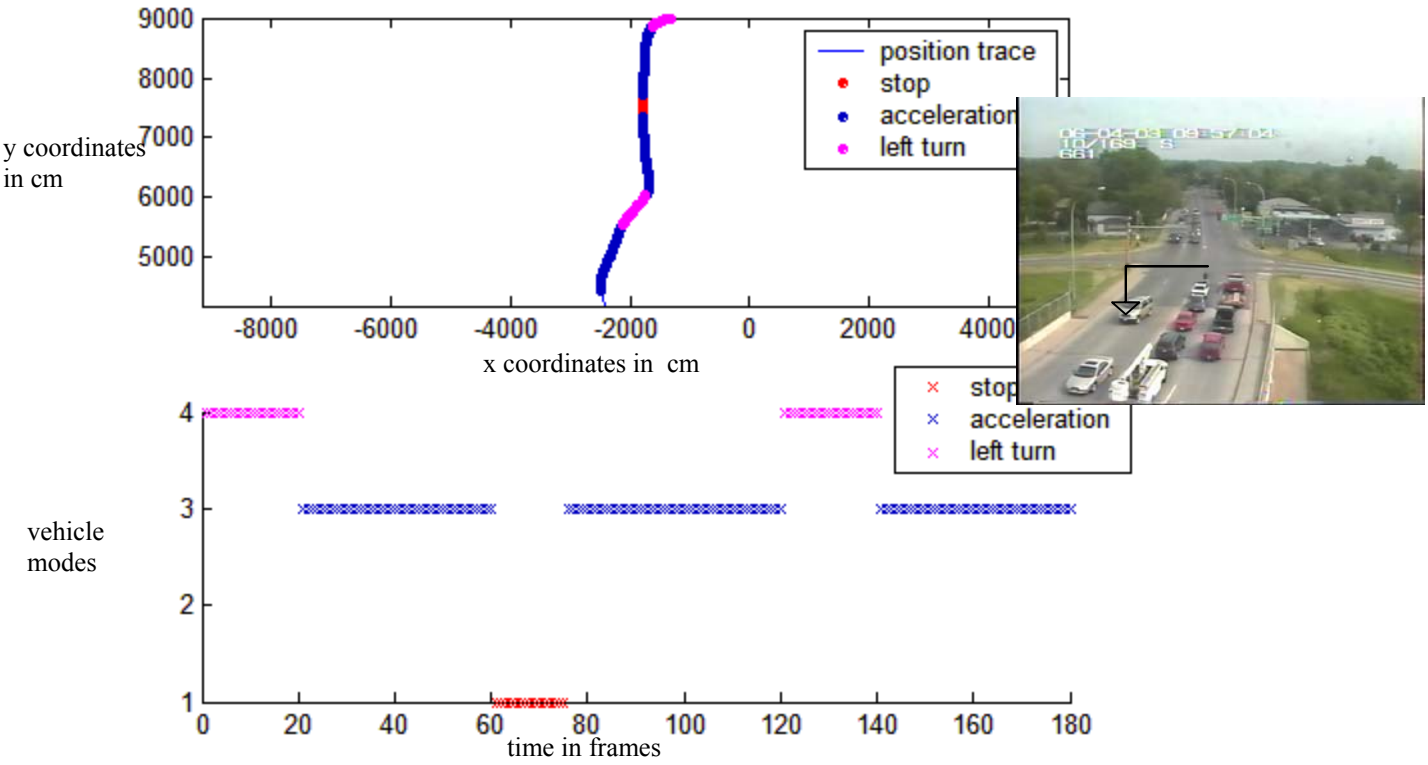


Fig. 4.9. Trajectory of a lane-changing vehicle. The lane change is depicted as a short left turn run. The vehicle and its turn path are shown by the arrow in the intersection image. Only part of the left turn was detected as the vehicle was picked up as a potential target later due to occlusions with other vehicles as the vehicle started its turn. The vehicle's position is expressed in cm along the x, y world coordinates.

