A SYSTEM FOR 3D VEHICLE TRACKING USING MULTIPLE CAMERAS

A THESIS

SUBMITTED TO THE FACULTY OF THE
UNIVERSITY OF MINNESOTA
BY

SUKRITI SUBEDI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

ADVISOR: DR. HUA TANG

OCTOBER 2017

ACKNOWLEDGEMENT

I wish to acknowledge those who made this research possible. The initial phase of the thesis work back in 2012 was funded by the Intelligent Transportation Systems (ITS) Institute, a program of the University of Minnesota's Center for Transportation Studies (CTS), and was also supported by the Northland Advanced Transportation Systems Research Laboratories (NATSRL), a cooperative research program of the Minnesota Department of Transportation, the ITS Institute, and the University of Minnesota Duluth Swenson College of Science and Engineering. I also thank St. Louis County for their support to set up a local test laboratory at the intersection of W Arrowhead Rd and Sawyer Avenue, and W Arrowhead Rd and Arlington Avenue in Duluth MN back in 2012. I appreciate Mr. Victor Lund's help with the test laboratory and Dr. Eil Kwon, director of Northland Advanced Transportation Systems Research Laboratory (NATSRL) at University of Minnesota Duluth, for very valuable discussions and comments.

I highly appreciate the help and encouragement provided by my thesis advisor Dr. Hua Tang throughout the thesis work. I also appreciate the support of committee members, Dr. Eil Kwon and Dr. Mohammed Hasan. Finally, I thank my husband, Dr. Puskal Pokharel, for his encouragement and unconditional support throughout this arduous process. His patience and constant guidance, even when I saw the challenges to be insurmountable, kept me moving along to reach this goal.

ABSTRACT

In this thesis, we present a 3D vehicle tracking system that allows 3D vehicle dimension estimation using fusion of information from multiple cameras. Compared to traditional 2D vehicle tracking systems with a single camera, the 3D tracking system can potentially improve tracking accuracy in the presence of similar noise, shadow or vehicle occlusion.

Taking synchronized images from multiple cameras as inputs, the developed 3D vehicle tracking system first processes image frames using image processing techniques to derive vehicle silhouettes. Using a square roadside pattern derived from parallel traffic lanes, multiple cameras are calibrated using an extended vanishing points based approach and post-optimization by minimizing the projection error. After segmentation, objects in image frames are projected to the real world, in which the overlap of the same object would ideally correspond to the chassis of the object, and this allows estimation of vehicle length and width. Once vehicle length and width are obtained, they can be further used in each image frame to estimate the vehicle height. The base of the vehicle can be represented as an oriented rectangle. The height of the object is sought in the image view that would create the top of the hyper rectangle of the 3D vehicle that has the edge furthest away from the vehicle base rectangle. In the 3D vehicle tracking process, Kalman filter is used to predict the state of the vehicle in the real world to allow data association of the vehicles.

Experiment results using realistic traffic videos of an intersection from two cameras have shown that the 3D vehicle tracking is fully functional. Vehicle dimension estimation results are mostly consistent in spite of some error caused by camera calibration and vehicle segmentation.

Table of Contents

List of Figures

Chapter 1 Introduction

Video surveillance of vehicular movement on public roads has been an active research topic in intelligent transportation systems community [1] [2] [3] [4]. A variety of methods have been proposed to use sensors, loop inductors, radars or cameras for video surveillance and traffic data collection. Among sensors, loop inductors, radars or cameras for intersection traffic data collection, visual information provided by cameras can potentially provide more comprehensive information on vehicle behavior during the vehicle's life time within the camera view, which allows more accurate traffic data collection. On the other hand, camera-based vision system has additional advantages, such as non-intrusiveness as they can be readily mounted on traffic light poles or other stationary platforms, easier management as they can be controlled remotely, lower cost as cameras are getting cheaper, smaller and smarter [5], and higher computing efficiency thanks to ever-increasing computing power of modern hardware processors. Therefore, camera-based vision system has been widely used today across the world to provide traffic videos which are processed either online or offline to provide traffic data on vehicular movement at highways, intersections and roundabouts. This can be an important source of traffic statistics for transportation engineers to improve traffic safety [6].

Deriving the trajectory of vehicular movement is an important component of traffic data collection in a camera based vision systems. Two major tasks involved here are vehicle detection or segmentation and vehicle tracking, which are closely related and in practice can be considered as a combined single task [5]. This has resulted in active research on

vehicle detection and tracking, which can be classified into edge based, region based, optical flow based, and feature based approaches [7] [8]. Two most widely used methods are region based tracking and edge based tracking. The simpler of the two is region based, that extracts an intermediary representation of the vehicle as a silhouette or blob from video frames, whereas an edge based algorithm extracts an intermediary representation of the vehicle as image edges from the video frames. The edge-based approaches are considered more robust to variation in brightness and image noise [9]. Still the region based approach is preferred because of the relatively lower complexity and consistent results [8]. In this thesis, we have tried to combine aspects of the edge based method and region based method for vehicle segmentation. The region-based method for vehicle segmentation is powered by the Mixture-of-Gaussian (MoG) algorithm, which has been widely used in object detection [10]. The final silhouette is formed by intersecting the blob representation produced by the MoG algorithm and blob representation obtained by filling the inside of estimated vehicle edges. This appears to provide a more accurate and reliable representation of the vehicle outline (benefit of edge based methods) but also avoid miss of detection of vehicle caused by various sources of noise and background artifacts (benefit of region based methods using the MoG algorithm).

When it comes to the actual vehicle tracking and association as vehicles move across the camera view, the workhorse in literature has been the Kalman filter that can efficiently combine noisy observations with a predefined model of vehicle movement [2] [4] [9] [11] so that vehicle silhouettes from the same vehicle can be properly corresponded. Authors in [8] suggest an alternative to the Kalman filter for vehicle tracking using a method of Monte Carlo sampling instead. One of the suggested benefits of the Monte Carlo method

is that it is not restricted to the Gaussian distribution of data, or assumptions of linearity, which is the case for the Kalman filter. However, most publications show acceptable or good performance of the Kalman filter [9] [11]. Given the widely accepted use and effectiveness of the Kalman filter we have used it for 3D tracking of the vehicle with multiple cameras in this thesis.

In this thesis, we have built on the work that was started by Hai Dinh in [2]. Hai Dinh in [2] reported a system for traffic data collection for roundabouts with a single camera, which is essentially a 2D tracking system without actual vehicle dimension estimation. The tracking system in [2] included region-based vehicle segmentation using the MoG algorithm followed by a shaking-removal algorithm to avoid false detection from camera shaking [2]. For our current work we have started with the segmentation and detection of the vehicle based on methods described in [2] and also combined it with the edged-based method for more accurate vehicle segmentation. The main difference is that we implemented a 3D tracking system with multiple cameras in contrast to 2D vehicle tracking with a single camera in [2]. With this work, the following enhancements and capabilities have been added compared to previous work [2]:

- Use of multiple cameras which improves the performance in the presence of shadow and partially occluded vehicles.
- World view tracking of vehicle giving the vehicle attributes (dimensions and speed) in real world units like feet and miles per hour.
- A combined region-based method using the MoG algorithm and edge-based method for more accurate and reliable vehicle detection or segmentation.

Below is a table with a list of literature examples along with a comparison.

Table 1: Quick list of representative vehicle tracking systems along with this thesis

System	Camera Setup	Tracking View	Tracking Method	Related Features
[2]	Single Camera	Camera View	Region based; Kalman tracking in camera view.	This thesis work builds on top of several aspects of this work with extension to multiple cameras and extraction of 3D vehicle dimensions in World view
[9]	Single Camera	Camera View	Edge based with fitting polyhedral vehicle model; extended Kalman filter tracking.	Vehicle 3D dimensions estimated through predefined vehicle models and needs to run multiple iterations to reach optimal results with relatively high computation burden.
[12]	Single Camera	Camera View	Edge based detailed vehicle model selection including parts like tires and windows. Tracking using iterated extended Kalman filter.	Vehicle 3D dimensions estimated by selecting and fitting an elaborate dictionary of precise vehicle models. Stable convergence is not guaranteed as with closed-form solutions and may require multiple iterations. Initial conditions required to be set carefully.
[8]	Single Camera	Camera View	Region based tracking using Monte Carlo or Bayesian estimation (treat all points in trajectory conditionally related) instead of Kalman filter.	Vehicle dimension estimation not primary focus. Chooses one of three vehicle parallelepiped models.
[3]	Multiple Cameras	World View	3D region based using probability fusion map with overlap in world view. Use of Kalman tracking or alternative not clear.	Estimation of length and breadth is similar to our work, but specific estimation of height is not presented (simply suggesting "blob analysis," without any details). Requires careful selection of several extra parameters to properly weigh the camera probabilities.
This Thesis	Multiple Cameras	World View	3D region based using overlap of multiple cameras in world view. Uses Kalman filter for tracking.	Simple direct solution for orientation, length, breadth, and height estimation. No convergence issues as with model fitting or Monte Carlo methods. Fewer number of user defined tuning parameters.

In addition, we have also built a Matlab® based analysis and recording system that tracks and stores the vital statistics of vehicles travelling through the camera frames. In summary, our traffic monitoring system can be represented with the block diagram in Figure 1.

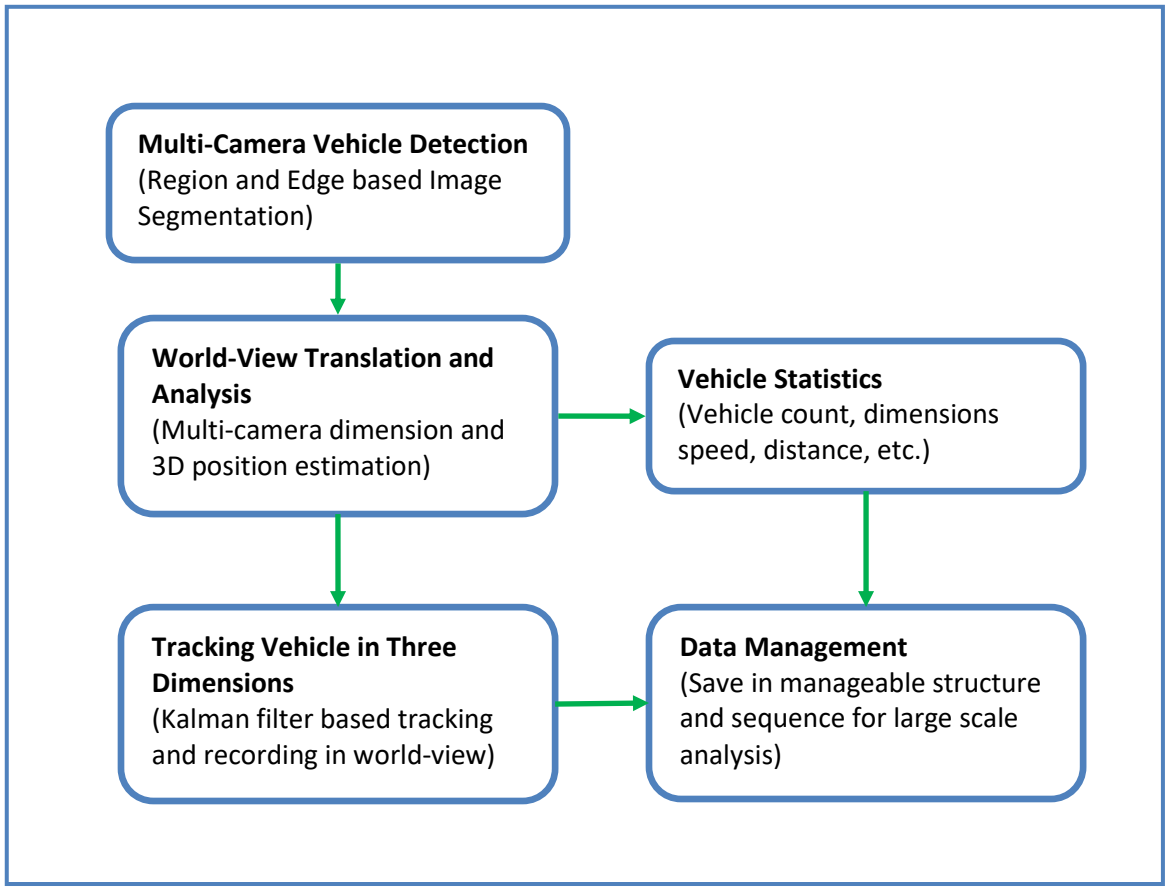

Figure 1: Proposed traffic monitoring system block diagram.

