A Framework for Dynamically Measuring Mean Vehicle Speed Using Un-Calibrated Cameras

Suree Pumrin

Intelligent Transportation Systems Laboratory
Department of Electrical Engineering
University of Washington, Seattle, WA 98195

December 12, 2001
Abstract

We present an algorithm to estimate mean vehicle speed from roadside cameras operated by a traffic management agency. These roadside cameras are neither calibrated nor are there calibration marks available in the camera views. However, estimating camera calibration coefficients is the most important step to extracting quantitative information about the 3D world from a 2D image. It is in this framework that we present an algorithm that: (1) performs a simplified dynamic calibration and (2) estimates mean vehicle speed.

Many algorithms depend on point correspondences between the earth coordinates and the image coordinates as well as targets of known shape to obtain accurate results. However, in the work presented, we desire to estimate the mean of a distribution of vehicle speeds and will demonstrate that a simplified form of calibration is adequate for making an accurate mean speed estimate.

We perform dynamic camera calibration using training sets of 10-second video sequences. Our proposed method detects moving vehicles in a set of consecutive frames. This information, together with mean vehicle dimension estimates, is used to create scaling factors that are then used to transform between motion in the image and motion in the earth coordinate system. Our proposed algorithm has a camera model with a reduced number of camera calibration parameters. We validate our algorithm with simulated data and real-world traffic scenes.
Contents

1 Overview ................................................................. 5
  1.1 Introduction .................................................... 5
  1.2 Overview of the proposed system .............................. 6

2 Lane Mask Detection and Moving Vehicle Tracking ............ 9
  2.1 Pre-processing ................................................. 10
  2.2 Lane mask region detection .................................. 10
  2.3 Moving vehicle tracking ....................................... 17

3 Camera Calibration and Mean Speed Estimation .................. 23
  3.1 Affine camera calibration model .............................. 23
  3.2 Scale modeling ................................................ 26
  3.3 Linear scaling model estimations ............................ 27
  3.4 Travel distance approximation and mean speed estimation 28

4 Simulation and Empirical Results ................................ 29

5 Conclusion and Future Work ....................................... 33
  5.1 Conclusion ..................................................... 33
  5.2 Future work ................................................... 33
List of Figures

1.1 The proposed methodology for dynamically measuring mean vehicle speed. 8

2.1 Actual traffic image. ................................................................. 9
2.2 The mean vehicle speed measuring system. ................................. 10
2.3 (a) Pre-processing filters; (b) high-boost filtering mask. ............... 10
2.4 The lane mask detection. .......................................................... 11
2.5 Using the mask in (a) to compute \( \frac{\partial I}{\partial x} \), and using the mask in (b) to compute \( \frac{\partial I}{\partial y} \). ......................................................... 12
2.6 Moving edge detection. ............................................................. 12
2.7 Object selection. ................................................................. 13
2.8 Convex hull of pixels (a) shows the structure of pixels and convex polygon (b) shows the positive \( D \) for counterclockwise cycle and the negative \( D \) for a clockwise cycle. ....................................................... 15
2.9 The collection of a moving edge image is shown in the top image, the closing edge image in the middle image, and the binary lane mask in the bottom image. ......................................................... 16
2.10 The binary lane masks \( M_1, M_2, M_3, \) and \( M_4 \) from top to bottom, respectively. ......................................................... 18
2.11 The moving vehicle tracking procedure. ..................................... 19
2.12 Spatio-Temporal segmentation. .................................................. 19
2.13 Typical image sequence (left). Spatial edge image in the right top image, and the Temporal edge in the middle right image. Moving edge image for the middle image on the left created by the Spatio-Temporal segmentation module in the bottom right image ........................................... 20
2.14 Moving vehicle selection. ....................................................... 20
2.15 Moving blobs tracking. .............................................................. 22

3.1 Camera geometry. .................................................................. 24
3.2 A basis of unit vectors in the earth coordinate system. .................. 25

4.1 Simulation cuboids (left). The cuboid binary blobs (middle). Tracking line for moving cuboids (right middle image). ......................... 30
4.2 Horizontal scaling estimation of the simulation data(left). Vertical scaling estimation of the simulation data(right). ................................. 30
4.3 Histogram of speed error of the simulation speed and the reference speed over the standard of deviation. ......................................................... 31
4.4 Typical image sequence (left). The binary blobs (middle). Line tracking of the blobs’ centroids sequence (right image). ................................. 31
4.5 Horizontal scaling estimation of the actual traffic data (left). Vertical scaling estimation of the actual traffic data (right). .......................... 32
4.6 Histogram of speed error of the actual traffic speed and the inductance loop speed. ................................................................. 32
List of Tables

2.1  The sample correlation coefficient for each pair of binary lane masks. . . . 17
Chapter 1

Overview

1.1 Introduction

Traffic monitoring is an important work for highway traffic management. It provides a way to collect information to regulate traffic, to manage traffic and to conduct urban planning. Typically, the inductive loop detector has been used to record the passage of vehicles as well as some parameters such as length and speed. However, this system requires an installation within the roadway. Nonintrusive means such as video, infrared, laser, and radar can monitor the traffic without a direct impact to the roadway.

Among many technologies being investigated, a video sensor has shown strong potential in providing robust collection of traffic data and wide range of usefulness. The video sensor has many advantages over other detector technologies [1, 2, 3, 4, 5]: (1) it can cover a wide area of traffic and monitor the behavior of each individual vehicle within the range of the video sensor, (2) it reduces installation and maintenance costs, (3) the detection performance is easily verified, (4) detectors are easy to reconfigure interactively, (5) a much larger set of traffic parameters can be estimated (measurement of traffic volumes, vehicle speeds, vehicle classifications, travel times, lane changes and incident detection), and (6) it provides precise vehicle tracking and classification.