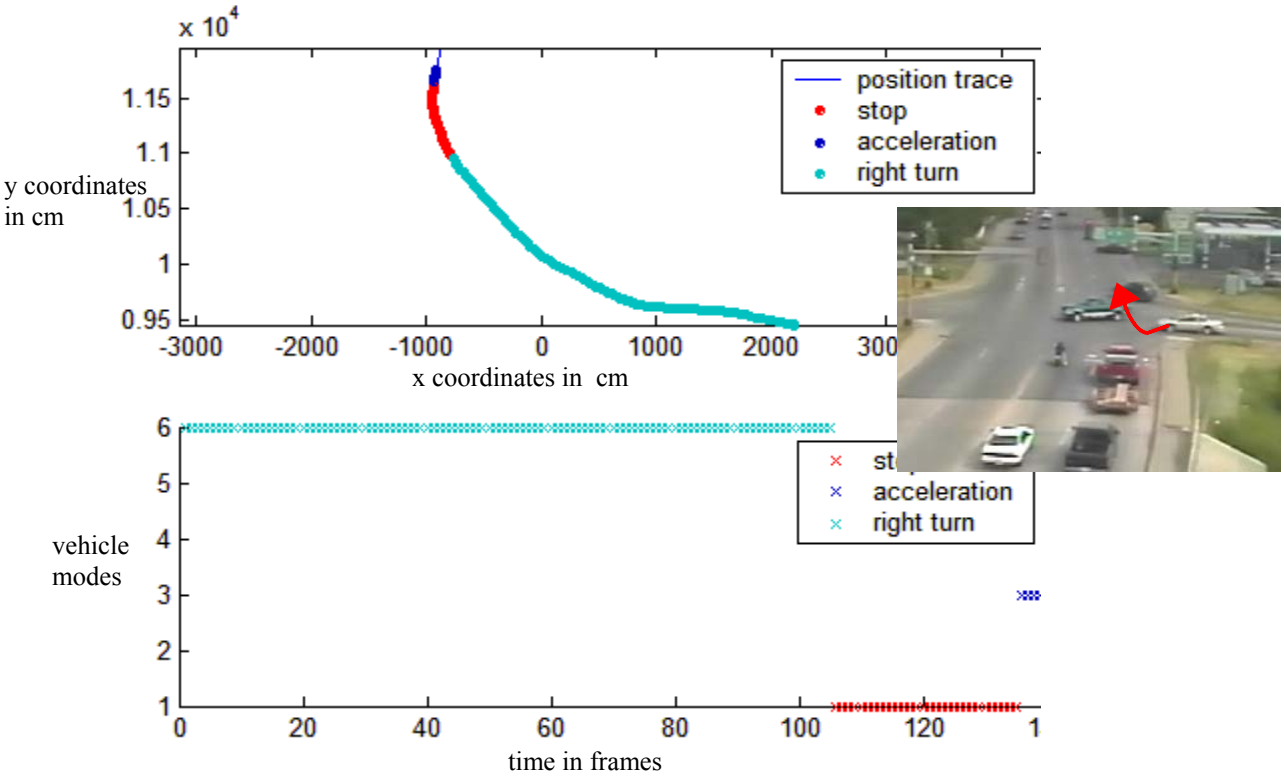


Fig. 4.10. Trajectory of a right-turning vehicle with modes. The target trajectory is plotted in scene coordinates (cm). The motion mode in the lower figure is plotted against time. The modes are 1 – slow/stopped, 2 – uniform velocity, 3 – accelerated motion, 4 – left turn, 5 – accelerated left, 6 – right turn, 7 – accelerated right

Turn Direction & Intersection	Mean wait Time frames	Standard Deviation
Left Intersection 1	21.6	25.14
Left Intersection 2	600	29.67
Right Intersection 1	32.87	23.94
Right Intersection 2	606.5	12.06

TABLE 4.1 Mean wait times for right and left turning vehicles in intersection I and II.

The higher wait time in intersection 2 is owing to the signalized nature of the turns. Further, the vehicles are visible to the tracker only very close to the intersection in the case of intersection 1 as a result of which very few vehicles are even detected before moving.