Chapter 2 Multi-Camera Vehicle Detection

One of the first steps of multi-camera vehicle detection is segmentation in order to remove the background and provide a reasonable representation of the vehicle. There are various methods for background removal [13] used in literature. One may use the MoG method employed by [2] or some form of edge detection, for example [14]. When both methods result in extraction of blobs or silhouettes of vehicles, we can combine them to get more accurate representation of vehicle. We can use the intersection or common portion of the two blob representations to get a tighter representation of the vehicle. Generating a silhouette representation through edge detection is explained next.

The immediate next step is to fit a rectangle of proper size and orientation that encloses the obtained silhouette in world view. We have derived an algorithm from first principles (based on geometrical analysis and linear algebra) to search for the smallest rectangle that contains the vehicle silhouette. This is the first step in generating the length, width and height of a vehicle, which is explained in subsequent sections.

2.1 Extraction of Vehicle Silhouette Image

Let us discuss generating an accurate silhouette representation of a vehicle by extracting the edges of the vehicle in the video image frame and then filling the associated convex region containing the edges.

2.1.1 Statistical Mode Based Background Removal

One of the first components of extracting the silhouette is the removal of background. For our video data base, a simple effective method is to construct the background based on the most frequently occurring pixel values. We propose to use statistical calculation of mode [15] for estimating the background similar to [1] [16]. The use of mode is more appropriate in this case because the most occurring pixel value is affected less by the occasional outliers, and avoids the smearing effect of simple average.

Though the intensity of pixel values will change over long periods of time, for instance due to the change of the position of the sun or movement of clouds, such changes are relatively very slow compared to the frame rate of the video. In order to accommodate for these slow changes in the background, a moving time-window of frames can be used for the calculation of the modes to estimate the background. This is explained next. Figure 2, Figure 3 and Figure 4 show the sequence of images resulting from this process starting with an example frame in Figure 2.

The mode of a population of data samples is defined as the sample value that occurs the most or the most frequent data point [15]. Using the mode of each pixel over a moving window of N frames, the background is constructed with the value at the ij^{th} pixel of the frame at time m given by,

$$B_{ij}(m) = \underset{m-\left\lfloor \frac{N}{2} \right\rfloor < n < m+\left\lceil \frac{N}{2} \right\rceil - 1}{\text{Mode}} [F_{ij}(n)] \quad (1)$$

where $F_{ij}(n)$ is the value of the ij^{th} pixel of the frame of the video at time n.

Figure 2: A frame of the video showing a vehicle.

Figure 3: Estimate of the background of Figure 2 based on the mode of each pixel over 30 frames.

Figure 4: Resulting image with the mode based background removal in Figure 2.

2.1.2 Silhouette Construction Using Edge Detection and Fill

Silhouette construction involves using a reliable method of edge detection [17] [18]. Since the Prewitt operator is sensitive towards the high frequency changes as it approximates the gradient of the image intensity [17], we have found it to be an effective edge detection method. The Prewitt method of edge detection (an option in Matlab® function *edge* [19]) involves filtering the image with an approximation to the gradient operator in the horizontal and vertical directions given by the masks G_x and G_y and a threshold is used to select the pixels that constitute the edges in image [18].

$$G_x = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$
$$G_y = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad (2)$$

This is then followed by image processing such as morphological closing (e.g., using Matlab® function *imclose*) that creates a silhouette by filling the segments within the edge boundaries [18]. The last step is to create a convex silhouette by computing a convex hull of the silhouette. This involves finding the smallest (area) convex polygon than contains the silhouette, with all pixels within the polygon filled in. Such a polygon is obtained by iteratively checking if a point in the silhouette lies inside or outside the current polygon and adjusting it for the next iteration until all points lie inside the resulting polygon, which is done efficiently by using the Matlab® function *bwconvhull* [19]. Figure 5 - Figure 7 show the result of these steps using the image in Figure 4. This method provides a tight silhouette representation of the vehicle that can be used to estimate the vehicle dimensions in the real world view, which will be discussed later

Figure 5: Result of edge detection on image of Figure 4 using the Prewitt operator.

Figure 6: Result of morphological closing on image of Figure 5 using the Prewitt operator.

Figure 7: Estimated convex hull from image in Figure 6.

2.2 Axis-aligned rectangle with the least area around silhouette

In order to estimate the length and breadth of an object, we propose to construct a rectangle with the least area containing the object's silhouette. Without loss of generality, this problem is simplified by considering an object that is aligned with the reference Cartesian axes. By axis aligned, we mean that the object's longest dimension is parallel to one of the axes. In such case, as shown in Figure 8, a rectangle can be obtained by using the four points corresponding to the maximum and minimum y-coordinate value, and maximum and minimum x-coordinate value.

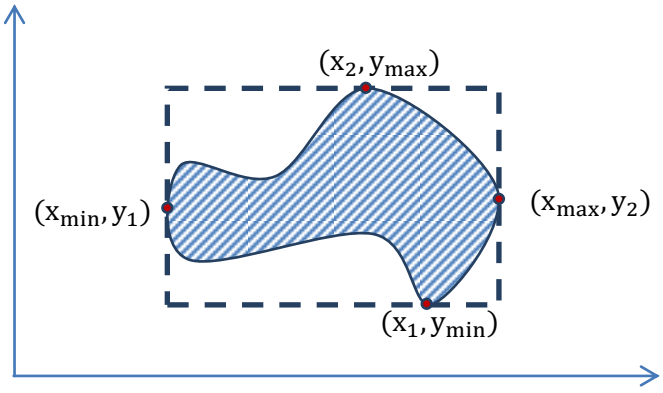

Figure 8: Example of an axis aligned rectangle.

Let the minimum and maximum x-coordinates of the points of the object silhouette be x_{min} and x_{max}, respectively, and the minimum and maximum y-coordinates of be y_{min} and y_{max}, respectively. It is obvious that the area of such a rectangle is given by $(x_{max} - x_{min}) * (y_{max} - y_{min})$.

Obviously, if the object is not aligned to the axes, one can rotate the object or the axes to achieve that. Let's define a minimum rectangle over a silhouette as an axis aligned rectangle with a minimum area fully enclosing the silhouette. The assumption here is that

the object of interest in real world view can be well characterized by a rectangle, which holds well when considering the bottom or chassis of the vehicle.

. Since the vehicle could be in any random orientation, one would need to find the minimum rectangle over the silhouette with the least area over the angles of rotation of the silhouette. Figure 9 and Figure 10 below demonstrate this approach. The area of the circumscribed rectangle provides a good approximation for the length and breadth of the object when its area is minimized over the angles of rotation of the silhouette.

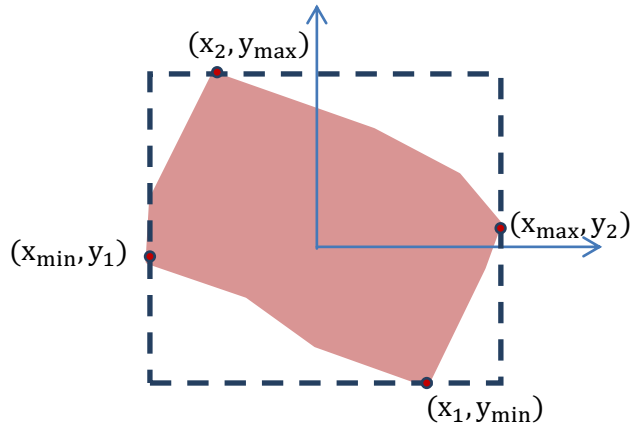

Figure 9: A minimum rectangle containing a silhouette. The length and breadth of the rectangle do not match the length and breadth of the object represented by silhouette because of a mismatch of the angle of rotation.

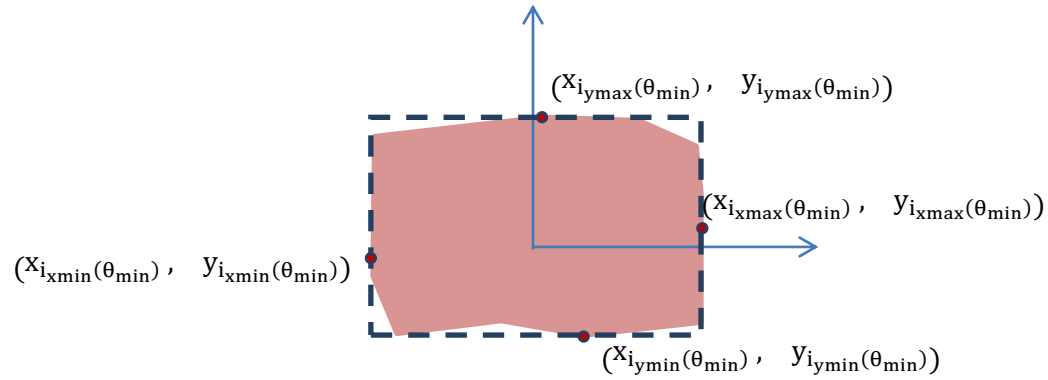

Figure 10: A minimum rectangle containing the silhouette shown in Figure 9 after the silhouette is rotated by the correct amount, so as to minimize the area of the rectangle.

The length and breadth of the rectangle are a good approximation to the length and breadth of the object represented by the silhouette after the rotation. The definitions of marked points are presented in the next subsection. We shall use this approach to estimate the dimensions of the vehicle in the world view (two dimensional Cartesian coordinates). Estimating height of the object involves additional work that will be discussed.