There have been few efforts to measure traffic parameters using video images from un-calibrated cameras. Most algorithms either use reference information in the scene or create such references interactively. For example, Dickinson and Waterfall [6] introduce a video image processing system for monitoring road traffic. They apply it to measure the number of vehicles, the vehicle speed, and the length of vehicles as the vehicles pass a point on roadway. Ashworth et al. [7] use background frame differencing, double thresholding, modified difference image, and an adaptive threshold to overcome the shadow effect. Ali et. al. [1] proposes methods to analyze road traffic automatically with a known camera height and angle of view on a stable camera. The detection, tracking, and classification algorithms are developed using computer vision and image processing techniques. The detection algorithm includes mathematical morphology and frame differencing. The tracking algorithm uses gray-level signature within a search area. The classification algorithm uses correlation of binary skeleton and medial axis transformation of the depth map. Harvey and Cohen [8] describe a feasibility method to measure vehicle speed by comparing known vehicle dimension from two consecutive half frames. There are some errors associated with a poor matching of the interlace half frame. Beyer and Malik [3] propose detecting and tracking vehicles on freeways when there is congestion. The feature-based tracking with sub-features (i.e., corner features) is used
for tracking purposes. All these methods require the operator to perform a calibration procedure before traffic parameters can be undertaken.

In this work, we present a framework to estimate speed using a sequence of video images from an un-calibrated camera. The cameras are typically not installed in a manner that they can easily be calibrated. It is assumed that we have no control over camera movements. We can not directly obtain information such as camera focus, tilt, or angle. It is assumed that the camera parameters can change with time. Given this scenario, we propose an algorithm to dynamically calibrate the cameras. Underlying assumptions and an overview of our proposed system are described in the next section.

1.2 Overview of the proposed system

Presently, the transportation agencies deploy two types of cameras: (1) cameras used only for visual surveillance and (2) expensive, calibrated cameras to measure speed. State-of-the-art camera technologies used by transportation agencies to measure speed require several things: (1) an expensive camera mounted in the roadway right-of-way, (2) a detailed calibration of the camera, and (3) that the camera remains stationary. The algorithm presented here creates a new virtual speed sensor that leverages the large numbers of low quality cameras already installed by the transportation agencies.

The overall goal is to produce a speed sensor that is similar in accuracy to existing sensors called inductance loops but at a vastly lower cost. Existing camera technologies, used in this arena, require an expensive camera to be placed by the roadway, calibrated, and never moved. Both of these techniques do not leverage the large numbers of cameras already installed by the transportation agencies. These camera pan, tilt, and zoom, but are not calibrated. Our algorithm presented here using this set of deployed cameras to create a new vision-based speed sensor.

In a typical site, there is a Charge Coupled Device (CCD) camera with a known height position and a minimum-maximum focal length. The operator can zoom, tilt, and pan this camera at any time. The camera calibration needs to be updated each time the camera pose changes.

In order to simplify our simulation, it is necessary to make some constraints and assumptions about the traffic scenario.

1. The camera motion is constrained to an up/down motion with an evaluation angle only.
2. The vehicles move from top to bottom or from bottom to top of the image plane.
3. The vehicle movement is smooth and no sudden change in direction and speed occurs between frames.
4. The vehicle dimensions are from known distributions.
5. The height of a camera is known.

The dynamic mean speed measuring system described in this proposal is summarized in the flow diagram of Figure 1.1. The system includes acquired image sequences, preprocessing, calibrating the cameras, generating lane mask, tracking moving vehicles, and estimating speed. The preprocessing improves image quality. The lane mask generation module produces a mask of the significant scene’s activity area. Two consecutive masks are compared for their similarity. If
the similarity is less than the threshold, we re-calibrate the camera. On the other hand, if the similarity between the masks is greater than the threshold, three consecutive image sequences are passed through the moving vehicle tracking algorithm. The speed estimation is calculated from the resuming calibration parameters and the moving vehicle tracking outputs. The process is repeated until the end of the 10-second image sequence. Then the system acquires the next 10-second image sequence and dynamically repeats the whole process.

We modify and extend several known approaches for our system. The work developed in this proposal addresses four major research contributions: 1) the development of a lane mask generation as a decision tool to update camera calibration parameters, described in Chapter 2; 2) the logical combination of a dilation of moving edge image and intensity edge image for Spatio-Temporal segmentation, described in Chapter 2; 3) the moving vehicle tracking module, including Spatio-Temporal segmentation, moving vehicle selection, and moving blobs tracking, presented in Chapter 2; and 4) the development of vertical and horizontal scale estimation for camera calibration, described in Chapter 3. Synthetic and real traffic images of highway traffic scenes demonstrating the feasibility of the developed paradigms is described in Chapter 4.
Figure 1.1: The proposed methodology for dynamically measuring mean vehicle speed.
Chapter 2

Lane Mask Detection and Moving Vehicle Tracking

In our system, one Closed Circuit Television (CCTV) camera installed by Washington State Department of Transportation (WSDOT) is used as an image sensor for each site. The image sequence is 320 by 240 pixels in JPEG format at a sampling rate of five-frames per second. An example of the traffic image is illustrated in Figure 2.1.

![Figure 2.1: Actual traffic image.](image)

Prior to performing any measurement, image quality is improved by a pre-processing module. The output images from this module have less noise components but retain some strong edges within a smaller part of the image. Next, the extraction-of-activity region in the image is generated. An accumulation of moving vehicle edges gives an activity region. A slight difference in the consecutive activity regions indicates a decision criteria for re-calibrating the camera. Finally, the moving vehicle is segmented and tracked along the sequence of images. A sequence of centroids for the tracked vehicles are used as inputs to compute camera calibration and to measure the mean vehicle speed. An overview of our mean vehicle speed measuring system is illustrated in Figure 2.2.
2.1 Pre-processing

The images taken from the CCTV camera are usually of reasonable quality. A typical traffic scene in Figure 2.1 shows that it contains some noise components arising from things such as dust and rain as well as camera operation. To reduce the effect of trade-off noises, we pre-process the images. Wu and Reed [9] suggest methods to improve the tradeoff in video processing algorithms with respect to noise by applying processes such as low-pass/high-pass filtering, sharp masking, and boundary-constrained filtering. The pre-processing module requires computational power and time. For our process, we select simple methods that improve input images and that are worth the extra time necessary for processing. We apply a median filter to act as a smoother that will preserve the small details in the image. Later we apply a high-boost filter to further enhance high frequency components.