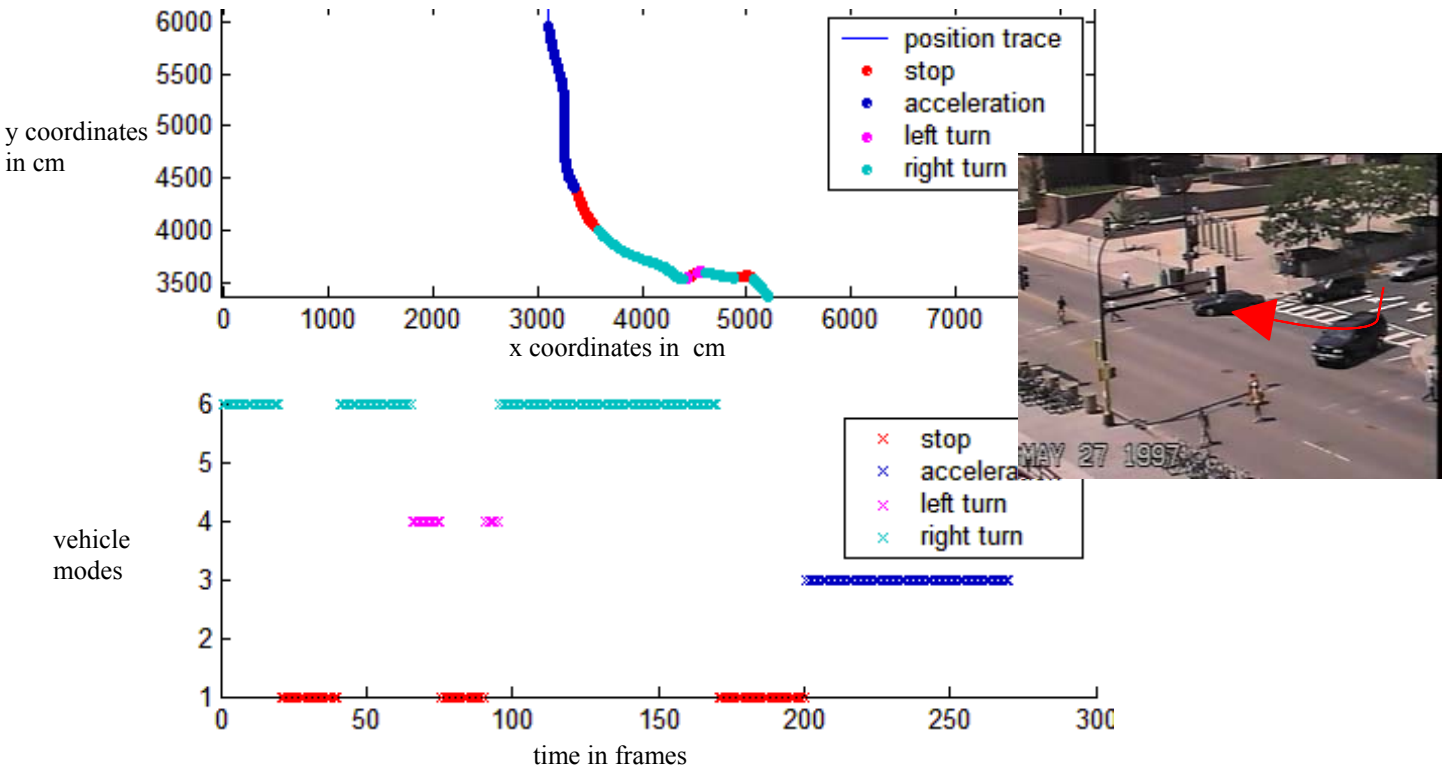
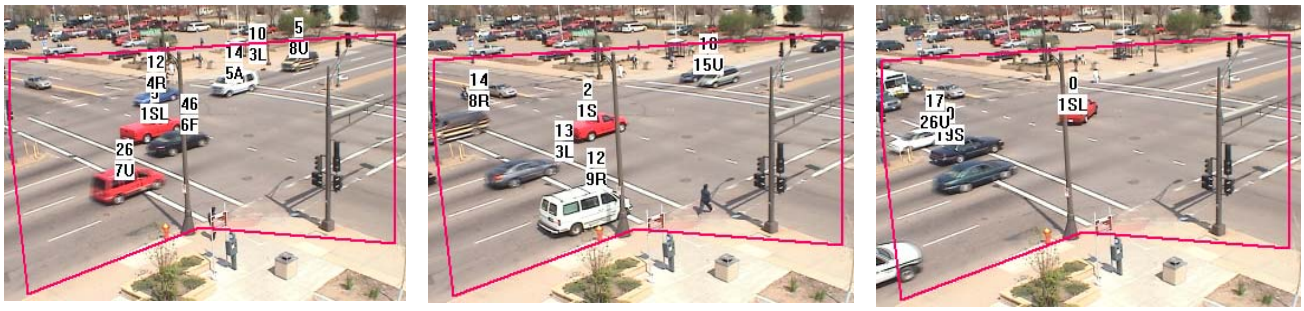


Fig. 4.11. Trajectory of right-turning vehicle stopping for pedestrians. The vehicle is shown marked with a red arrow in the intersection image on the side. The vehicle's position is expressed in cm in x and y world coordinates.



Frame 183 Frame 270 Frame 668

Fig. 4.12 (a). Tracking sequence. The target number is indicated in the bottom with the event, and the top number corresponds to the speed of the vehicle in mph at the given frame. As shown, target numbered 1 is tracked successfully despite the occlusions. The events strings correspond as follows: S – slow/stopped, U-uniform motion, F-overspeeding, R-right turning, A – acceleration, SL-slow left.