2.2.1 Orientation of Object Rectangle through Angular Search

The algorithm we propose to find the least area rectangle that encloses a silhouette is by minimizing the following area cost function.

$$A(\theta) = \{x_{i_{xmax}(\theta)} - x_{i_{xmin}(\theta)}\}\{y_{i_{ymax}(\theta)} - y_{i_{ymin}(\theta)}\} \tag{3}$$

with respect to θ such that,

$$i_{xmax}(\theta) = \arg\max_{i} x_i(\theta) \tag{4}$$

$$i_{xmin}(\theta) = \arg\min_{i} x_i(\theta) \tag{5}$$

$$i_{ymax}(\theta) = \arg\max_{i} y_i(\theta) \tag{6}$$

$$i_{ymin}(\theta) = \arg\min_{i} y_i(\theta) \tag{7}$$

$$\begin{bmatrix} x_i(\theta) \\ y_i(\theta) \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \hat{x}_i \\ \hat{y}_i \end{bmatrix} \tag{8}$$

The matrix, $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$, above is a standard rotation matrix that rotates the i^{th} pixel, $\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \end{bmatrix}$, of the silhouette by an angle of θ.

Let θ that minimizes the cost function in (3) be θ_{min} such that,

$$\theta_{min} = \arg\min_{\theta} A(\theta) \tag{9}$$

Once θ_{min} is found, the rectangle (with the least area) that represents the silhouette is given by the rectangle that passes through the following four points

$$\begin{aligned} &(x_{i_{xmax}(\theta_{min})}, \quad y_{i_{xmax}(\theta_{min})}), (x_{i_{xmin}(\theta_{min})}, \quad y_{i_{xmin}(\theta_{min})}), \\ &(x_{i_{ymax}(\theta_{min})}, \quad y_{i_{ymax}(\theta_{min})}), \text{ and } (x_{i_{ymin}(\theta_{min})}, \quad y_{i_{ymin}(\theta_{min})}) \end{aligned} \tag{10}$$

Example of these points forming the rectangle is seen in Figure 10.

Though a closed-form analytical solution is not available, θ_{min} can be found through search over the interval, $[-\frac{\pi}{4}, \frac{\pi}{4}]$. An appropriate scan step size of $\Delta\theta$ can be chosen, such that $A(\theta)$ is evaluated with $\theta \in \{-\frac{\pi}{4} + k\,\Delta\theta : k = 0, 1, 2, 3, \dots, N_\theta \text{ and } N_\theta = \left\lceil \frac{\pi/2}{\Delta\theta} \right\rceil\}$. So

the computational complexity of this algorithm will be O ($N_\theta \cdot N_p$), where N_p is the number of pixel points in the silhouette. Search can be sped up by choosing a large step size in the first run and then using a finer step size in the second scan in the step interval around the optimum θ found in the first step.

2.2.2 Experimental Result

The method above to identify the orientation of a silhouette was tested with the vehicle silhouettes obtained from traffic camera video frames used in [2]. The result is shown in the Figure 11 below as an example. It shows three silhouettes (grey, white, and overlapping light grey region) and corresponding orientation estimated using angular search method described earlier.

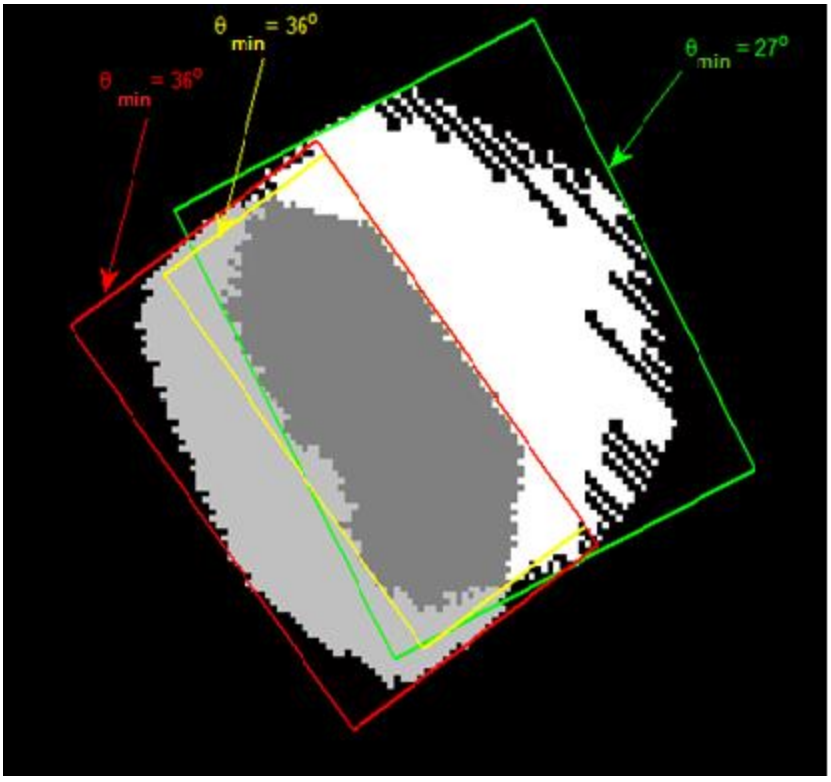

Figure 11: Rectangles containing light-gray, white and common dark gray silhouettes with correctly estimated orientations (this comes from overlapping the world views from two cameras for an example vehicle).

Chapter 3 3D Vehicle Dimension Estimation and Vehicle Tracking

In this chapter, we discuss how to process a vehicle silhouette representation in world-view in order to obtain the basic 3D geometry of the vehicle. This includes estimates of length, width and height. We first transform the derived vehicle silhouettes from the image coordinate to the real world coordinate and this is done for the silhouettes from all cameras. When considering only 2D world coordinates, the overlap of the transformed vehicle silhouettes in the real world would allow one to estimate the vehicle length and width from the oriented rectangle discussed in last Chapter. Subsequently, using the oriented rectangles and considering 3D world coordinates, one can estimate the vehicle height through 2D blob analysis and 3D space analysis assuming that the vehicle can be roughly characterized by a 3D polyhedral.

3.1 Camera Geometric Manipulation for Object Height

To allow 3D dimension estimation of the vehicle in a camera-based vision system, camera calibration is necessary that establish the projection relationship between the 3D real world objects to its appearance in the 2D image. In this thesis, it should be noted that camera calibration of multiple cameras is beyond the topic and the multiple cameras has been assumed already calibrated using the method proposed in [20], which uses the same multiple cameras and processes the same images used in this thesis. In the following, we only provide some brief background information on the projection between the real world p9and the image, which are needed later to estimate 3D dimension of the vehicles.

Figure 12 below show the typical setup of a roadside camera from both side view (upper-left) and top view (upper-right), (w_x, w_y, w_z) denote the unit vector of the world coordinate, (c_x, c_y, c_z) the unit vector of the camera coordinate, (i_x, i_y, i_z) the unit vector of the image coordinate, ϕ the tilt angle, θ the pan angle, and h the camera height above the ground plane of the roadway. The figure also shows that parallel traffic lane lines with a width w (upper-right in the figure) is typically projected to unparalleled lines in the image that converges to a so-called vanishing point when extended. The projection relationship between 3D world coordinates (w_x, w_y, w_z) and 2D image coordinate (i_x, i_y) is sought below in order to allow 3D vehicle tracking later.

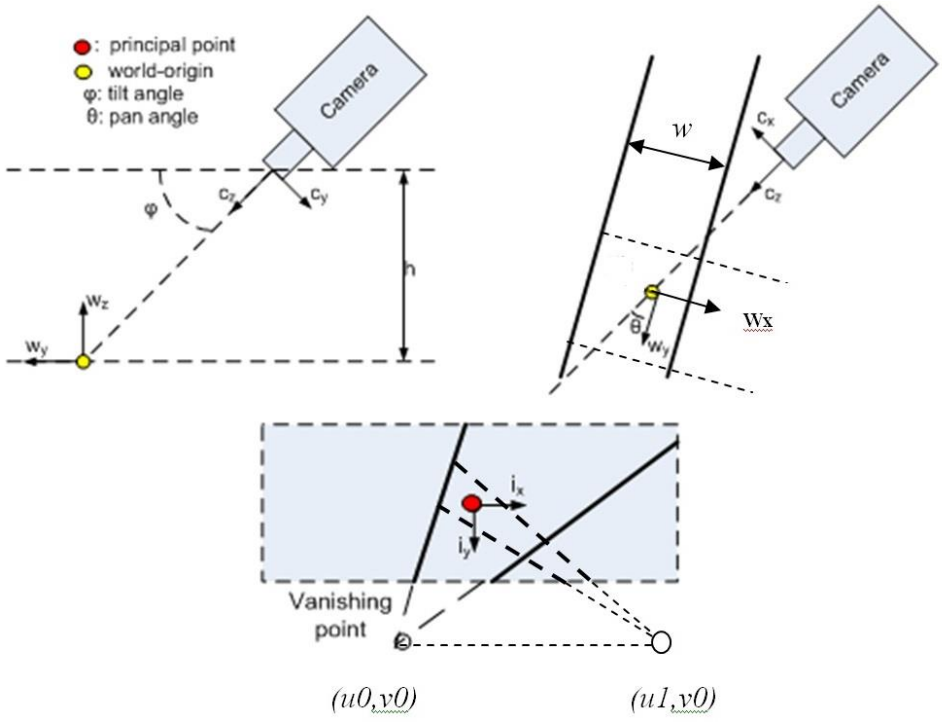

Figure 12: Typical setup of a roadside camera with Intrinsic Parameters and extrinsic parameters to project the world coordinates to the image coordinate [20].

The intrinsic parameters of a traffic camera are formulated as follows [21] [22]:

$$K = \begin{bmatrix} f & s & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{11}$$

where parameter f represents the focal length of the camera in units of pixels. (p_x, p_y) is the principle point in the image coordinate and following [23], is assumed to be (0,0), incurring very little error [20]. The skew factor s which represents the deviation of the angle between horizontal and vertical axis from 90 degrees is also set to 0 since CCD cameras are assumed to have perpendicular x and y coordinate axis [21].

