Development of a Multiple-Camera Tracking System for Accurate Traffic Performance Measurements at Intersections

Final Report

Prepared by:

Hua Tang

Department of Electrical and Computer Engineering
Northland Advanced Transportation Systems Research Laboratories
University of Minnesota Duluth

CTS 13-10

Technical Report Documentation Page

1. Report No.	2.	3. Recipients Accession No.	
CTS 13-10			
4. Title and Subtitle Development of a Multiple-Camera Tracking System for Accurate Traffic Performance Measurements at Intersections		5. Report Date February 2013 6.	
7. Author(s) Hua Tang		8. Performing Organization Report No.	
9. Performing Organization Name and Address Department of Electrical and Computer Engineering University of Minnesota Duluth 1023 University Drive Duluth, MN 55812		10. Project/Task/Work Unit No. CTS Project #2012016 11. Contract (C) or Grant (G) No.	
12. Sponsoring Organization Name and Address Intelligent Transportation Systems Institute Center for Transportation Studies University of Minnesota 200 Transportation and Safety Building 511 Washington Ave. SE Minneapolis, Minnesota 55455		13. Type of Report and Period Covered Final Report 14. Sponsoring Agency Code	
15. Supplementary Notes http://www.its.umn.edu/Publications/ResearchReports/			

16. Abstract (Limit: 250 words)

Automatic traffic data collection can significantly save labor work and cost compared to manual data collection. However, automatic traffic data collection has been one of the challenges in Intelligent Transportation Systems (ITS). To be practically useful, an automatic traffic data collection system must derive traffic data with reasonable accuracy compared to a manual approach. This project presents the development of a multiple-camera tracking system for accurate traffic performance measurements at intersections. The tracking system sets up multiple cameras to record videos for an intersection. Compared to the traditional single-camera based tracking system, the multiple-camera one can take advantage of significantly overlapped views of the same traffic scene provided by the multiple cameras such that the notorious vehicle occlusion problem is alleviated. Also, multiple cameras provide more evidence of the same vehicle, which allows more robust tracking of the vehicle. The developed system has mainly three processing modules. First, the camera is calibrated for the traffic scene of interest and a calibration algorithm is developed for multiple cameras at an intersection. Second, the system tracks vehicles from the multiple videos by using powerful imaging processing techniques and tracking algorithms. Finally, the resulting vehicle trajectories from vehicle tracking are analyzed to extract the interested traffic data, such as vehicle volume, travel time, rejected gaps and accepted gaps. Practical tests of the developed system focus on vehicle counts and reasonable accuracy is achieved.

17. Document Analysis/Descriptors	18. Availability Statement		
Intelligent transportation systems, Camera-based vision systems, Cameras, Multiple cameras, Tracking systems, Traffic data, Software, Implementation	No restrictions. Document available from: National Technical Information Services, Alexandria, Virginia 22312		
19. Security Class (this report)	20. Security Class (this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	76	

Development of a Multiple-Camera Tracking System for Accurate Traffic Performance Measurements at Intersections

Final Report

Prepared by:

Hua Tang

Department of Electrical and Computer Engineering
Northland Advanced Transportation Systems Research Laboratories
University of Minnesota Duluth

February 2013

Published by:

Intelligent Transportation Systems Institute
Center for Transportation Studies
University of Minnesota
200 Transportation and Safety Building
511 Washington Ave. S.E.
Minneapolis, Minnesota 55455

Acknowledgments

The author wishes to acknowledge those who made this research possible. The study was funded by the Intelligent Transportation Systems (ITS) Institute, a program of the University of Minnesota's Center for Transportation Studies (CTS). Financial support was provided by the United States Department of Transportation's Research and Innovative Technologies Administration (RITA).

The project 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.

The authors would like to thank St. Louis County for its special 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. The authors also thank Mr. Victor Lund for providing valuable information on the test laboratory.