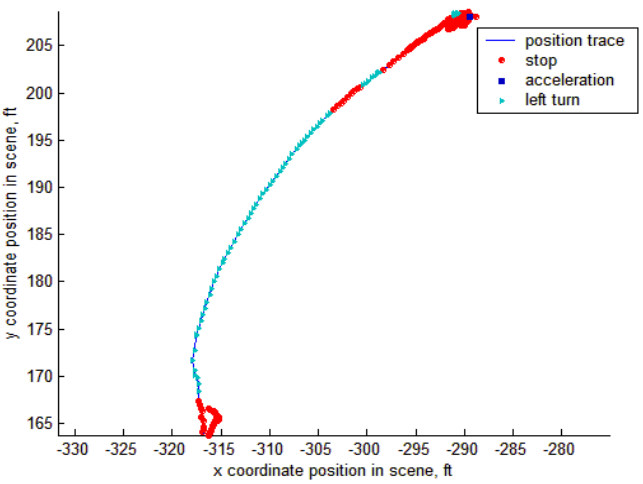
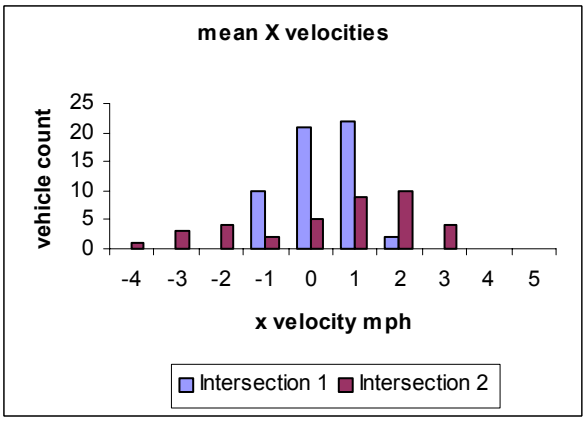
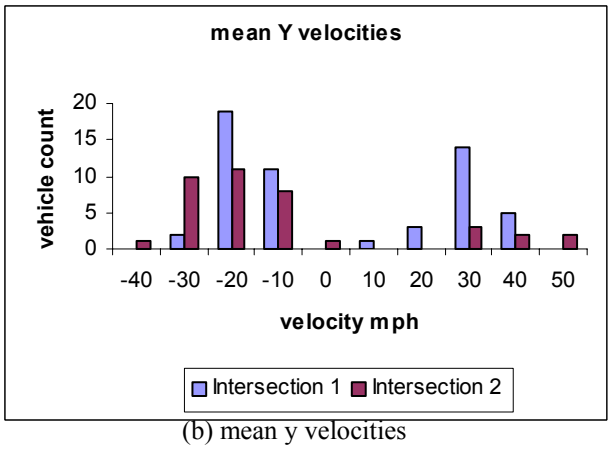


Fig. 4.12(b) Trajectory of a target. Trajectory of the target numbered 1 shown in Fig. 4.12(a) with the detected modes.

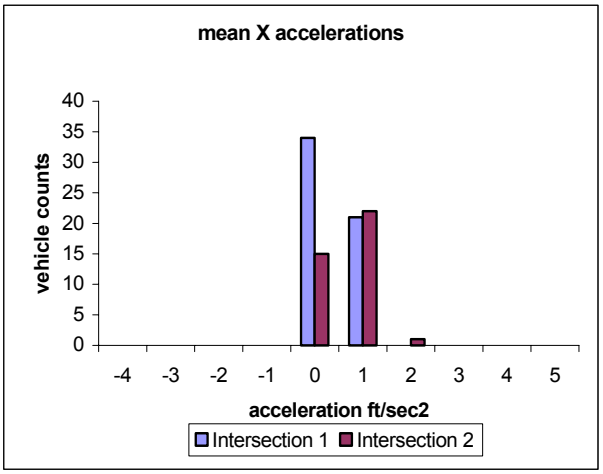


(a) mean x velocities

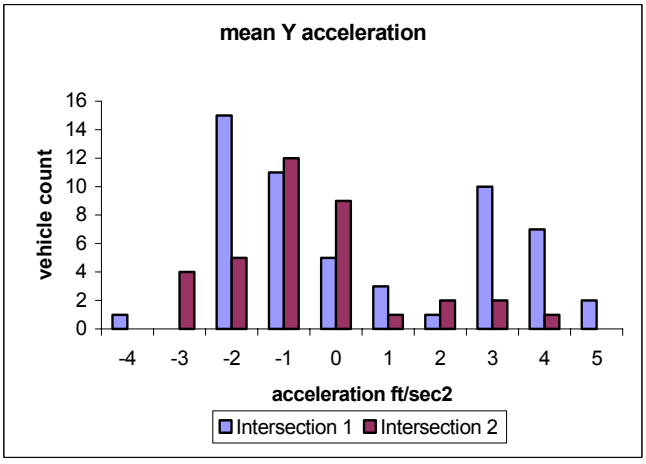


(b) mean y velocities

Fig. 4.13. Mean x and y velocities of straight-moving vehicles in intersection I and II. The velocities are expressed in mph.