In order to represent the orientation of the camera coordinate with respect to the world coordinate, a rotation matrix for an angle of rotation of ϕ be written as

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \sin\phi & \cos\phi \\ 0 & -\cos\phi & \sin\phi \end{bmatrix} \tag{12}$$

Then, the perspective transformation from the world coordinate system to the image planar coordinate system can be represented as

$$\begin{bmatrix} u \\ v \\ g \end{bmatrix} = K[R, t] \begin{bmatrix} w_x \\ w_y \\ w_z \\ 1 \end{bmatrix} \tag{13}$$

and

$$\begin{bmatrix} i_x \\ i_y \end{bmatrix} = \frac{1}{g} \begin{bmatrix} u \\ v \end{bmatrix} \tag{14}$$

where $\begin{bmatrix} u \\ v \\ g \end{bmatrix}$ is the homogenous vector and $t = -RC$ is the transformation vector with the optical center in the world coordinate C defined as

$$C = \begin{bmatrix} w_{cx0} \\ w_{cy0} \\ w_{cz0} \end{bmatrix} = \begin{bmatrix} 0 \\ -\frac{h}{\tan\phi} \\ h \end{bmatrix} \tag{15}$$

(13) above can be rewritten by substituting variables defined above

$$\begin{bmatrix} u \\ v \\ g \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f\sin\phi & f\cos\phi & 0 \\ 0 & -\cos\phi & \sin\phi & -h/\sin\phi \end{bmatrix} \begin{bmatrix} w_x \\ w_y \\ w_z \\ 1 \end{bmatrix} \tag{16}$$

In many texts, the object height w_z is set to 0 since they are interested in ground level planar view and also in simplifying the equations [20] [22] but here we intend to find the height of the object and will keep w_z as a variable in the expression. From (14) and (16), using matrix manipulation, the image planar coordinates can be found as

$$i_x = \frac{fw_x}{g} \tag{17}$$

and

$$i_y = \frac{fw_y \sin\phi + fw_z \cos\phi}{g} \tag{18}$$

where

$$g = -w_y \cos\phi + w_z \sin\phi - h\csc\phi \tag{19}$$

Note the image planar coordinates are dependent on the height of the object. When w_z is set to 0, the projection on the image planar coordinates i_x and i_y correspond to the base of the object. Once the axis aligned rectangle with the least area is fitted to the common silhouette in the world view, each vertex (as represented by w_y and w_x) of the rectangle

can be projected back to the image view using the above equations, and this creates the corresponding quadrilateral in the image view corresponding to the base of the object . We will elaborate more on this and exploit this relationship to estimate the height of the vehicle in the next section.

In order to get the projection of the ground plane in the world coordinates, w_z is set to zero, resulting in the simplification of the equations above.

$$i_y = \frac{f w_x \sin\phi}{g} \tag{20}$$

$$g = -w_y \cos\phi - h \csc\phi \tag{21}$$

Solving for w_x and w_y using equations (17), (20), and (21),

$$w_x = L \frac{-h\, i_x}{f \sin\phi + i_y \cos\phi} \tag{22}$$

$$w_y = L \frac{-h\, i_y}{\sin\phi\,(f \sin\phi + i_y \cos\phi)} \tag{23}$$

Here, the scaling factor L has been added to improve the resolution of the world view coordinates w_x and w_y since in practice the fractional component of w_x and w_y will be lost when they are used as pixel indices in the world view image (as they can only be integers). For all subsequent theoretical analysis we set $L = 1$ for simplicity, but in practice, it should be set to values several times greater than 1 to improve accuracy of calculations done in world-view. One should also remember to normalize the parameters estimated in world view like height, width, speed, etc. with the same parameter after the results have been computed to compensate for the effect of L in the final results.

3.1.1 Using Multiple Cameras to Obtain Length and Breadth of Object

Inverse projection equations (22) and (23) can be used to obtain the world view coordinates of the object silhouette at the ground plane ($w_z = 0$). This can also be done with more than two cameras. Here we discuss the case for two cameras, camera 1 and camera 2. In the world view, inverse projections of the silhouettes of the same object that appears in both two cameras should overlap at the ground plane, assuming the cameras are properly calibrated and overlapped portions of the inverse projected silhouettes provide the true bottom base representation of the object (at height 0). For instance, if the cameras are in the opposite direction, possible shadows of the object will not overlap between the two and will be selected out. This provides some amount of robustness towards the vehicle geometry estimation.

Figure 13 below shows the silhouette extracted from the image shown in Figure 4 (camera 1). Figure 14 shows an image obtained from a second traffic camera (camera 2) of the same vehicle as shown in Figure 4 (camera 1), with the background removed through the same process discussed above, and similarly Figure 15 shows the silhouette corresponding to the image from camera 2 (Figure 14) . The result of using the method of overlap of two camera views in world coordinates to estimate the 2-D dimensions (length and breadth) is shown in Figure 16. The three rectangles enclosing the silhouettes from camera 1, camera 2 and the overlapping region are shown. The latter gives the estimate of the vehicle length and breadth.

Figure 13: Silhouette extracted of vehicle from Figure 4 (camera 1).

Figure 14: Resulting image with mode based background removed using a frame of camera 2.

Figure 15: Silhouette extracted of vehicle from Figure 14 (camera 2).

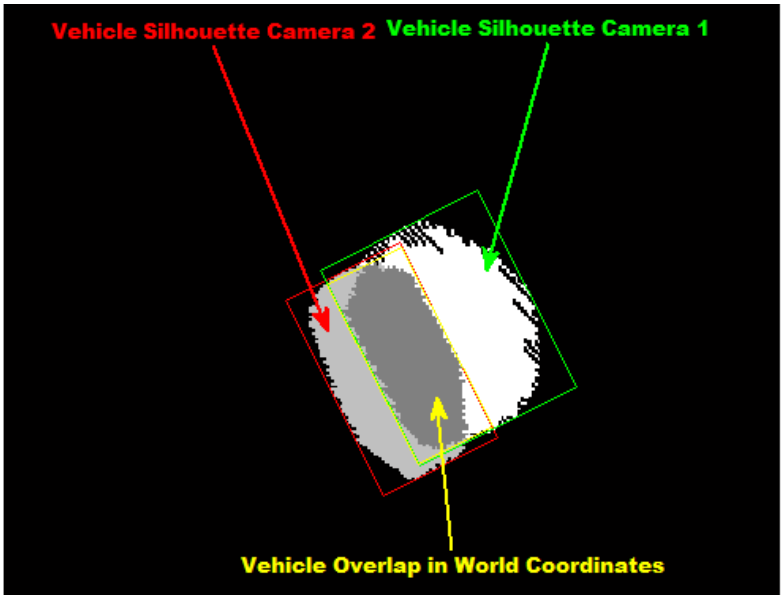

Figure 16: Overlap of the silhouettes of the same vehicle in the world view from the two cameras.

3.2 Height of Object by Fitting Rectangle in Real View

In this section, we discuss the methods to estimate the height of the vehicle from the result of applying the techniques discussed earlier to obtain the estimation of the base dimensions of the vehicle (length and breadth). Let us start with a rectangular silhouette

obtained by inverse projection from the image view, represented by the red shaded rectangle below.

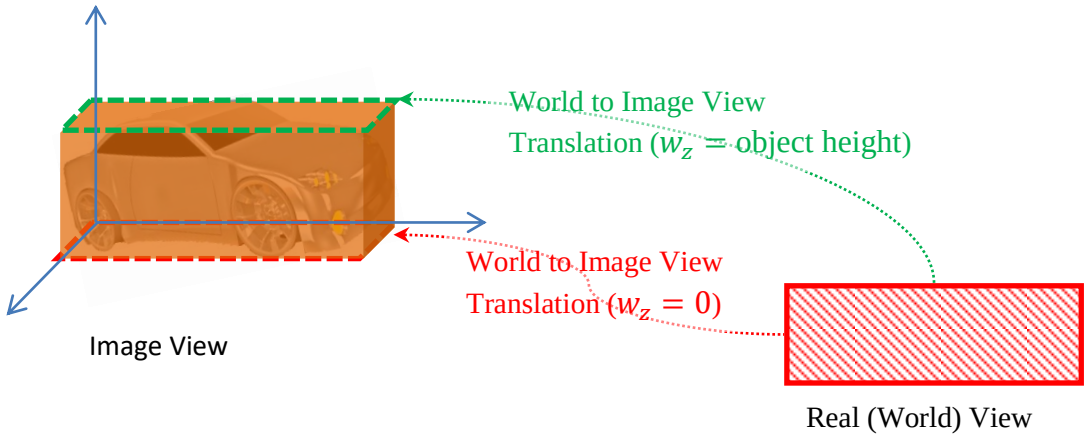

Figure 17: Graphical representation of estimating height by fitting a hyper rectangle to an object.

Using (17) and (18) and setting $w_z = 0$, one can obtain the base of the object in image view. Now the height of the object is the value of w_z that would create the top of the hyper rectangle that contains the object in image view as shown in the figure above. Note that the condition required to fit the top of the hyper rectangle is for the rectangle from the world view projected back to image view needs to have the edge furthest away to just touch the object at the edge of the object furthest away. By extension, when using a silhouette that represents the object (since using silhouettes lowers computation and space complexities), the top rectangle needs to just touch the top of the silhouette. Figure below shows an example using a silhouette of the vehicle above. The furtherness is then taken as the height. In the following, we present two possible algorithms to estimate the height.

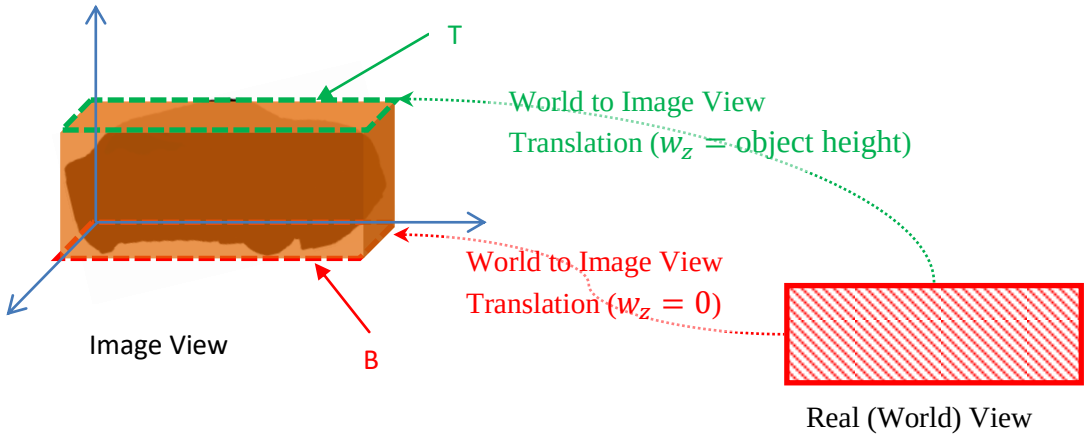

Figure 18: Graphical representation of estimating height by fitting a hyper rectangle to a silhouette representing a vehicle.

3.2.1 Algorithm 1: Iterative Projection

Once the world view coordinates of the rectangle containing the object are calculated, either using the method proposed with two cameras, or using any other method, equations (17), (18), and (19) are used iteratively by sweeping $w_z \in (0, H]$ through a range of values, less than some limit, H, until all of the object pixels in the image view are below the top most quadrilateral (shown in green in Figure 18). In order to do so, one needs to check whether each pixel lies below the top line of the top quadrilateral. This is possible by using 2D geometry to detect the direction of a point from a line. This is described next.

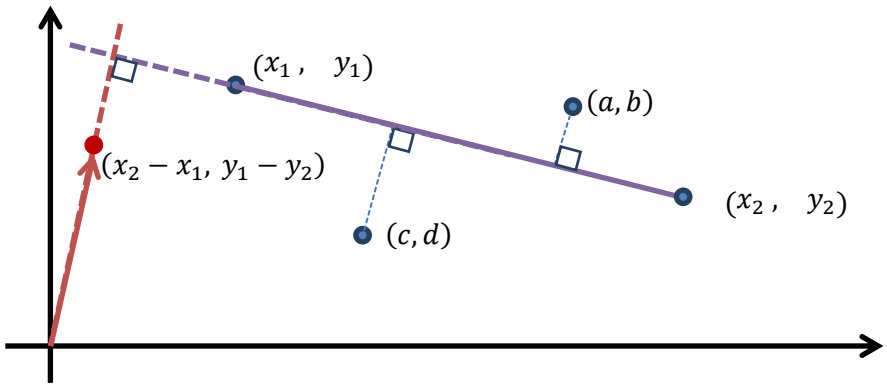

Figure 19: Estimating the direction and distance of a point from a line.