The authors would like to thank Dr. Eil Kwon, director of NATSRL at the University of Minnesota Duluth, for very valuable discussions and comments.

Table of Contents

List of Figures

List of Tables

Executive Summary

In Intelligent Transportation Systems (ITS), traffic performance measurements or traffic data collection has been a challenge. The collected traffic data is necessary for traffic simulation and modeling, performance evaluation of the traffic scene, and eventually (re)design of a traffic scene. Video-Based traffic data collection has become popular in the past twenty years thanks to the technology advancements in computing power, camera/vision technology and image/video processing algorithms. However, manual data collection by human inspection of the recorded video is traditionally used by traffic engineers. Such a manual approach is very laborious and costly. Therefore, in past years, automatic traffic data collection has become an important research topic in ITS. Automatic traffic data collection for highways has become available now and even commercial tools have been developed today. However, due to much more complicated traffic scenes and traffic behavior, automatic traffic data collection for intersections and roundabouts has been left behind.

The proposed project is to develop a multiple-camera tracking system for automatic accurate traffic performance measurement for roundabouts and intersections. Traditional tracking system employs a single camera for traffic performance measurements. Compared to the traditional single-camera system, a multiple-camera one has two major advantages. One is that multiple cameras provide overlapped view coverage of the same traffic scene from different angles, which may significantly alleviate the vehicle occlusion problem. Also, multiple cameras provide multiple views of the same vehicle giving additional evidence of the vehicle, which allows robust vehicle tracking hence improves tracking accuracy.

The proposed multiple-camera tracking system consists of three processing modules, camera calibration, vehicle tracking and data mining. For calibration of multiple cameras, the main difference from that of a single camera is that a common world coordinate must be selected for multiple cameras. In this project, we extend the traditional method for single-camera calibration based on vanishing points to accommodate multiple cameras. The key of the extended method is the setup of a square or rectangular pattern such that parallel lines are available to allow the traditional method to derive vanishing points. The distance estimations using the calibration results are shown to be above 90% on average.

Once the camera is calibrated, the next processing module in the developed traffic data collections system is to process the videos to allow vehicle tracking. Compared to vehicle tracking with a single video from a single camera, vehicle tracking with multiple cameras has some similarities but also significant differences. In our vehicle tracking module, the first step is to segment objects based on the Mixture-of-Gaussian (MoG) algorithm. In MoG, each pixel is modeled probabilistically with a mixture of several Gaussian distributions (typically 3 to 5), and each background sample is used to update the Gaussian distributions, so that the history of pixel variations can be stored in the MoG model. Latest light and weather changes can be reflected by updating the mean and variance of MoG. Hence, the sensitivity and accuracy of vehicle detection is highly improved compared to traditional techniques. Another very important advantage of MoG is its robustness against camera shaking, which is one very practical problem that affects tracking accuracy. With segmented objects, we perform image processing, such as noise filtering, to refine the objects. Then, all potential objects are transformed from the image

coordinate to the world coordinate, followed by object extraction and validation. Subsequently, the extracted objects from multiple cameras are all checked for correspondence based on overlap, as two objects from two different cameras should ideally overlap exactly if they are the same object. This step of vehicle correspondence is not a problem in a single-camera system but a very important one for a multiple-camera system. Once objects from different cameras are corresponded, they are used to create potential vehicles for the current image frame with their states (which describes the vehicle's 3D details including position and speed in the world coordinate, shape, size, etc.). The next step is to associate the created potential vehicles at current image frame to existing vehicles tracked so far up to the previous image frame. This association is again based on overlap between potential vehicles and existing vehicles in our system design. To facilitate this association, we used Kalman filtering techniques for prediction of the states of existing vehicles in the current image frame from the previous image frame. We consider mainly three types of associations between potential vehicles and existing vehicles, namely one potential vehicle to one existing vehicle association, one potential vehicle to multiple existing vehicles association, and multiple potential vehicles to one existing vehicle association. Each type of association could possibly have multiple scenarios to consider, and each scenario requires a different scheme of updating existing vehicles using potential vehicles, which is the last step of the vehicle tracking module. We did not consider the case of many potential vehicles associated to many existing vehicles in order to simplify the vehicle tracking confusion. So finally, the existing vehicles that had been tracked up to previous frame can now be tracked to the current image frame. The vehicle tracking module generates the outputs as vehicle trajectories for all vehicles that have been detected and tracked by the module. The accuracy of these vehicle trajectories ultimately determines the traffic data accuracy.