![Image](image.png)

Figure 2.3: (a) Pre-processing filters; (b) high-boost filtering mask.

The median filter is good for removing shot noise, which consists of strong spike-like isolated values. Each pixel \( I_m(u, v) \) in the filtered image is determined by the median value of all selected 3x3 neighbors of the input image pixel \( I_p(u, v) \). The median value of the 3x3 window equals the 5th element of the sorted array of those 9 pixels in the window. Even though the median filter removes noise, it sometimes blurs some sharp details, such as edges and corners of the image. To enhance the high frequency components and keep the effect of the median filter, we apply a high-boost filter to the median filter output image. An output at pixel \( I(u, v) \) is defined as the median filtering followed by the high-boost filtering. Our high-boost filter is defined as in Figure 2.3 (b), where \( W > 8 \), in our case we use \( W = 18 \).

2.2 Lane mask region detection

To make a decision criteria for re-calibrating the camera, the lane mask region detection algorithm accumulates the moving vehicle edges. A collection of the moving vehicle edges gives an activity region. A 10-second image sequence is accumulated to make the lane mask region. The edge information in a moving car sequence identifies the mask of this zone.

Stewart et. al. [10] introduces an automatic lane finding algorithm for an active area of the scene where motion is occurring. The lane finding algorithm extracts the lane positions and movement
Recently, Stauffer and Grimson [11] developed a visual monitoring system that passively observes moving objects in a site. The network of sensors provides inputs for the large site. A particular pixel is modeled as a mixture of Gaussians. It provides statistical descriptions of typical activity patterns and detects unusual events.

Our lane mask detection is similar to the algorithm developed by [10]. However, their lane finding algorithm shows the activity zone from one traffic angle. Its scenario has a perpendicular roadside angle for the camera. In our algorithm, the whole activity region is determined for various camera angles. Our lane mask region detection includes moving edge detection algorithm and an object selection algorithm. After finding the two consecutive mask regions, our system compares these two regions.

The proper vehicle image size for processing is at the front end of the image. To process our algorithm faster, we confine the active window selection to two-thirds of the image height from the bottom up. Morphological opening and image thresholding are used to detect the active area and identify the lanes. The next three subsections describe the details of these algorithms: (1) moving edge detection, (2) object selection, and (3) camera motion detection. Figure 2.4 shows the lane mask detection algorithm.

**Figure 2.4: The lane mask detection.**

### 2.2.1 Moving edge detection

The stationary background is removed using a backward and forward difference method [1]. We apply the Sobel gradient operator to both the forward and the backward differencing image to detect edges within various regions of an image. These gradient vectors point in the direction of the maximum rate of change between pixel and the neighborhood. Assume that we apply the Sobel edge detection to the forward differencing pixel $I_s$ at location $(u,v)$. In our case, we use a $3x3$ neighbor window for each pixel. The Sobel edge detection provides a smoothing and differencing effect [12]. The Sobel edge kernels are defined as the masks in Figure 2.5. The gradient $\nabla I_s$ is defined as
\[ \nabla I_s = \begin{pmatrix} \frac{\partial I_s}{\partial u} \\ \frac{\partial I_s}{\partial v} \end{pmatrix}. \] (2.1)

\[
\begin{array}{c|c|c}
-1 & -2 & -1 \\
0 & 0 & 0 \\
1 & 2 & 1 \\
\end{array}
\begin{array}{c|c|c}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1 \\
\end{array}
\]

Figure 2.5: Using the mask in (a) to compute \( \frac{\partial I_s}{\partial u} \), and using the mask in (b) to compute \( \frac{\partial I_s}{\partial v} \).

The magnitude of the gradient \( ||\nabla I_s|| \) gives an amount of edge information,

\[ ||\nabla I_s|| = \sqrt{\left( \frac{\partial I_s}{\partial u} \right)^2 + \left( \frac{\partial I_s}{\partial v} \right)^2}. \] (2.2)

The gradient orientation \( \alpha(\nabla I_s) \) gives the direction of the largest change,

\[ \alpha(\nabla I_s) = \tan^{-1}\left( \frac{\partial I_s}{\partial u} / \frac{\partial I_s}{\partial v} \right). \] (2.3)

In similar fashion, we apply the Sobel edge detection to the backward differencing image. The edge pixels are determined using an appropriate threshold, which depends on the content of the images. The threshold operation gives the binary edge images. Finally, we multiply these two binary edge differences from the forward and backward differences together. It gives us the final moving edge image \( ME_i(u, v) \).

Figure 2.6: Moving edge detection.
2.2.2 Object selection algorithm

The morphological operations of dilation and erosion are used. As we are not interested in extracting every vehicle, the size of a structure element determines the range of the vehicle sized and the distance between the vehicles. Some vehicles that are small in the image are removed by the erosion operations, and some vehicles are merged by the dilation operation due to their proximity or occlusion. Each detected object is labeled in order to distinguish between the different objects. Then the convex hull is used to make the contour for each object. And each bounding box object is filled in to make a closed object. Our object selection algorithm consists of: (1) morphological closing operation, (2) labeling of connected component object, (3) contouring the selected object by convex hull, and (4) filling within the region of interest. (See Figure 2.7.)

(i) Extract the connected components with morphological closing operation

Let an image $A$ and a structure element $B$ be sets in the 2-D integer space. Set $A$ contains component $(a_1, a_2)$, and set $B$ contains component $(b_1, b_2)$. We use binary image morphology (e.g., dilation, erosion, and closing) as defined in [12, 13].