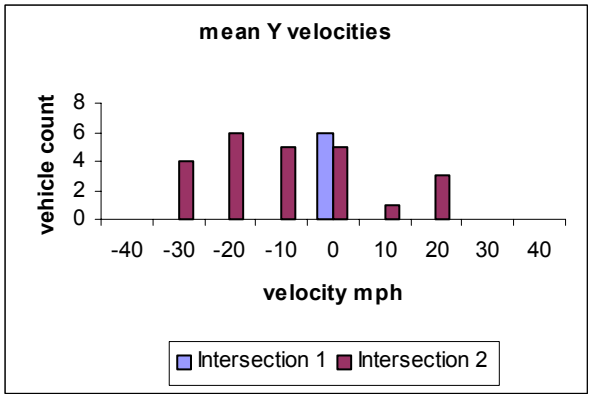


(a) x accelerations

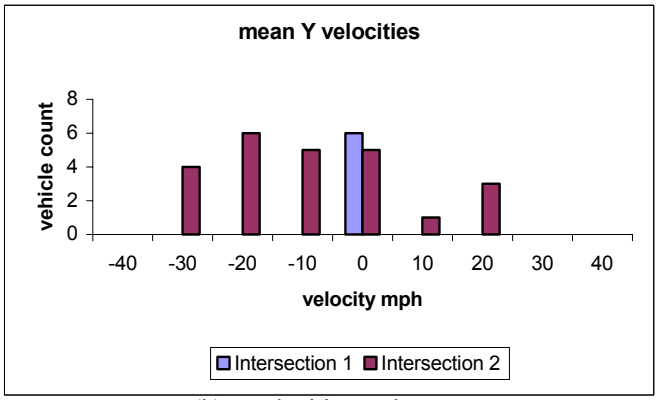


(b) y accelerations

Fig. 4.14. Mean x and y accelerations of straight-moving vehicles in ft/sec^2 in intersection I and II.

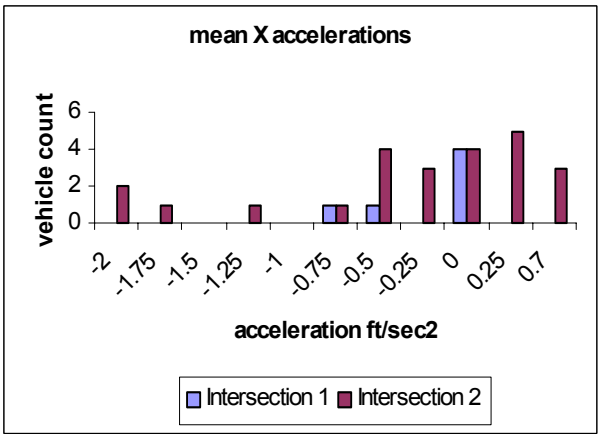


(a) x velocities mph

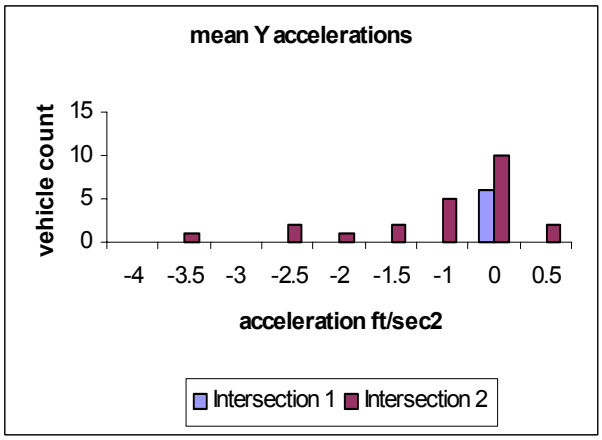


(b) y velocities mph

Fig. 4.15. Mean x and y velocities of left-turning vehicles in intersection I and II. Velocities are expressed in mph.