Given vehicle trajectories from the vehicle tracking module, the next processing module of the traffic performance measure system is to mine the vehicle trajectories to automatically extract interested traffic data, including vehicle volume, vehicle speed, acceleration/de-acceleration behavior, accepted gaps, rejected gaps, follow-up time, lane change and so on. In our system design, we focus mainly on vehicle counts for all directions of traffic at an intersection. To find a vehicle's moving direction, we detect how its x and y coordinates in the world coordinate change throughout its lifetime in the camera views.

The developed traffic performance measurement system is currently implemented in combined C programming languages and MATLAB software running on personal computers. Also, a local test laboratory is setup at the intersection of W Arrowhead Rd and Sawyer Avenue and the intersection of W Arrowhead Rd and Arlington Avenue in Duluth, MN. Two cameras are set up for each intersection, giving two videos for the same intersection from two different sites. We have performed extensive testing of the system using offline videos recorded for the intersections. The automatically extracted traffic data have been compared to ground truth data that were manually collected by inspecting the video and it was found that the automatically extracted traffic data are reasonably accurate with up to 90% accuracy, which is an improvement compared to a single-camera system based on our experience. It was also found that the main factors that affect the accuracy are relatively poor vehicle detections from vehicle segmentations and neglect of some complex types of vehicle associations in the vehicle tracking algorithm.

Chapter 1. Introduction

1.1 Traffic Data Collection and Different Methods

Traffic performance measurements or traffic data collection has been an important topic in Intelligent Transportation Systems (ITS). Traffic data, such as vehicle count, vehicle speed, acceleration/de-acceleration behavior, accepted gaps, rejected gaps, follow-up time, lane changes and so on, are vitally important for understanding driver behavior, simulation and modeling of traffic, performance/safety evaluation of the traffic scene and eventually design/redesign of a traffic scene. In literature, there are many technologies or techniques for traffic data collection, such as inductive loop, microwave radars, sensors or cameras. Each method has its strength and weakness. For example, whereas the inductive loop technique is mature and well understood, its installation requires pavement cut and also the accuracy decreases when a large variety of vehicle classes are to be detected. Microwave radar, whereas it is easier to install and could measure speed directly, has difficulty when detecting stopped vehicles. Small and portable sensor devices that can be easily installed on the ground without pavement cut are being developed to allow efficient vehicle detection and counting. Another emerging technology is vehicle-based sensor networks, which collects data by locating vehicles via mobile phones or Global Positioning System (GPS) devices over the entire road network. But this technology has raised many concerns about drivers' privacy and it also requires devices installed in all vehicles.

The type of data that can be collected also varies between these methods. Whereas most methods could produce vehicle count and speed, not all of them could classify vehicle type or be employed for multiple lane detection. Table 1.1 shows traffic output data and communication bandwidth of typical technologies. Their equipment costs and lifetime can be found in Table 1.2. Those data gives us a glimpse of how to choose appropriate technology given available budget and desired traffic data to be collected. As it is seen from Table 1.1, regarding the most variety of traffic data that could be collected, video image processor is the winner. This is a very important advantage for some complex traffic scenes such as a roundabout or an intersection for which there are many traffic data specifications that are interesting to the traffic engineers in order to fully assess the performance of the traffic scene. For example, traffic performance measurements for highway target mostly vehicle volume, vehicle speed and lane use, but for roundabouts or intersections, other measurements such as origin-destination pairs, waiting time and gap size are needed as well.