The equation of a line joining two points $(x_1\ ,\ \ y_1)$ and $(x_2\ ,\ \ y_2)$ is given by

$$(y_1 - y_2)x + (x_2 - x_1)y + x_1y_2 - x_2y_1 = 0 \qquad (24)$$

It can be shown that the vector $((y_1 - y_2), (x_2 - x_1))$ is perpendicular to the line [24]. In order to find the distance and direction of a point (a, b) from the line, one needs to find the projection of the vector $(a - x_1, b - y_1)$ on the perpendicular vector$((y_1 - y_2), (x_2 - x_1))$, which is obtained as the following [24].

$$\frac{(a - x_1)(y_1 - y_2) + (b - y_1)(x_2 - x_1)}{\sqrt{(x_2 - x_1)^2 + (y_1 - y_2)^2}} \qquad (25)$$

The magnitude of expression (25) gives the distance of the point from the line, whereas the sign gives the direction of the point from the line. In order to compare the relative distances of various points from the line (as shown later) and identify their directions, we can drop the denominator, since it is the same for all of the points and is always positive. Then, the following can be derived from (25),

$$D = (a - x_1)(y_1 - y_2) + (b - y_1)(x_2 - x_1). \qquad (26)$$

Thus, if

$$(a - x_1)(y_1 - y_2) + (b - y_1)(x_2 - x_1) > 0, \quad (27)$$

then the point lies above the line, if

$$(a - x_1)(y_1 - y_2) + (b - y_1)(x_2 - x_1) < 0, \quad (28)$$

then the point lies below the line, and if

$$(a - x_1)(y_1 - y_2) + (b - y_1)(x_2 - x_1) = 0, \quad (29)$$

then the point lies on the line.

The draw-back of the above method is that one requires computing equations (17), (18), and (19) repeatedly until a solution is found. With some approximation, assuming that the height of the camera is significantly greater than the height of objects/vehicles, one can come up with a closed-form solution that will provide an estimate of the height of the vehicle in a single step. So, for camera that are placed at high above the ground, we propose the following more efficient method.

3.2.2 Algorithm 2: Closed Form Computation

The algorithm above can be considered a direct algorithm. A close look at the algorithm above shows height of the object is estimated as w_z satisfying equations (17), (18) and (19), and the back projection of the real-view rectangle in ground plane into image view creating the top quadrilateral such that all of the points of the silhouette are below its top line. We will now attempt to derive the algorithm to estimate the height of the object based on this understanding. This includes first obtaining the bottom quadrilateral B as shown in Figure 18. Given that the height of the camera is much larger than the height of the object, the top side of the top quadrilateral T can be taken to be parallel to the top side

of the bottom quadrilateral B. Thus one needs to find the top quadrilateral T such that its top side passes through the point in silhouette furthest away from the top line of the bottom quadrilateral B. This requires one to find the point in silhouette that is furthest away from the top line of the quadrilateral B. For this, we will use equation (26), we will find the point, say P_{max} that maximizes D with respect to the top line of the quadrilateral B. Once this point is found, one can derive a relationship between w_z and this point, P_{max}.

Let P_{max} be represented by the image coordinates (i_x, i_y), and the two corners of the top line of the upper quadrilateral T as $\left(i_{x_1}, i_{y_1}\right)$ and $\left(i_{x_2}, i_{y_2}\right)$. Correspondingly, let those two points be translated from $\left(w_{x_1}, w_{y_1}\right)$ and $\left(w_{x_2}, w_{y_2}\right)$ in the world view. This is represented Figure 20.

For (i_x, i_y) to pass through the line joining $\left(i_{x_1}, i_{y_1}\right)$ and $\left(i_{x_2}, i_{y_2}\right)$, (i_x, i_y) divides the line into two segments with the same slope, i.e. ,

$$\frac{i_{y_1} - i_y}{i_{x_1} - i_x} = \frac{i_{y_2} - i_y}{i_{x_2} - i_x}. \tag{30}$$

The two points $\left(i_{x_1}, i_{y_1}\right)$ and $\left(i_{x_2}, i_{y_2}\right)$ can be obtained from the corresponding points $\left(w_{x_1}, w_{y_1}\right)$ and $\left(w_{x_2}, w_{y_2}\right)$ in the world view respectively, using the equations (17), (18), and (19), and combined with equation (30) resulting in

$$\frac{fw_{y_1}\sin\phi + fw_z\cos\phi - i_y(-w_{y_1}\cos\phi + w_z\sin\phi - h\csc\phi)}{fw_{x_1} - i_x(-w_{y_1}\cos\phi + w_z\sin\phi - h\csc\phi)} = \frac{fw_{y_2}\sin\phi + fw_z\cos\phi - i_y(-w_{y_2}\cos\phi + w_z\sin\phi - h\csc\phi)}{fw_{x_2} - i_x(-w_{y_2}\cos\phi + w_z\sin\phi - h\csc\phi)}. \quad (31)$$

Note that the height parameter w_z remains the same for translating each of these points from the world view to the image view.

For ease of representing these long expressions, let's define the following,

$$A_k = fw_{y_k}\sin\phi \quad (32)$$

$$B_k = i_y w_{y_k}\cos\phi \quad (33)$$

$$C = i_y h\csc\phi \quad (34)$$

$$D_k = fw_{x_k} \quad (35)$$

$$E_k = i_x w_{y_k}\cos\phi \quad (36)$$

$$F = i_x h\csc\phi \quad (37)$$

$$G = f\cos\phi - i_y\sin\phi \quad (38)$$

$$H = i_x\sin\phi \quad (39)$$

With these new variables, the following is derived by solving equation (31),

$$\frac{A_1 + Gw_z + B_1 + C}{D_1 + E_1 - Hw_z + F} = \frac{A_2 + Gw_z + B_2 + C}{D_2 + E_2 - Hw_z + F}. \quad (40)$$

Solving for w_z above gives height of the object as,

$$w_z = \frac{(A_2 + B_2 + C)(D_1 + E_1 + F) - (A_1 + B_1 + C)(D_2 + E_2 + F)}{G(D_2 - D_1 + E_2 - E_1) + H(A_2 - A_1 + B_2 - B_1)}. \quad (41)$$

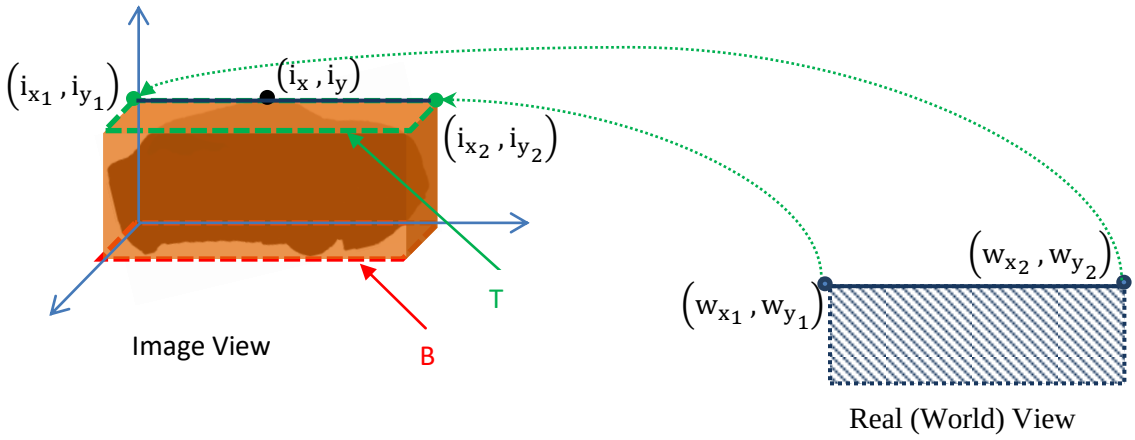

Figure 20: Estimating the upper rectangle in the image view to estimate the height of the object represented by the silhouette.

3.3 Tracking Vehicle in Three Dimensions

The results presented earlier also provide for the input parameters for vehicle tracking in real world. Since the vehicle can be accurately represented in the world view, with its dimensions, position, and height extracted, this information can also be used for tracking the vehicle in three dimensions by simply assuming a 3D cubic model for the vehicle. Note that in literature work, more refined 3D models with different levels of granularity have been used [7] [9]. In terms of tracking of vehicle as they move across the camera view, the Kalman [25] filter is very useful because it can efficiently combine noisy observations assumed to be Gaussian distributions with a predefined model of vehicle movement [11] [9] [5]. There are other methods like using Monte Carlo sampling that have also been proposed [8]. One of the suggested benefits of the Monte Carlo method is that it is not restricted to the Gaussian distribution of data, or assumptions of linearity, which is the case for the Kalman filter. However, most publications show adequate

performance of the Kalman filter [2] [4] [9] so we have also used it here for 3D tracking of the vehicle with multiple cameras.

3.3.1 Description of the Kalman Filter

The Kalman filter is a very well established technique for linear estimation and tracking used in a varieties of fields including aircraft navigation, object tracking, robotic control etc., because of its effectiveness for on-line estimation and efficient implementation on a digital computer [26] [27]. Here we quickly summarize the state space implementation of the Kalman filter as a solution to the problem of estimating the state of a discrete-time linear stochastic process that can be described by the following equation [28] [29].

$$x_t = F_t x_{t-1} + B_t u_t + w_t \tag{42}$$

- x_t is the state vector containing the terms of interest for the system (e.g., position, velocity) at time t
- u_t is the vector containing any control inputs (e.g., acceleration)
- F_t is the state transition matrix providing the relation of state at time t with that at time $t-1$ (e.g., the position and velocity a time $t-1$ effecting position at time t)
- B_t is the control input matrix which relates the control input to the state vector at time t
- w_t is the vector containing the process noise for each element in the state vector. The process noise is assumed to have zero mean with covariance given by the covariance matrix Q_t

Additionally, the measurements of the system are assumed to follow the following model.

$$y_t = H_t x_t + v_t \tag{43}$$

- y_t is the vector of measurements
- H_t is transformation matrix relating the state vector to the measurement
- v_t is the vector of measurement noise, assumed to be zero mean Gaussian white with covariance matrix R_t

The solution with the assumptions above can be stated in terms of predict-update equations below.