Dilation of $A$ by $B$, denoted $A \oplus B$, is defined as a translation of copies of $A$ by movement vectors defined by each of the pixels in $B$. In other words, we union all of the copies together to get $A \oplus B$,

$$A \oplus B = \bigcup_{b \in B} (A)_b. \quad (2.4)$$

Erosion of $A$ by $B$, denoted $A \ominus B$, is defined as a translation of copies of $A$ in the opposite direction (-b) and an intersection of the copies together,

$$A \ominus B = \bigcap_{b \in B} (A)_{-b}. \quad (2.5)$$

A dilation followed by an erosion using the same kernel makes a closing operation, denoted $A \bullet B$,

$$A \bullet B = (A \oplus B) \ominus B. \quad (2.6)$$

The output $CL$ is obtained by applying the closing operation with a 7x7 kernel structure element $B$ to the edge image $ME$,

$$CL = ME \bullet B. \quad (2.7)$$
(ii) Labeling of connected component object

The morphological operators fill holes and separate each moving blob. To individually manipulate each blob, we assign each blob a unique label. In a scanning process, we examine connectivity at any point $CL(u, v)$ if its intensity is ‘1.’ The scanning process scans one row at a time from top to bottom. The 4-neighboring pixels of $CL(u, v)$ (i.e., the left, the above, and the two diagonally above $CL(u, v)$) are already scanned. The row-by-row labeling algorithm [13] is conducted to label separate regions. It makes two passes over the image. The first pass records the equivalences and assigns temporary labels. We assign a label to $CL(u, v)$ when the label operator meets either one of these cases:

1. All its neighbors are ‘0.’
2. Only one of its neighbors is labeled.
3. Two or more neighbors are ‘1.’

For the last case, we assign one of the labels to $CL(u, v)$ and save the rest as the equivalences. After completing the scan, the algorithm makes a second pass and replaces the equivalences with a unique label.

(iii) Convex hull

The convex hull function is used to bound each labeled object. The convex hull of a set of points is defined as the smallest convex set that contains all the points. We adopt the same method as in [14] to extract the convex hull set. The algorithm starts to look for the leftmost and rightmost pixel of each scan line. The 2-D data are already sorted by u and v coordinates. Let $p_i = (u_i, v_i)$, for $i = 1, 2, 3$. This method determines the convex hull point, $p_2$, by the preceding points $p_1$ and the selected leftmost or rightmost $p_3$.

Let $D$ be the determinant of points’ coordinates $p_1, p_2, p_3$. $D$ determines the orientation of the $p_1, p_2, p_3$ [14, 15].

$$D = \begin{vmatrix} u_1 & v_1 & 1 \\ u_2 & v_2 & 1 \\ u_3 & v_3 & 1 \end{vmatrix}. \quad (2.8)$$

[15] describes how the determinant $D$ gives twice the signed area of the triangle $p_1p_2p_3$, where the positive sign determines that the angle $p_1p_2p_3$ forms a counterclockwise (left turn) cycle. Whereas, the negative sign determines that the angle $p_1p_2p_3$ is a clockwise (right turn) cycle. (See Figure 2.8.)

Therefore, the algorithm selects $p_2$ as a vertex of the convex hull if

$$D \begin{cases} 
\geq 0 & \text{for } p_3 \text{ on the left side} \\
\leq 0 & \text{for } p_3 \text{ on the right side} 
\end{cases} \quad (2.9)$$
(iv) **Filling within the region of interest**

The vertices \( u_i \) and \( v_i \) from the convex hull output determine a region of interest (ROI) for each bounding blob. Each ROI is filled with ‘1.’ Then each ROI becomes a binary mask image within the same size of the convex polygon.

To obtain a binary mask \( M \) for each bounding box \( CO(u, v) \), the filling operator performs these steps:

1. Scan each line \( u_i \).
2. Fill all \( u_i \) within the same \( v_i \) with ‘1.’
3. Result in binary mask region.

The bottom image of Figure 2.9 shows the activity region (lane mask) after the completed lane mask region detection.

### 2.2.3 Camera motion detection algorithm

We begin with a collection of 10-second moving edge images using our moving edge detection algorithm. Lane masks are generated using our object selection algorithm. Usually, there is more than one activity zone. However, we select the biggest area as the binary lane mask. Let \( M_k(i, j) \) be a binary mask of the lane mask \( k \). The similarity of the two lane mask regions \( M_1, M_2 \) can be estimated using the sample correlation coefficient [16]. A new camera calibration is required if the sample correlation coefficient \( r_{M_1M_2} \) is less than 0.8,

\[
r_{M_1M_2} = \frac{\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} (M_1(i, j) - \bar{M}_1)(M_2(i, j) - \bar{M}_2)}{\left[ \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} (M_1(i, j) - \bar{M}_1)^2 \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} (M_2(i, j) - \bar{M}_2)^2 \right]^{1/2}},
\]  

(2.10)
Figure 2.9: The collection of a moving edge image is shown in the top image, the closing edge image in the middle image, and the binary lane mask in the bottom image.
where

\[
\tilde{M}_1 = \frac{1}{n_1 n_2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} M_1(i, j)
\]

\[
\tilde{M}_2 = \frac{1}{n_1 n_2} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} M_2(i, j).
\]

(2.11)

Examples of the binary lane masks are shown in Figure 2.10. The sample correlation coefficient for each pair of these masks are shown in Table 2.1.

<table>
<thead>
<tr>
<th>A pair of masks</th>
<th>(r_{M_3, M_k})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(M_1, M_2)</td>
<td>0.5583</td>
</tr>
<tr>
<td>(M_2, M_3)</td>
<td>0.9368</td>
</tr>
<tr>
<td>(M_3, M_4)</td>
<td>0.0478</td>
</tr>
</tbody>
</table>

**2.3 Moving vehicle tracking**

The pre-processed image sequence is processed through the moving vehicle tracking procedure when the similarity of the two consecutive masks is greater than the threshold. The moving vehicle tracking procedure includes three modules: (1) Spatio-Temporal segmentation, (2) moving vehicle detection, and (3) moving blob tracking. (See Figure 2.11.)