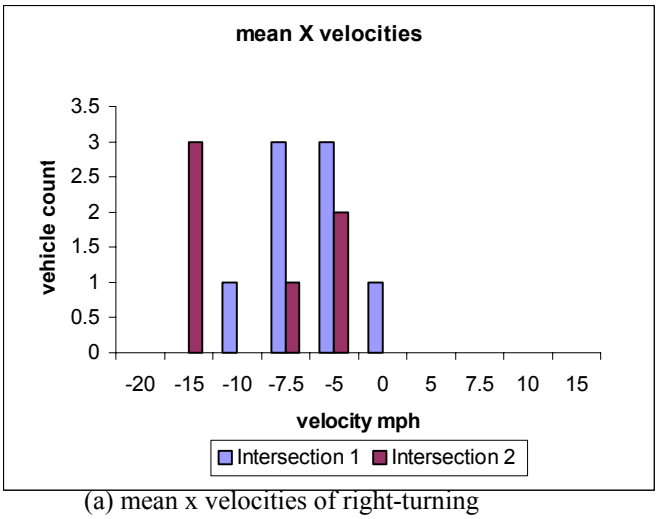


(a) x accelerations mph

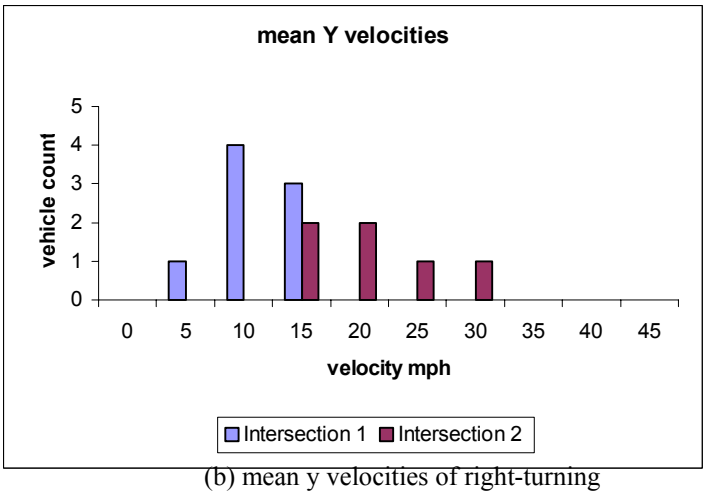


(a) y accelerations mph

Fig. 4.16. Mean x and y accelerations of left-turning vehicle in intersections I and II. Accelerations are expressed in ft/sec^2.

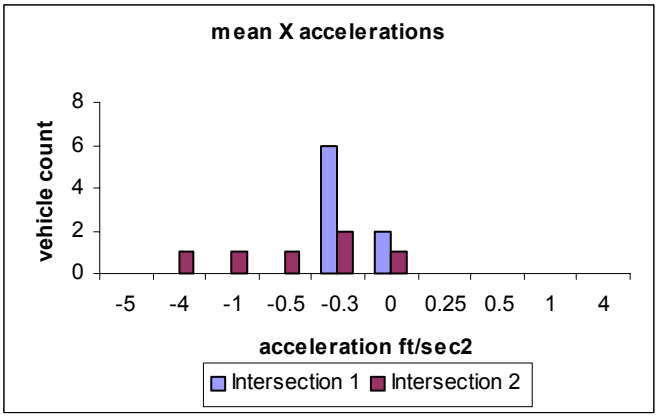


(a) mean x velocities of right-turning vehicles in mph

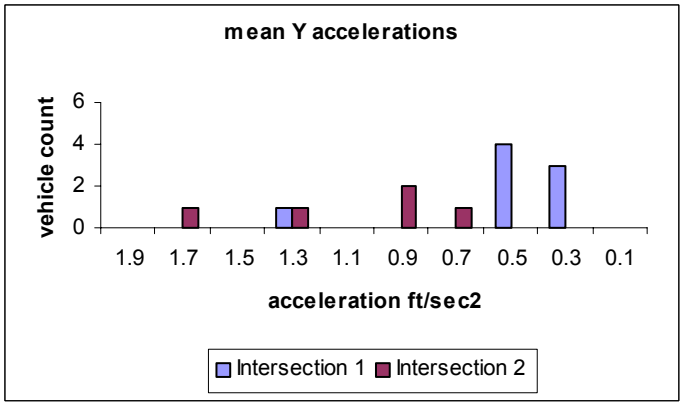


(b) mean y velocities of right-turning vehicles in mph

Fig. 4.17. Mean x and y velocities of right-turning vehicles in intersections I and II. Velocities are expressed in mph.



(a) mean x accelerations in mph



(b) mean y accelerations in mph

Fig. 4.18. Mean x and y accelerations of right-turning vehicles in intersections I and II. Accelerations are expressed in ft/sec^2.

Motion Direction	Actual Counts	Detected Counts
South to North	65	32
North to South	69	63
West to East	87	75
East to West	72	35
East to South	8	5
South to West	102	102
North to West	15	6

Table 4.2 Vehicle classification counts for a 20 min video segment. The actual counts correspond to the manual count of vehicles in each direction, while the detected counts correspond to the result of trajectory classification.

5 CONCLUSIONS

5.1 Conclusions

This report presented a data collection system for outdoor traffic intersections using a single vision-based camera system. We proposed an algorithm for obtaining good spatial resolution and minimizing occlusions through an optimization-based camera-placement algorithm. A camera calibration algorithm along with the camera-calibration-guided user interface tool was also presented. Finally, we presented a computationally simple data collection system using a multiple cue-based tracker. Extensive experimental analysis of the system was performed using three different outdoor traffic intersections.