Predict:

$$\hat{x}_{t|t-1} = F_t \hat{x}_{t-1|t-1} + B_t u_t \tag{44}$$

$$P_{t|t-1} = F_t P_{t-1|t-1} F_t^T + Q_t \tag{45}$$

Update:

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(y_t - H_t \hat{x}_{t|t-1}) \tag{46}$$

$$K_t = P_{t|t-1} H_t^T \left(H_t P_{t|t-1} H_t^T + R_t\right)^{-1} \tag{47}$$

$$P_{t|t} = (I - K_t H_t) P_{t|t-1} \tag{48}$$

$\hat{x}_{t|t-1}$ represents the prediction of the state for time t based on the updated state at time $t-1$. Likewise, $P_{t|t-1}$ is the state variance matrix predicted for time t based on the updated state variance at time $t-1$. Predicted state $\hat{x}_{t|t-1}$ and predicted covariance matrix $P_{t|t-1}$ are updated with the new observation y_t obtained at time t resulting in $\hat{x}_{t|t}$ and $P_{t|t}$ respectively.

3.3.2 Kalman Filter for Vehicle Tracking

Tracking a moving vehicle perturbed by zero mean random acceleration assumes each component of the vehicle position is independently measured in a Cartesian system in the world view. The problem is to obtain the optimum estimates of position and velocity of the vehicle. For our tracking the state vector consists of the three Cartesian coordinates of the centroid of the 3D vehicle Cubic representing the vehicle in world view with length, width and height, as obtained through methods explained earlier. The state vector also consists of the 3D velocity components of the vehicle model. Based on Friedland's model [30] of tracking a moving object and applying it to a 3D space, the state space representation of the moving vehicle can be done in the following manner. Let us assume that the world view position (p) of the vehicle be represented as $p = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$, where p_x and p_y are the Cartesian coordinates representing the position in the world view (as given by the minimum rectangle method described before) and p_z is the height estimated through method proposed earlier. In addition, obtaining the velocity in each direction as the derivative of the individual components, we can define the state variable as follows.

$$x_t = \begin{bmatrix} p_x(t) \\ p_y(t) \\ p_z(t) \\ \dot{p}_x(t) \\ \dot{p_y}(t) \\ \dot{p_z}(t) \end{bmatrix} \tag{49}$$

Where $\dot{p}_x$, $\dot{p_y}$, and $\dot{p_z}$ represent the velocity components in the respective dimensions. Given that the sampling rate (frame rate) is much higher that velocity of the vehicle, one can assume that it remains constant between successive samples (frames) and can thus be

approximated with the difference between the values at consecutive samples or frames. That is,

$$\dot{p}_w(t) \cong p_w(t) - p_w(t-1);\ w \in \{x, y, z\} \tag{50}$$

In addition, one can also derive the following [30],

$$H_t = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{51}$$

$$F_t = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{52}$$

For some applications, F_t and H_t might change with each time step, but here it is assumed to be constant.

With real world problems, the process noise covariance Q_t and the measurement noise covariance R_t can be time varying. But we assume that these noise components are wide-sense stationary (WSS), making these matrices constant [31], since the noise properties are assumed to not change during the short duration of the camera coverage. We assume the noise between samples to be uncorrelated and thus assign them to be a diagonal matrix of the form,

$$Q_t = \sigma_w^2 I \tag{53}$$

$$R_t = \sigma_v^2 I \quad (54)$$

Here I is the 6×6 identity matrix and the parameters σ_w^2 and σ_v^2 represent the variances of process noise and measurement noise respectively and can be usually chosen heuristically with trial and error, with the understanding that a relatively high σ_w^2 results in more reliance on measurement and a relatively high σ_v^2 results in more reliance on the model by the Kalman filter for the state estimation [11].

Additionally, in order to track multiple vehicles at a time and to avoid erroneous association between vehicles and objects, we have adopted the method of 'strong match' and 'loose match' based on percentage of silhouette overlap as proposed in [2], but applied to the silhouette extracted from two cameras in world view as proposed in the earlier sections. The method involves breaking all loose matches if an object and a vehicle has a strong match and all multiple-object to multiple-vehicle associations are broken with the lowest overlap percentage. This simplifies the tracking process into associations restricted to either one-to-one, multiple-to-one or one-to-multiple cases.

Chapter 4 Data Management and Results

The 3D vehicle tracking system has been implemented as software with MATLAB® [19]. The system has been run on a Windows® 10 PC with a 1.6 GHz Intel® Core™ i7-720QM (released 2009) processor. The speed of our implementation with this setup has been measured at 1.04 sec per frame. With more recent processors (and graphical processing units) and multi-core code architecture, the speed is expected to be much faster. For future work, there is also an opportunity to further optimize the Matlab® scripts for speed (e.g., use parallel processing for multi-core architecture) or implement in lower level programming languages like C, which will also boost the speed.

The system logs the following statistics for each identified vehicle at each frame.

Table 2: List of recorded vehicle attributes

Vehicle Tracking Attribute	**Description**
start_Frame	Frame index where tracking started
end_Frame	Frame index where tracking ended
Detail	Raw (original orientation) silhouette of vehicle for each frame
Upperleft	Upper left pixel location of rectangle containing raw silhouette
Width	Width of rectangle containing raw silhouette
Length	length of rectangle containing raw silhouette
State	Kalman State consisting of vehicle location and velocity
predicted_state	Predicted Kalman State

height1	Estimated height based on worldview rectangle projected to camera 1
height2	Estimated height based on worldview rectangle projected to camera 2
RectangleLength	Estimated vehicle length
RectangleWidth	Estimated vehicle width
Rect	Structure containing estimated angle, area and vertices of vehicle rectangle
rectCenter	Center of vehicle rectangle
Cam1_3dImageCenterHt1	3D centroid of vehicle based on worldview rectangle projected to camera 1
Cam2_3dImageCenterHt2	3D centroid of vehicle based on worldview rectangle projected to camera 2
kalmanH	Structure containing Kalman filter parameters
Speed	Estimated speed of vehicle

Figure below shows an example of the Matlab® structure containing these properties of the vehicle tracking.

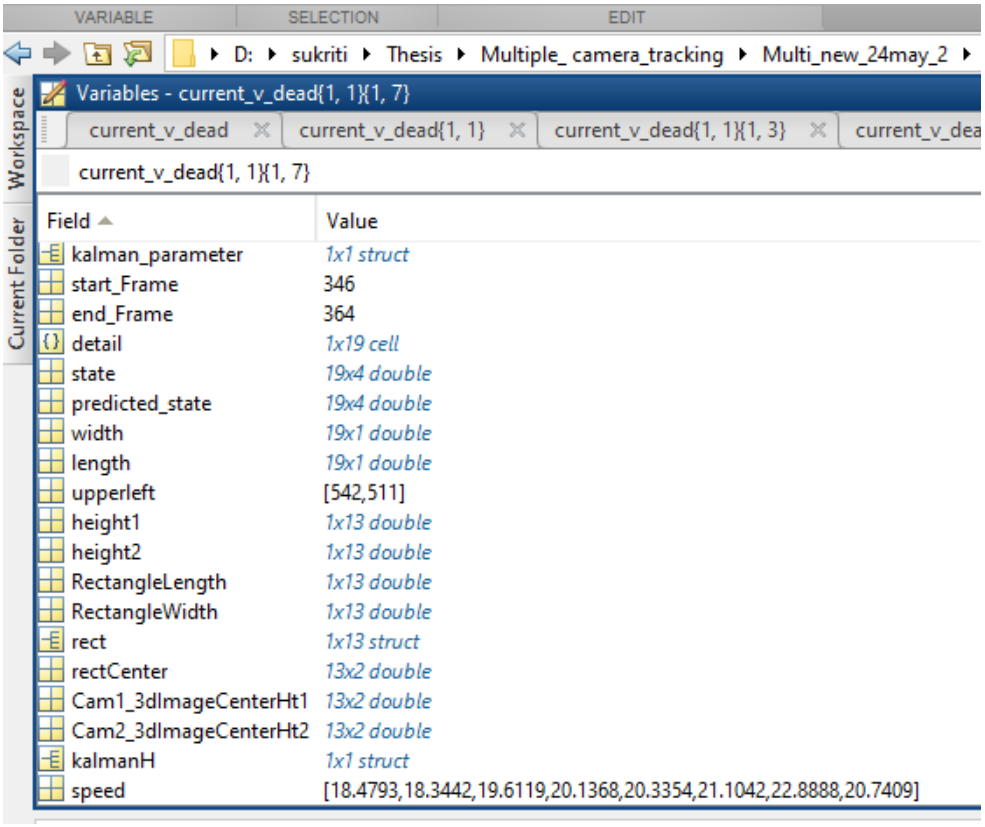

Figure 21: Screen capture showing vehicle properties recorded in Matlab® structure.

4.1 Extracting 3D Vehicle Dimensions

The proposed methods have been tested on a video recorded for a local intersection from two cameras, and camera calibration had already been done. After evaluating the two methods above to estimate vehicle height, the results from both matched almost exactly. So, given that the closed form method is computationally less expensive, we have demonstrated the results using this method in the following. Figure 22 and Figure 23 show the fit of the top and bottom planes of vehicle 1 based on the estimated dimensions in camera 1 view and camera 2 view respectively. Similarly, Figure 24 and Figure 25 show the fit of the top and bottom planes of vehicle 2 based on the estimated dimensions in camera 1 view and camera 2 view respectively.

Figure 26 shows the estimation of the various dimensions for a set of consecutive image frames for vehicle 1. And Figure 27 shows the estimation of the various dimensions for a set of consecutive frames for vehicle 2. Though we do not have the actual vehicle measurements to compare against, the obtained values are within expected ranges of average vehicles. It is noteworthy that the height obtained from Camera 1 is under-estimated and this is likely due to camera calibration mismatch and vehicle detection/segmentation error. Camera calibration itself is out of scope of this thesis and recalibrating the camera was not attempted. But the estimation of the length and breadth and height using the camera 2 are reasonable. Also, the results seem to show that the estimated vehicle heights from both cameras are converging.

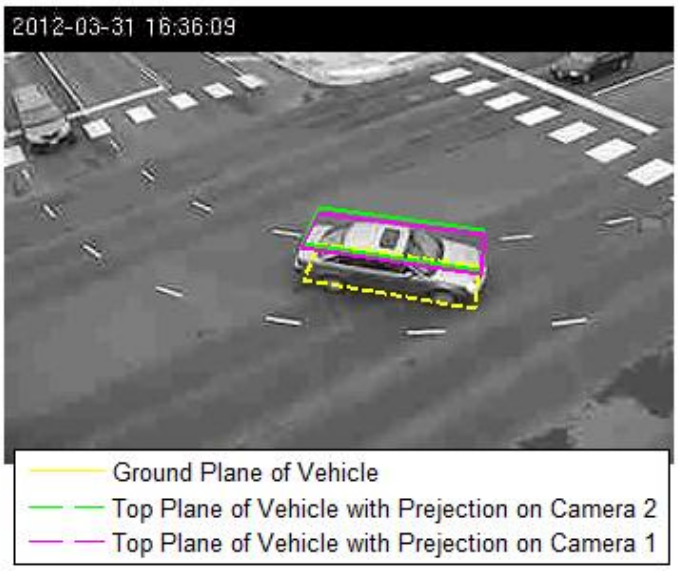

Figure 22: Top and bottom planes calculated on vehicle 1 in camera 1 view.