**2.3.1 Spatio-Temporal segmentation**

Both the intensity edge detection and the differencing edge detection identify edge boundaries. However, they infrequently provide complete connected edges. To acquire more edge boundaries from the intensity and the differencing edge detection, we consider both results together within a small neighborhood (7x7):

1. A moving edge detection algorithm stated earlier is applied to get a differencing edge image \(ME(u, v)\).
2. The differencing edge image \(ME(u, v)\) is dilated by a 7x7 kernel.
3. A Sobel edge detection and a threshold operation is applied to a current image for a binary intensity edge detection within the lane mask region.
4. An intersection of the output image from (3) and the dilated moving edge image from (2) creates some connections between the intensity edges and the moving edges within small neighborhoods.
5. A union of the output image from (4) and the differencing edge image creates the final combination of these extracting edges.
Figure 2.10: The binary lane masks $M_1, M_2, M_3$, and $M_4$ from top to bottom, respectively.
2.3.2 Moving vehicle selection

The moving vehicle selection includes the object selection and the blob’s area indexing. (See Figure 2.14.) The object selection identifies the moving blobs. This process is the same as the object selection in the lane mask region. Since an individual vehicle’s edge image is smaller than the collection of vehicle edges in lane mask generation, for the vehicle closing operation, we estimate the moving blobs object $CL$ by using two different structure elements: (1) an 8x5 kernel structure element $B_1$ and (2) a 2x2 kernel structure element $B_2$. We apply a closing operation using $B_1$ to the Spatio-Temporal edge image $MS$. The output from this closing operation is refined by dilation three times followed by erosion three times, using $B_2$,

$$CL = (((((MS \cdot B_1) \oplus B_2) \oplus B_2) \oplus B_2) \ominus B_2) \ominus B_2).$$  \hspace{1cm} (2.12)

Next, the convex hull module is applied to make the bounding polygon of moving vehicles, which are binary masks for the moving vehicles. The blob’s area indexing indicates the appropriate size of the moving blob relative to its neighborhood. Every moving object $MO$ is identified by its centroid position $(X, Y)$. The area of each blob is calculated and is indexed by the vertical location of its centroid in the image. This area is an indicator of blob size relative to its neighborhood. We use only blobs whose size is greater than 7x7 pixels but less than two and a half times the area for that neighborhood. To guarantee that the sizes of the selected blobs are significant enough, we select only the blobs that have their centroid positions located at the row at the bottom-half of the image.
Figure 2.13: Typical image sequence (left). Spatial edge image in the right top image, and the Temporal edge in the middle right image. Moving edge image for the middle image on the left created by the Spatio-Temporal segmentation module in the bottom right image.

Figure 2.14: Moving vehicle selection.
2.3.3 Moving blobs tracking

The selected moving blob $MB$ is identified by its centroid position $(X, Y)$. The centroids in the image at times $i-1$, $i$, and $i+1$ are labeled “a,” “b,” and “c” with the centroids $(x_1, y_1), (x_2, y_2), (x_3, y_3)$. Assuming that the same car is moving straight line, the “a,” “b,” and “c” points should form straight lines within a certain length. Therefore, we constrain the search area within a blob’s length. Figure 2.15 shows the moving blobs tracking flow diagram.

The moving blobs are tracked using the linear regression correlation coefficient of the series “a,” “b,” and “c.” A set of centroids is associated if the sample correlation coefficient of $X$ and $Y$, $r_{xy}$, is at least 0.95,

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\left[\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2\right]^{1/2}}, \quad (2.13)$$

where

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i,$$
$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i.$$

(2.14)
Sample Correlation Coefficient

\[ C(X_i, Y_i) \]

\[ \text{dist}_1 = \text{norm}[(X_a, Y_a), (X_b, Y_b)], \]
\[ \text{dist}_2 = \text{norm}[(X_b, Y_b), (X_c, Y_c)], \]

\[ \text{dist}_1, \text{dist}_2 < \text{threshold}, \]

\[ r_{xy} \]

\[ r_{xy} > \text{threshold} \]

\[ C(X_i, Y_i) \]

\[ C^*(X_i, Y_i) \]

Figure 2.15: Moving blobs tracking.
Chapter 3

Camera Calibration and Mean Speed Estimation

Camera calibration establishes a relationship between the earth coordinate system (3D) and the image coordinate system (2D). Full camera calibration is a complex problem with many parameters. For example, the camera eyepoint location, the camera height, Euler angles, orientation of the camera (azimuth, elevation, and roll), and focal length need to be estimated in parallel for a particular camera pose. The calibration problem considered here is further complicated by the ability of the operation staff of the highway management agency to pan, tilt, and zoom the un-calibrated roadside cameras. Many researchers have contributed precise 3D-2D transformation algorithms [17, 18, 19, 20, 21, 22]. Full calibration normally requires solving for a large number of related parameters, typical using an iterative optimization along with an initial set of parameters sufficiently close to the solution so that the objective function is quadratic in the region between the initial guess and the solution. However, in the work presented we will demonstrate a simplified form of calibration to estimate a mean traffic speed.

3.1 Affine camera calibration model

An estimation of the camera calibration parameters is a very important step in camera tracking problems. Figure 3.1 shows the geometry of the problem. We assume that the horizontal axis of the image plane is parallel to the ground plane and that the camera’s image axis orientation is at a down angle of $\phi$ with respect to the horizontal. We use two coordinate systems as defined in [23]: (1) a camera/sensor centered coordinate system as shown at the top of Figure 3.1, where the $x_s$ axis is the line of sight of the camera and the $y_s - z_s$ plane is parallel to the image plane and (2) an earth-fixed system as shown at the bottom of 3.1, where the ground plane is defined by $x_e$ and $y_e$ with the vertical axis $z_e$ downward.