5.2 Summary of System Capabilities

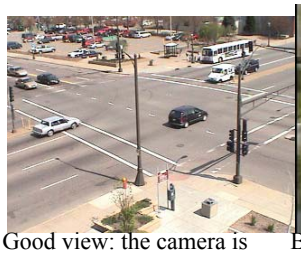
Camera Placement  Good view: the camera is very close to the intersection and the view angle is about 45°.  Bad view: the camera is very distant from the intersection which helps to capture a large portion of the scene, but also reduces the resolution of targets, thereby, making tracking very difficult.	The accuracy of tracking depends on how well the targets can be detected in the image sequences. This depends significantly on the placement of the camera in the scene. Long distance views, oblique angles (less than 30º) provide very limited accuracy in detections. Good and bad views are illustrated by the figures on the left.
Camera motion	The system is robust to most camera motion and jerks resulting from wind, as long as the camera motion is within 30 pixels.
Illumination  Shadow cast by clouds, distant view of the scene, and bad video quality, make tracking very difficult.	The system is designed to work in daylight conditions. Hence, the better the illumination, the better will be the system's performance in discriminating the targets. Good tracking can also be obtained in cloudy conditions as long as good quality video can be obtained. Poor formats include, videos recorded on VHS tapes, as well as, highly compressed .DivX formats. The system is robust to most fluctuations in illuminations, although tracking might disrupted for short periods until the systems adapts to the new illumination. In most cases, new targets are not detected since the background model is incorrect. Shadows affect tracking adversely due to occlusions between the targets. Another case of tracking failure occurs in the case of darkening of the surrounding background of a dark target (dark vehicle passing on a shadowed region).
Congestion  For oblique view, vehicles were segmented as a single blob due to large occlusions, which limits the accuracy of the tracker.	The extent of congestion the system can handle depends on the view of the scene. The more oblique the view, the less congestion the system can handle. This means that the accuracy with which individual targets can be tracked decreases with congestion. A congested scene is depicted on the left.

TABLE 5.1 Capabilities and limitations of the software.

5.3 Software Functionalities

Function	Input	Output
Calibration	Image of scene to be calibrated. The system currently handles images in .png format. The user provides measurements in the form of distance between landmarks (lane-marking–to-lane marking, ground measurements between identifiable landmarks), parallel lines. Perpendiculars and horizontal lines as shown in Fig.3.7 (a-d). The user may also provide the name of the video file to be processed. The accepted format is .avi .	The system computes the calibration matrices for converting the image measurements to the scene and vice versa and stores the information as an .fml file. These transformations are computed using the ground truth measurements and scene structure (parallel lanes, vertical structures, horizontal markers, etc) provided by the user.
Region of Interest	Input image in .png format and a set of points specifying a closed polygon. Currently, the user can specify one region of interest to be monitored in the image sequence. This is specified using a point click interface as shown in Fig. 3.6.	The result is stored as a polygon in the same .fml file as the calibration.
Video Analysis	This is instantiated using Analyze video in the interface as shown in Fig. 3.5. The user can select between two different algorithms: trajectory data collector (based on shape estimation), or switch filter data collector.	Summary statistics: number of left-turning, right-turning, stopped, over-speeding, and straight-moving vehicles, and average speeds at different time intervals (10 sec, 20 sec, 30 sec, 5 min, 30 min, 60 min), trajectory of vehicles. The summary statistics besides the trajectory are stored in a .txt file.

TABLE 5.2 Summary of the data collection software.

6 REFERENCES

[1] A. Bartoli, R.I. Hartley, and F. Kahl, “Motion from 3D line correspondences: linear and non-linear solutions,” in Proc. CVPR’03, pp. 477–484, June 2003.

[2] A. Bartoli and P. Sturm, “Constrained structure and motion from multiple uncalibrated views of a piecewise planar scene,” IJCV – International Journal of Computer Vision, vol. 52, no. 1, pp. 45–64, April 2003.

[3] P.L. Bazin, “A parametric scene reduction algorithm from geometric relations,” in Proc. Vision Geometry IX, SPIE's 45th annual meeting, 2000.

[4] D. Bondyfalat, B. Mourrain, and T. Papadopoulos, “An application of automatic theorem proving in computer vision,” In Proc. Automated Deduction in Geometry, pp. 207–231, 1998.

[5] Jean-Yves Bouguet, “Camera calibration toolbox for Matlab,” http://www.vision.caltech.edu/bouguetj/calib_doc/index.html.

[6] R. Cipolla and D.P. Robertson, “3D models of architectural scenes from uncalibrated images and vanishing points,” In Proc. IAPR 10th International Conference on Image Analysis and Processing, Venice, pp. 824–829, September 1999.

[7] R. Collins and R. Weiss, “Vanishing point calculation as a statistical inference on the unit sphere,” in Proc. International Conference on Computer Vision (ICCV’90), pp. 400–403, Osaka, Japan, December 1990.

[8] A. Criminisi, I. Reid, A. Zisserman, "Single view metrology", IJCV – International Journal of Computer Vision, vol. 40, no. 2, pp. 123–148, 2001.