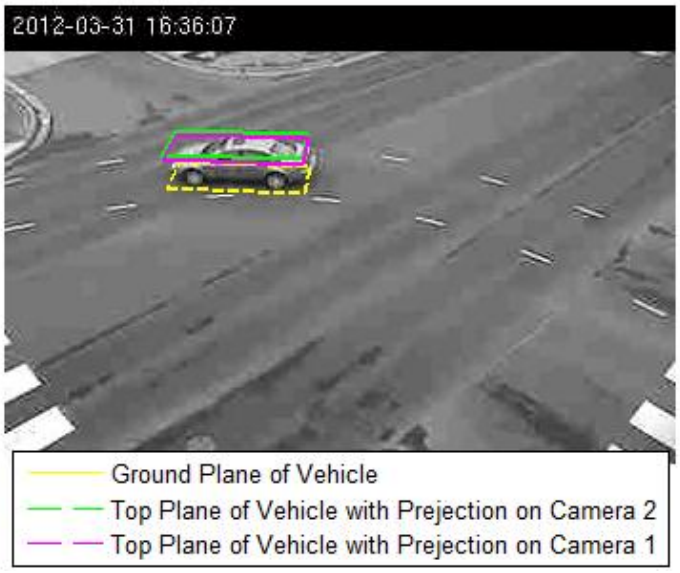

Figure 23: Top and bottom planes calculated on vehicle 1 in camera 2 view.

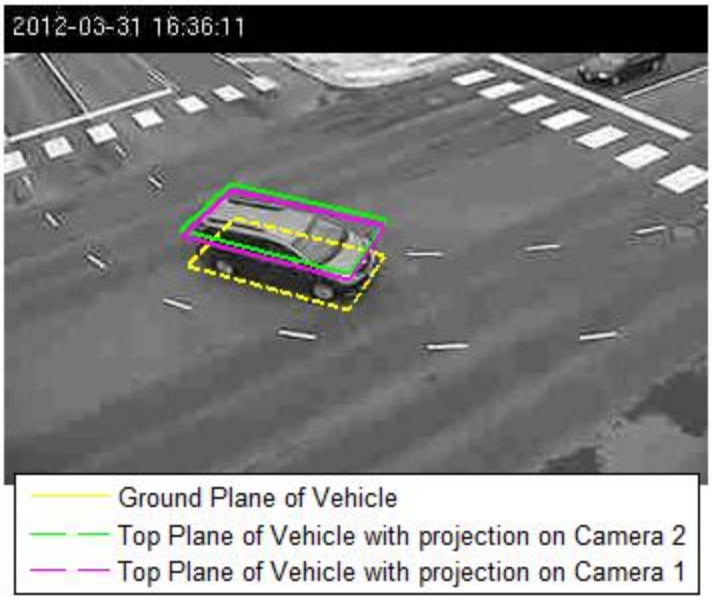

Figure 24: Top and bottom planes calculated on vehicle 2 in camera 1 view.

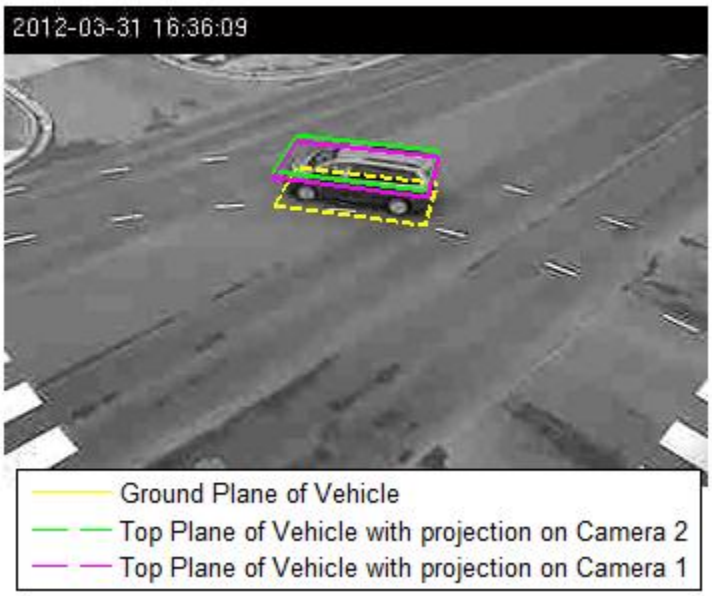

Figure 25: Top and bottom planes calculated on vehicle 2 in camera 2 view.

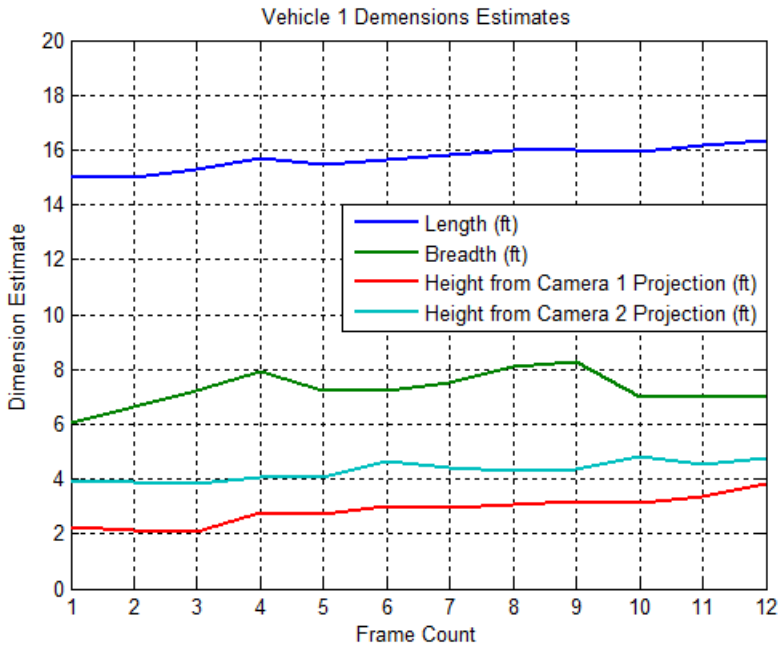

Figure 26: The estimated dimensions of vehicle 1 in for a sequence of frames in the video.

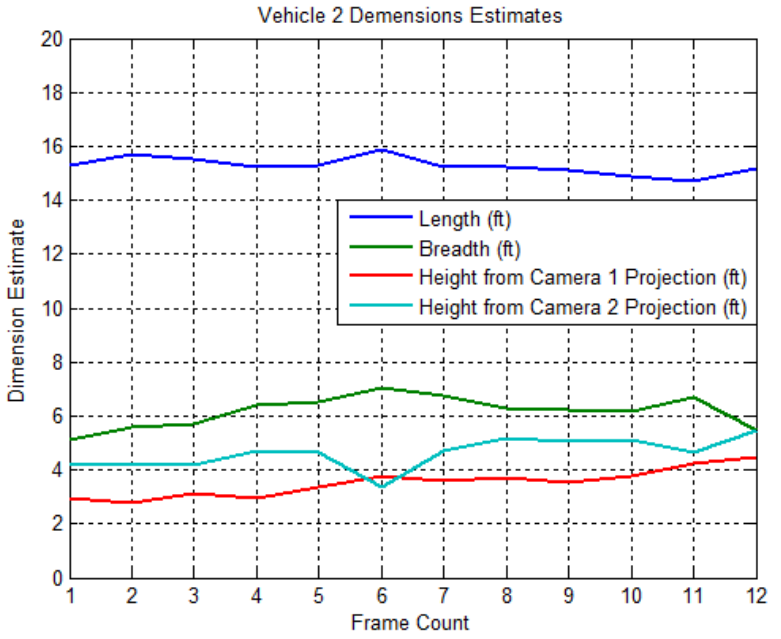

Figure 27: The estimated dimensions of vehicle 2 in for a sequence of frames in the video.

4.2 3D Vehicle Tracking

Since we are tracking the vehicles in the world view with the techniques mentioned above, we can also estimate the speed of the vehicle (along with dimensions) in real units like miles per hour. The proposed algorithms are able to reliably track the vehicles and provide the real world dimension and speed. The Matlab® script that implements these ideas also generates real-time tracking video to demonstrate the proposed methods. Figure 28 and Figure 29 show example frames of such a video with each showing one of the tracked vehicles at the traffic intersection. They also show the result of the intermediate steps of edge detection (skipped in the second video, Figure 29, for clarity), dimension estimation, real world projection with two cameras and continuous vehicle tracking.

Figure 28: Frame of a tracking video showing the intermediate steps of edge detection, dimension estimation, real world projection with two cameras and continuous vehicle tracking (video link: https://drive.google.com/open?id=0BzxBQjIl2GdZNlBSdGVsWldPYjg).

Figure 29: Frame of a tracking video showing another vehicle with the steps of dimension estimation, real world projection with two cameras and continuous vehicle tracking (video link: https://drive.google.com/open?id=0BzxBQjIl2GdZN3dfVjIzMk9kblE).

As discussed one of the main applications of vehicle tracking is generation of statistics of the vehicle attributes, for example speed and dimensions. Following figures show the overlaid trajectories of 311 tracked vehicles that had at least life of 6 frames or more over 10000 frames (16.6 minutes) as projected to the two camera views.

Figure 30: Vehicle example trajectories projected on to camera view of camera 1.

Figure 31: Vehicle example trajectories projected on to camera view of camera 2.

4.3 Advantage of Multi-Camera System

By using two cameras instead of just one, we have observed that the vehicle statistics collection has been more accurate. When looking at the vehicle detection we found numerous cases of misdetection with a single camera. When two vehicles are very close or partially hidden by one another in the video, they are observed to be detected as one vehicle by the single camera system.

Based on the vehicle count from the individual camera view detection and the proposed multi-camera vehicle tracking system, we found the following observations over 10000 frames (16.6 min) of the video. Out of the 311 of detected vehicles (by using the two-camera system) following were misdetected when using one camera only.

- Misdetections using only camera 1: 21 (7%)
- Misdetections using only camera 2: 18 (6%)

We also verified by visually inspecting the corresponding frames in the images. Figure 32 shows an example of misdetection using only camera 1 only. Since the two vehicles in

the camera 1 view are very close and they move together throughout the view of the camera, they are represented as one vehicle silhouette by the segmentation algorithm, resulting in error. This is represented by the single red rectangle in the world view. But since the camera 2 view has the vehicle more clearly separated, the segmentation and detection algorithm correctly represents them as two vehicle silhouettes. In the second example shown in Figure 33, it is in the camera 2 view that the vehicles are misidentified, and camera 1 is accurate. Hence, by combining both the camera views we get the correct vehicle detection in all of these cases. The yellow rectangles in the world view represent these correctly identified vehicles by using the combination both cameras in the figures below.