The affine transformation is defined as a mapped transformation between the sensor/camera coordinate system and the earth coordinate system. It composes of a sensor to earth rotation matrix $H_{se}$ and a translation of the eyepoint of the camera vector $B$,

$$X_e = H_{se} X_s + B,$$  \hspace{1cm} (3.1)
where

\[
X_e = \begin{pmatrix} x_e \\ y_e \\ z_e \end{pmatrix}, \quad X_s = \begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix}, \quad \mathbf{H}_{se} = \begin{pmatrix} \cos \phi & 0 & -\sin \phi \\ 0 & 1 & 0 \\ \sin \phi & 0 & \cos \phi \end{pmatrix}, \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} 0 \\ 0 \\ -h \end{pmatrix}.
\]

(3.2)

The earth to sensor rotation \( \mathbf{H}_{es} \) is \( \mathbf{H}_{se}^{-1} \). Since \( \mathbf{H}_{se} \) is orthogonal, \( \mathbf{H}_{es} \) is \( \mathbf{H}_{se}^T \) [24]. A basis of unit vectors in the earth coordinate system is defined as: (1) \( \mathbf{L}_e \) is a unit vector along the roadway, (2) \( \mathbf{W}_e \) is a unit vector perpendicular to the roadway, and (3) \( \mathbf{K}_e \) is a unit vector in the vertical direction. (See Figure 3.2.) We also define a set of coordinates \( l, w, k \) along these directions.

\[
\mathbf{L}_e = \begin{pmatrix} \cos \theta \\ \sin \theta \\ 0 \end{pmatrix}, \quad \mathbf{W}_e = \begin{pmatrix} -\sin \theta \\ \cos \theta \\ 0 \end{pmatrix}, \quad \mathbf{K}_e = \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix},
\]

(3.3)

where \( \theta \) is relative to the \( x \) axis in earth coordinates. A differential change in position along the road in earth coordinates is \( d\mathbf{X}_e = \mathbf{L}_e \, dl \), where \( dl \) is the magnitude of the differential distance along the road. The same change in position in sensor coordinates is

\[
d\mathbf{X}_s = \mathbf{H}_{es} (\mathbf{L}_e \, dl).
\]

(3.4)

We define \((u, v)\) as the location in the image plane and assert a simple perspective relationship
between the 3-D world and the image plane,

$$u = b \frac{y_s}{x_s}, \quad v = b \frac{z_s}{x_s},$$  \hspace{1cm} (3.5)

where $b$ is a scaling constants. Using these perspective relationships and differential calculus, we can obtain an expression for differentials in the image plane,

$$du = \frac{(bdy_s - udx_s)}{x_s}, \quad dv = \frac{(bdz_s - vdx_s)}{x_s}. \hspace{1cm} (3.6)$$

Since the vehicles are moving on the road so $z_e = 0$ for a vehicle, using equations (3.1), (3.2), and (3.5), we find that $x_s$ is

$$x_s = \frac{bh}{v \cos \phi + b \sin \phi}. \hspace{1cm} (3.7)$$

We use the projection of the vehicle size/motion on the $u$ and $v$ axes as the measurement of interest. First, combining equations (3.4), (3.5), (3.6), and (3.7), we construct the differential of changes in the $u$ direction in three directions: (1) along the line of motion of the vehicles, $\partial l$, (2) in a perpendicular to the line of vehicle motion (horizontal), $\partial w$, and (3) in a radially outward from the earth’s center (up), $\partial k$,

$$\frac{\partial u}{\partial l} = \frac{(v \cos \phi + b \sin \phi)(-u \cos \phi \cos \theta + b \sin \theta)}{bh}$$  \hspace{1cm} (3.8)

$$\frac{\partial u}{\partial w} = \frac{(v \cos \phi + b \sin \phi)(b \cos \theta + u \cos \phi \sin \theta)}{bh}$$  \hspace{1cm} (3.9)

$$\frac{\partial u}{\partial k} = \frac{u \sin \phi(v \cos \phi + b \sin \phi)}{bh}. \hspace{1cm} (3.10)$$

Second, in similar fashion, we combine equations (3.4), (3.5), (3.6), and (3.7), we can construct the differential relationship between motion in $\vec{L}_e, \vec{W}_e$, and $\vec{K}_e$ and changes in the $v$ direction in the image plane,
\[
\frac{\partial v}{\partial l} = -\frac{\cos \theta (v \cos \phi + b \sin \phi)^2}{bh} \quad (3.11)
\]
\[
\frac{\partial v}{\partial w} = \frac{(v \cos \phi + b \sin \phi)^2}{bh} \sin \theta \quad (3.12)
\]
\[
\frac{\partial v}{\partial k} = \frac{(v \cos \phi + b \sin \phi)(v \sin \phi - b \cos \phi)}{bh}. \quad (3.13)
\]

If the mean vehicle length \(\Delta \bar{l}\), mean vehicle width \(\Delta \bar{w}\), and mean vehicle height \(\Delta \bar{k}\), as taken from [26], are used with the differentials in (3.8), (3.9), and (3.10), the projection of a vehicle on the roadway in the \(u\) direction in the image plane can be written,

\[
\Delta u \approx \Delta \bar{l} \frac{\partial u}{\partial l} + \Delta \bar{k} \frac{\partial u}{\partial k} + \Delta \bar{w} \frac{\partial u}{\partial w}. \quad (3.14)
\]

The mean length and height can be expressed as a fraction of the mean width, \(\Delta \bar{l} = c_l \Delta \bar{w}\) and \(\Delta \bar{k} = c_k \Delta \bar{w}\),

\[
\frac{\Delta u}{\Delta \bar{w}} \approx \left\{ c_l \frac{\partial u}{\partial l} + c_k \frac{\partial u}{\partial k} \right\} + \frac{\partial u}{\partial w}. \quad (3.15)
\]

Using the same notation, the mean vehicle length \(\Delta \bar{l}\), mean vehicle width \(\Delta \bar{w}\), and mean vehicle height \(\Delta \bar{k}\), as taken from [26], are used with the differentials in (3.11), (3.12), and (3.13), and the projection of a vehicle on the roadway in the \(v\) direction in the image plane can be written,

\[
\Delta v \approx \Delta \bar{l} \frac{\partial v}{\partial l} + \Delta \bar{k} \frac{\partial v}{\partial k} + \Delta \bar{w} \frac{\partial v}{\partial w}. \quad (3.16)
\]