[9] P.E. Debevec, C.J. Taylor, and J. Malik, “Modeling and rendering architecture from photographs,” in Proc. SIGGRAPH'96, pp. 11–12, August 1996.

[10] E. Grossmann, D. Ortin and J. Santos-Victor, “Algebraic aspects of reconstruction of structured scenes from one or more views”, in Proc. BMVC’01, pp. 633–642, 2001.

[11] P. Gurdjos, R. Payrissat, “About conditions for recovering the metric structures of perpendicular planes from the single ground plane to image homography,” in Proc. ICPR’00, pp. 1358–1361, 2000.

[12] K. Kanatani, “Statistical optimization for geometric computation: theory and practice,” Technical report, AI Lab, Dept of Computer Science, Gunma University, 1995.

[13] D. Liebowitz, A. Zisserman, “Metric rectification for perspective images of planes,” In Proc. CVPR’98, pp. 482–488, 1998.

[14] A. Ruiz, P.E. Lopez-de-Teruel, G. Garcia-Mateos, “A note on principal point estimability,” in Proc. ICPR’02, pp. 304–307, 2002.

[15] M. Spetsakis and J. Aloimonos, “Structure from motion using line correspondences,” IJCV – International Journal of Computer Vision, vol. 4, pp. 171–183, 1990.

[16] C.J. Taylor and D.J. Kriegman, “Structure and motion from line segments in multiple images,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17, no.11, pp. 1021–1032, November 1995.

[17] M. Wilczkowiak, G. Trombettoni, C. Jermann, P. Sturm, and E. Boyer, “Scene modeling based on constraint system decomposition techniques,” in Proc. 9th International Conference on Computer Vision (ICCV’03), pp. 1004–1010, October 2003.

[18] A.D. Worrall, G.D. Sullivan, and K.D. Baker, "A simple, intuitive camera calibration tool for natural images," in Proc. 5th British Machine Vision Conference, pp. 781–790, 1994.

[19] Z. Zhang, “Estimating motion and structure from correspondences of line segments between two perspective images,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 17, no. 12, pp. 1129–1139, June 1994.

[20] Y. Bar-Shalom and T.E. Fortmann. *Tracking and Data Association*. Academic Press, 1987.

[21] Y. Bar-Shalom, X. Rongli, and T.Kirubarajan. *Estimation with applications to tracking and navigation.* John-Wiley and Sons, 2001.

[22] B. Coifman, D. Beymer, P. McLauchlan, and J. Malik. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research*: Part C, 6(4): 271-288. 1998.

[23] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, volume 25, 2003.

[24] R. Cucchiara, P. Mello, and M. Piccardi. Image analysis and rule-based reasoning for a traffic monitoring system. In *IEEE Transactions on Intelligent Transportation Systems,* volume 119-130, 2000.

[25] D. Koller, K. Dandillis, and H.H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, 10(3): 257-281, June 1993.

[26] B. Heisele, U. Kressel, and W. Ritter. Tracking non-rigid, moving objects based on color cluster flow. In *Proc. Computer Vision and Pattern Recognition Conf.*, pages 257-260, 1997.

[27] S. Khan and M. Shah. Object based segmentation of video using color, motion and spatial information. In *Proc. Computer Vision and Pattern Recognition Conf.*, volume 2. pages 746-751. December 2001.

[28] P. Perez, J. Vermaak, and A. Blake. Data fusion for visual tracking with particles. *Proceedings of the IEEE*, 92(3): 495-513, February 2004.

[29] C. Stauffer and W.E.L. Grimson Adaptive background mixture models for real-time tracking. In *Proc. Computer Vision and Pattern Recognition Conf.*, June 1999.

[30] H. Veeraraghavan, O. Masoud, and N.P. Papanikolopoulos. Computer vision algorithms for intersection monitoring. *IEEE Trans. on Intelligent Transportation Systems,* 4(2):78-89, June 2003.

[31] H. Veeraraghavan, and N.P. Papanikolopoulos. Combining multiple tracking modalities for vehicle tracking in traffic intersections. In *IEEE Conf. on Robotics and Automation,* 2004.

[32] K.P. Murphy. Switching Kalman filters. Technical report, U.C. Berkeley, 1998.

[33] Z. Ghahramani and G.E. Hinton. Switching state-space models. *Neural Computation*, 12(4): 831-864, April 2000.

[34] R. A. Jacobs. What determines visual cue reliability? *Trends in Cognitive Sciences*, 6(8): 345-350, 2002.

[35] J. Triesch and C. von der Malsburg. Democratic integration: Self organized integration of adaptive cues. *Neural Computation*, 13(9): 2049-2074, 2001.

[36] L. Zelnik-Manor, P. Perona. Self-tuning spectral clustering. *Advances in Neural Information Processing Systems*, 17: 1601-1608, 2005.