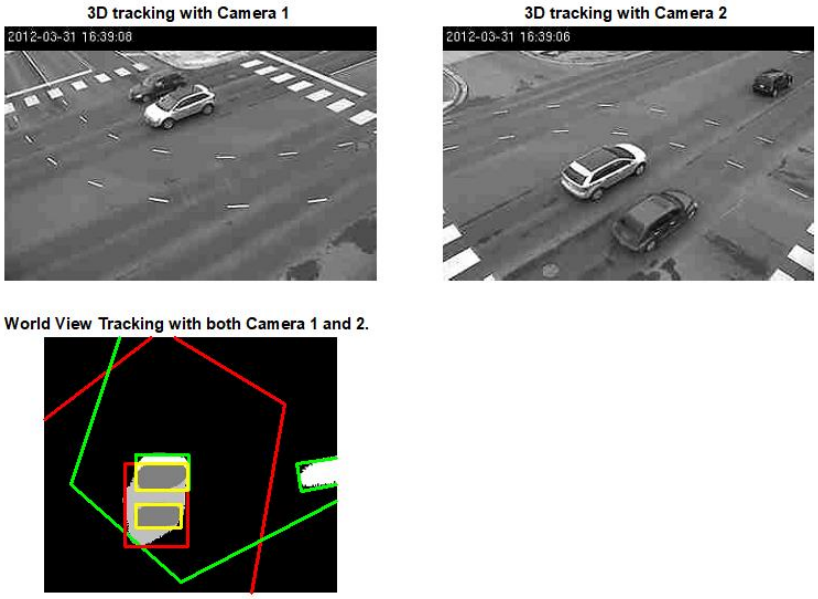

Figure 32: Demonstration of misidentification of two vehicles as one when using only camera 1 compared to using both cameras.

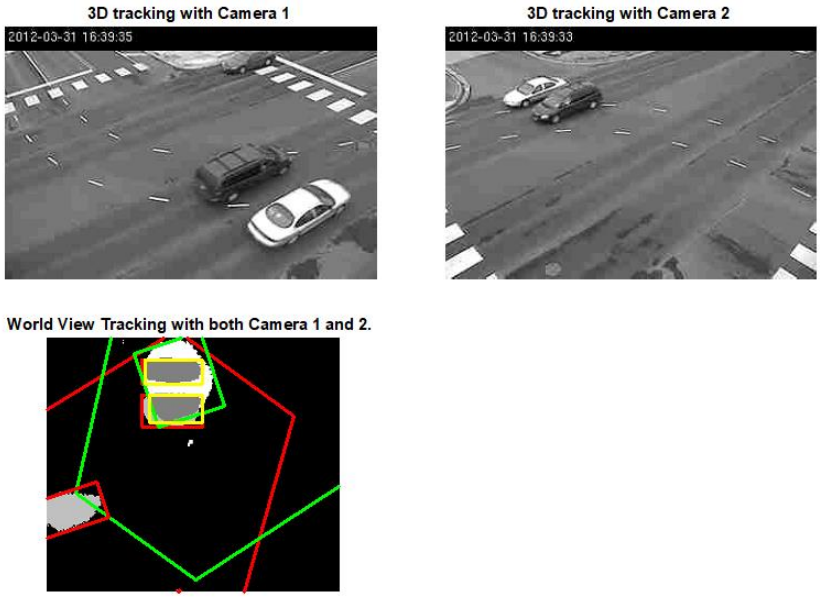

Figure 33: Demonstration of misidentification of two vehicle as one when using only camera 2 compared to using both cameras.

Chapter 5 Conclusion

In this thesis we have presented a 3D vehicle tracking system, involving 3D dimension extraction from fusion of information between two cameras. Using more than one camera improves the performance of the tracking system due to additional view evidence of the vehicle on average and especially when vehicle occlusion is present in the camera view of one camera but not in other. The work here can easily be extended to any number of cameras.

This work builds on top of the single camera tracking system developed by Hai Dinh where the tracking was done in the camera view [2]. With this work the following enhancements and capabilities have been added:

- Use of multiple cameras which improves the performance in the presence of shadow, and partially occluded vehicles.
- World view tracking of vehicle giving the vehicle attributes (dimensions and speed) in real units like feet and miles per hour.
- Estimation of 3D vehicle dimensions using fusion of information from multiple cameras and image processing techniques.

This vehicle tracking system has been implemented with Matlab® scripts and collects a database of estimated vehicle attributes like length, width, height, and speed for each of the tracked vehicles and also generates a real time video of the tracking process.

The Matlab® scripts developed during this thesis work are available through the following Google® drive folder link:

https://drive.google.com/open?id=0BzxBQjIl2GdZQjFyUGt6eWNJMTA

This folder includes the sample videos. One would need to download the folder and run *Tracking3dMain.m* in the folder named *Tracking3dScripts* after adjusting the variable *MoG_dir* to the local directory path.

There are improvements possible in various fronts for future work:

- Reduce the computational complexity of the equations to speed up the calculations.

- Improve camera calibration. In this work, we utilized the provided camera parameters, but noticed that the parameters may not be very accurate on the second camera. There seems to be opportunity to explore this further, to quantify the quality of the calibration by comparing the performance between the multiple cameras and possibly adjust the parameters of one camera calibration based on the additional cameras with known accurate parameters.

References

[1] S. Gupte, O. Masound, R. F. K. Martin and N. Papanikolopoulos, "Detection and Classification of Vehicles," *IEEE Transaction on Intelligent Transportation System,* vol. 3, no. 1, pp. 37-47, 2002.

[2] H. Dinh , "Development of a Video-Based Traffic Data Collection System," M.S. thesis, University of Minnesota Duluth, Duluth, 2011.

[3] Z. Hu, C. Wang and K. Uchimura, "3D Vehicle Extraction and Tracking from Multiple Viewpoints for Traffic Monitoring by using Probability Fusion Map," in *Proceedings of the IEEE Intelligent Transportation Systems Conference*, Seattle, 2007.

[4] Z. Qiu and D. Yao, "Kalman Filtering Used in Video-Based Traffic Monitoring System," *Journal of Intellegent Transportation Systems,* vol. 10, no. 1, pp. 15-21, 2006.

[5] S. R. E. Datondji, Y. Dupuis, P. Subirats and P. Vasseur, "A Survey of Vision-Based Traffic Monitoring of Road Intersections," *IEEE Transactions on Intelligent Transportation Systems,* vol. 17, no. 10, pp. 2681-2698, 2016.

[6] P. St-Aubin, N. Sauner and L. Miranda-Moreno, "Large-Scale Automated Proactive Road Safety Analysis using Video Data," *Transportation Research Part C,* no. 58, pp. 363-379, 2015.

[7] V. Lepetit and P. Fua, "Monocular Model-Based 3D Tracking of Rigid Objects: A Survey," *Foundations and Trends in Computer Graphics and Vision,* vol. 1, no. 1, pp. 1-89, 2005.

[8] A. Romanoni, L. Mussane, D. Rizzi and M. Matteucci, "A Comparison of Two Monte Carlo Algorithms for 3D Vehicle Trajectory Reconstruction in Roundabouts," *Pattern Recognition Letters,* no. 51, pp. 79-85, 2014.

[9] J. Lou, T. Tan, W. Hu, H. Yang and S. Maybank, "3-D Model-Based Vehicle Tracking," *IEEE Transactions on Image Processsing,* no. 14, pp. 1561-1569, 2005.

[10] C. Stauffer and W. Grimson, "Adaptive Background Mixture Models for Real-Time Tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Fort Collins, 1999.

[11] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," July 2006. [Online]. Available: http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html. [Accessed 2013].

[12] M. J. Leotta and J. L. Mundy, "Vehicle Surveillance with a Generic, Adaptive, 3D Vehicle Model," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 33, no. 7, pp. 1457-1469, 2011.

[13] Marco Cristani, Michela Farenzena, Domenico Bloisi and Vittorio Murino, "Background Subtraction for Automated Multisensor Surveillance: A Comprehensive Review," *EURASIP Journal on Advances in Signal Processing,* no. 1, 2010.

[14] M. Haag and H.-H. Nagel, "Combination of Edge Element and Optical Flow Estimates for 3D-Model-Based Vehicle Tracking in Traffic Image Sequences," *International Journal of Computer Vision ,* vol. 35, no. 3, pp. 295-319, 1999.

[15] D. C. Montgomery and G. C. Runger, "Random Sampling and Data Description," in *Applied Statistics and Probability for Engineers*, 3th ed., New York, John Wiley & Sons, Inc., 2003, pp. 189-219.

[16] L. Wang and N. Yung, "Extraction of Moving Objects From Their Background Based on Multiple Adaptive Thresholds and Boundary Evaluation," *IEEE Transaction on Intelligent Transportation System,* vol. 11, no. 1, pp. 40-51, 2010.

[17] W. K. Pratt, "Edge Detection," in *Digital Image Processing*, 4th ed., Hoboken, John Wiley & Sons, Inc., 2007, pp. 465-534.

[18] R. C. Gonzalez, R. E. Woods and S. L. Eddins, "Point, Line, and Edge Detection," in *Digital Image Processing Using Matlab*, Upper Saddle River, Pearson Prentice Hall, 2003, pp. 379-392.

[19] "Matlab Online Documentation," Mathworks, [Online]. Available: http://www.mathworks.com/help/releases/R2017b/images/index.html. [Accessed 2013].

[20] H. Tang, "Development of a Multiple-Camera Tracking System for Accurate Traffic Performance Measurements at Intersections," Final Report, Minnesota Department of Transportation, Minneapolis, 2012.

[21] R. Hartley and A. Zisserman, "Camera Geometry and Single View Geometry," in *Multiple View Geometry in Computer Vision*, 2nd ed., Cambridge, Cambridge University Press, 2003, pp. 151-194.

[22] Z. Zhang, "A Flexible New Technique for Camera Calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22, no. 11, pp. 1330-1334, 2000.

[23] K. Song and J. Tai, "Dynamic Calibration of Pan-Tilt-Zoom Cameras for Traffic," *IEEE Transaction on System, Man, and Cybernetics,* vol. 36, no. 5, pp. 1091-1103, 2006.

[24] H. Anton and C. Rorres, "Euclidean Vector Spaces," in *Elementary Linear Algebra*, 10th ed., John Wiley & Sons, Inc., 2010, pp. 119-170.

[25] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME-Journal of Basic Engineering,* vol. 82, no. Series D, pp. 35-45, 1960.

[26] R. A. Singer, "Estimating Tracking Filter Performance for Manned Maneuvering Targets,"

IEEE Transactions on Aerospace and Electronics Systems, Vols. AES-6, no. 4, pp. 437-483, 1970.

[27] K. Ramchandra, B. R. Mohan and B. R. Geetha, "A Three-State Kalman Tracker using Position and Rate Measurements," *IEEE Transactions on Aerospace and Electronics Systems,* vol. 29, no. 1, pp. 215-222, 1993.

[28] R. Faragher, "Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation," *IEEE Signal Processing Magazine,* pp. 128-132, 2012.

[29] S. O. Haykin, "Kalman Filters," in *Adaptive Filter Theory*, 4th ed., New Jersey, Prentice Hall, 2002.

[30] B. FriedLand, "Steady State Behaviour of Kalman Filter with Discrete- and Continuous-Time Observations," *IEEE Transactions on Automatic Control,* vol. 25, no. 5, pp. 988-992, 1980.

[31] B. P. Lathi, "Random Processes," in *Modern Digital and Analog Communication Systems*, 3rd ed., New York, Oxford University Press, 1998, pp. 487-531.