The mean width and height can be expressed as a fraction of the mean length, \(\Delta \bar{w} = a_w \Delta \bar{l}\) and \(\Delta \bar{k} = a_k \Delta \bar{l}\),

\[
\frac{\Delta v}{\Delta \bar{l}} \approx \frac{\partial v}{\partial l} + \left\{ a_k \frac{\partial v}{\partial k} + a_w \frac{dv}{\partial w} \right\}. \quad (3.17)
\]

### 3.2 Scale modeling

Using (3.15) with equations (3.8), (3.9), and (3.10), we get a linear function for the scale factor \(q(u) = \frac{\Delta \bar{w}}{\Delta \bar{w}}\) in \(u\),

\[
q(u)^{-1} = \frac{\Delta u}{\Delta \bar{w}} \approx u \left[ \frac{(v \cos \phi + b \sin \phi)(c_k \sin \phi + \cos \phi(-c_l \cos \theta + \sin \theta))}{bh} \right] - \left[ \frac{(v \cos \phi + b \sin \phi)(\cos \theta + c_l \sin \theta)}{h} \right]. \quad (3.18)
\]

In a similar fashion, we get a quadratic expression for a scale factor \(q(v) = \frac{\Delta \bar{l}}{\Delta \bar{w}}\) in \(v\),

\[
\text{In a similar fashion, we get a quadratic expression for a scale factor } q(v) = \frac{\Delta \bar{l}}{\Delta \bar{w}} \text{ in } v,
\]
\[ q(v)^{-1} = \frac{\Delta v}{\Delta l} \approx v^2 \left[ \frac{\cos \phi (a_k \sin \phi + \cos \phi (a_k \sin \theta - \cos \theta))}{bh} \right. \\
- v \left[ \frac{a_k \cos 2\phi + \sin 2\phi (a_k \sin \theta - \cos \theta)}{h} \right] \\
- \left[ \frac{b \sin \phi (a_k \cos \phi + \sin \phi (\cos \theta - a_k \sin \theta))}{h} \right]. \]  

(3.19)

Assuming vehicles travel upward or downward in direction, \( q(u) \) is horizontal scaling and \( q(v) \) is vertical scaling. \( q(u) \) is a ratio mapping of the mean width of the car, \( \Delta \bar{w} \), and the blob’s width \( \Delta u \). \( q(v) \) is a ratio mapping of the mean length of the real car, \( \Delta \bar{l} \), and the blob’s height, \( \Delta v \),

\[ q(u|v) = \frac{\Delta \bar{w}}{\Delta u}, \]  

(3.20)

A vertical scaling, \( q(v) \), is a ratio mapping of the length of the real car and the blob’s length, \( \Delta v \).

\[ q(v) = \frac{\Delta \bar{l}}{\Delta v}. \]  

(3.21)

A training set of 10-second video sequences will be collected. A computed scaling value as shown in (3.20) and (3.21) will be assigned for each selected “a,” “b,” and “c” sequences.

### 3.3 Linear scaling model estimations

We observe that the general form for the scale factors can be written:

\[ q(u|v) = \frac{1}{a_1(v) + a_2(v)u}, \]

\[ q(v) = \frac{1}{b_1 + b_2v + b_3v^2}. \]  

(3.22)

These inverse polynomial forms can be approximated using a Taylor series as,

\[ q(u|v) \approx c_1(v) + c_2(v)u, \]

\[ q(v) \approx d_1 + d_2v + d_3v^2 + d_4v^3. \]  

(3.23)

In order to simplify fitting data sets, the coefficients for this simple camera calibration model are estimated using a selected tracking sequence, where \( \{u_1, u_2, \ldots, u_n|v\} \) is a set of centroid columns at a given row \( v \); and \( \{v_1, v_2, \ldots, v_n\} \) is a set of centroid rows, which generate each corresponding scaling. A least squares approach [27] is used to obtain quantitative values for the horizontal scaling coefficients,
\[
Y = [q(u_1), q(u_2), \ldots, q(u_n)]^T
\]
\[
Y_i = P_i C
\]
\[
P = [P_1 \ P_2 \ \ldots \ P_n]^T
\]
\[
where \ C = [c_2 \ c_1]^T \ and \ P_i = [u_i \ 1]
\]
\[
C = [P^T P]^{-1} P^T Y. \quad (3.24)
\]

In similar fashion, we can approximate the coefficients for the vertical scaling \( q(v) \) using the same least square approach,
\[
Y' = [q(v_1), q(v_2), \ldots, q(v_n)]^T
\]
\[
Y'_i = P'_i C'
\]
\[
P' = [P'_1 \ P'_2 \ \ldots \ P'_n]^T
\]
\[
where \ C' = [d_4 \ d_3 \ d_2 \ d_1]^T \ and \ P'_i = [v^3_i \ v^2_i \ v_i \ 1]
\]
\[
C' = [P'^T P']^{-1} P'^T Y'. \quad (3.25)
\]

### 3.4 Travel distance approximation and mean speed estimation

#### 3.4.1 Travel distance approximation

We estimate the horizontal and vertical scaling factors from (3.23) using a training set of 10-second video sequences. The distance traveled “\( d \)” is the combination of the integral of those functions in horizontal and vertical along \( u \) and \( v \) direction, respectively,
\[
d = \sqrt{\left( \int_{v_1}^{v_2} q(v) dv \right)^2 + \left( \int_{u_1}^{u_2} q(u) du \right)^2}. \quad (3.26)
\]

#### 3.4.2 Mean speed estimation

We estimate the vehicle speed for a distance traveled \( d \) at a given time \( \Delta t \),
\[
\hat{S} = \frac{d_k}{\Delta t}. \quad (3.27)
\]

The mean speed estimation \( \bar{S} \) is an average of the estimated speed \( \hat{S} \),
\[
\bar{S} = \frac{1}{n} \sum_{i=1}^{n} \hat{S}_i. \quad (3.28)
\]
Chapter 4

Simulation and Empirical Results

To demonstrate the effectiveness of our simplified camera model, we present two types of validation. First, we compare the results with simulated data, and then we present results from real roadside cameras.

We create a simulated set of camera data by defining a set of rectangular parallelepipeds (cuboids) in a 3-D space and projecting these cuboids onto a 2-D image plane using a complete perspective transformation based on a camera model and perspective transformation in [25, 12]. We create a sequence of images by moving the cuboids toward the camera location in 3-D space and sequentially projecting the result onto the image plane. To simulate vehicles of various lengths, the cuboid lengths are varied about the mean length using values from a normal distribution with a standard deviation of 10% of the mean value. We used our tracking algorithm with a series of artificial images. Examples of which are shown on the left of Figure 4.1, and their corresponding blobs are shown on the middle of Figure 4.1. The cuboids are detected and tracked from their centroids as shown on the right of Figure 4.1.

We calculated scaling factors, as presented earlier, for this simulation. The horizontal scaling factor is shown on the left of Figure 4.2, and the vertical scaling factors are plotted on the right of Figure 4.2. The distribution of the error in the speed estimate is shown in Figure 4.3 on a scale normalized to be of the standard deviations. The mean of the distribution is near the true speed, and the variability of the vehicle length is reflected in the variability of the speed estimate. This demonstrates that the algorithm presented works in principle as long as the speed estimates are recognized as a random variable and only expected values are used as estimates of the traffic speed.

Similarly, we applied our algorithm to the actual traffic scenes, some examples are shown in Figure 4.4. The algorithm correctly associates the objects between images and tracks the moving objects as shown on the right of Figure 4.4. The horizontal scaling factor for actual traffic scenes is shown on the left of Figure 4.5, and the vertical scaling factor of the actual traffic scenes is shown on the right of Figure 4.5. Inductance loop data is available near the camera location from which the traffic scenes were obtained. The deviation between the camera speed estimates and the loop speed estimates were constructed using speed data from the Traffic Data Acquisition and Distribution (TDAD) data mine (http://www.its.washington.edu/tdad). In particular, Figure 4.6 is the distribution of the difference between 20-second video speed estimates and 20-second averaged inductance loop speed estimates.
Figure 4.1: Simulation cuboids (left). The cuboid binary blobs (middle). Tracking line for moving cuboids (right middle image).

Figure 4.2: Horizontal scaling estimation of the simulation data(left). Vertical scaling estimation of the simulation data(right).
Figure 4.3: Histogram of speed error of the simulation speed and the reference speed over the standard of deviation.

Figure 4.4: Typical image sequence (left). The binary blobs (middle). Line tracking of the blobs’ centroids sequence (right image).
Figure 4.5: Horizontal scaling estimation of the actual traffic data (left). Vertical scaling estimation of the actual traffic data (right).

Figure 4.6: Histogram of speed error of the actual traffic speed and the inductance loop speed.
Chapter 5

Conclusion and Future Work

5.1 Conclusion

We present a computer vision algorithm that uses information from a freeway traffic scene to perform dynamic camera calibration and make mean vehicle speed estimates. We assert that our algorithm is suitable for use with un-calibrated CCTV cameras used by freeway management agencies. Our camera calibration procedure creates a relationship between the image coordinate and the world coordinate systems without deploying any registration marks in the physical scene. Changes in binary masks created from image sequences are used as a decision tool to update camera calibration parameters. We use the Spatio-Temporal segmentation to identify the moving vehicles. The Spatio-Temporal segmentation technique presented produces fewer broken edges than using only Temporal segmentation.

Our camera calibration provides a simple way of making a relationship between the image coordinate and the world coordinate without any point correspondence. We demonstrate that the algorithm works well not only with simulated data but also with the actual traffic data. The algorithm provides speed estimates similar to inductance loop estimates over the same time period. However, the result shows that the difference between our speed estimation and the inductance loop estimation has a negative bias. That means two things: (1) there are many large vehicles present in the scene during acquiring the image sequence acquisition, and (2) there is some occlusion between vehicles; our indexing object module has failed to reject these integrated vehicle blobs. Some occlusion is allowed in our method. Our vehicle-tracking algorithm is negatively affected by occlusion.

5.2 Future work

To make the dynamically measuring mean vehicle speed algorithm more efficiency, several issues should be addressed and resolved.

5.2.1 Various weather conditions

The pre-processing process improves and adjusts the image quality according to various weather conditions (e.g., cloudy, rainy, bright, and shadow scenes). The decision criteria for image adjustment for these conditions is very important prior to the main algorithm. The pre-processing
process must have the ability to account for these environmentally induced variations to enhance image quality.

5.2.2 Improve lane mask detection criteria

The similarity of the binary masks determines the changing state of a camera’s parameters. Further experiment of the statistical similarity result of decision criteria needs to be address. There are pros and cons of deciding a specific criteria. The extension study on those issues will improve a decision making tool.

5.2.3 Vehicle classification

An image recognition algorithm that differentiates between a passenger car, a truck, and a multi-truck can make the camera calibration more accurate. [28] introduces a simple vehicle classification method based on a rule-based classifier from a vehicle’s dimension and its features. The algorithm classified three major subclasses: small, medium, and large vehicle based on a variety of conditions, such as day-time, night-time, and raining. We can adopt and extend this approach to assist our algorithm. If we can separate vehicles into groups, the algorithm can select the appropriate mean vehicle dimension accordingly. These statistical results have less dimensional variance. Therefore, the mean speed approximation produces less error variance.

5.2.4 Occlusion reasoning in congested traffic

The most difficult problem associated with vehicle tracking is the occlusion effect among vehicles. A heavily populated highway traffic scene can be seen in urban area during rush hours. It is quite crucial to detect the mean speed of that area and send some signals to the traffic control system. Then, the traffic control operators will be able to locate an exact position for that traffic. They can manage the city’s traffic signals based on this information.

Currently, our algorithm rejects some of those integrated-vehicle blobs. In congested traffic scene, the algorithm can easily obtain the wrong number of vehicles per blob or reject most of the useful data set. The occlusion reasoning will be able to make this type of data set available for further computation.
Bibliography