1.2 Camera-Based Vision System

Camera-based vision systems have been applied for traffic data collection since 1970s. However, in the early days, collection of traffic data are achieved by human inspection of the real-time video or an offline recorded video. Even today, the manual approach is still very popular, especially for traffic data collection of complex traffic scenes such as roundabouts and intersections. For example, in [1] and [2] pre-recorded videos are inspected to collect vehicle counts at roundabouts. While the manual approach gives more accurate traffic data, it is very laborious and costly. Therefore, in recent years automatic traffic data collection has become one of the active research areas in ITS, in order to reduce cost and improve efficiency.

Table 1.1: Traffic output data and communication bandwidth of available sensors [3].

Sensor technology	Count	Presence	Speed	Output data	Classification	Multiple lane, multiple detection zone data	Communication bandwidth
Inductive loop	✓	✓	✓[b]	✓	✓[c]		Low to moderate
Magnetometer (two axis fluxgate)	✓	✓	✓[b]	✓			Low
Magnetic induction coil	✓	✓[d]	✓[b]	✓			Low
Microwave radar	✓	✓[e]	✓	✓[e]	✓[e]	✓[e]	Moderate
Active infrared	✓	✓	✓[f]	✓	✓	✓	Low to moderate
Passive infrared	✓	✓	✓[f]	✓			Low to moderate
Ultrasonic	✓	✓		✓			Low
Acoustic array	✓	✓	✓	✓		✓[g]	Low to moderate
Video image processor	✓	✓	✓	✓	✓	✓	Low to high[h]

Table 1.2: Equipment cost of some detectors [4].

Unit Cost Element	Lifetime (years)	Capital Cost ($1000)	Cost Date	O&M Cost ($1000)	Cost Date
Inductive Loop Surveillance on Corridor	5	3-8	2001	0.4-0.6	2005
Inductive Loop Surveillance at Intersection	5	8.6-15.3	2005	0.9-1.4	2005
Machine Vision Sensor on Corridor	10	21.7-29	2003	0.2-0.4	2003
Machine Vision Sensor at Intersection	10	16-25.5	2005	0.2-1	2005
Passive Acoustic Sensor on Corridor		3.7-8	2002	0.2-0.4	1998
Passive Acoustic Sensor at Intersection		5-15	2001	0.2-0.4	2002
Remote Traffic Microwave Sensor on Corridor	10	9-13	2005	0.1-0.58	2005
Remote Traffic Microwave Sensor at Intersection	10	18	2001	0.1	2001
Infrared Sensor Active		6-7.5	2000		
Infrared Sensor Passive		0.7-12	2002		
CCTV Video Camera	10	9-19	2005	1-2.3	2004
CCTV Video Camera Tower	20	4-12	2005		

On the other hand, thanks to vast technology advancements in cameras, computing powers (both software and hardware), and image processing algorithms, automatic traffic data collection subject to acceptable accuracy using camera-based vision systems has become possible. In literature, there has been a significant amount of work on automatic traffic performance measurements for highways using camera-based vision systems. Systems developed for highway scenes are relatively mature and successful thanks to the relatively simple moving patterns of vehicles and simple performance measurements such as volume and speed. Some representative work can be found in [5-9]. In the following, we will not go into detail for the developed systems for highways. Instead, we focus on systems that are developed for traffic performance measurements at roundabouts and intersections, as these traffic scenes are more challenging.

The most popular approach to traffic performance measurements for intersections is based on detection of vehicles, for instance using loop detectors, radars or sensors. Recently, in [10], an approach using wireless sensor networks is proposed for traffic performance estimation at intersections. For each way of the intersection, a sensor is placed in each lane. When a vehicle drives through the sensor, the timing is recorded in the sensor's log. Then, the logs of all sensors are analyzed offline to estimate the vehicle turning volume. While the detection part could be accurate using sensors, however, a number of issues exist. First, there is inherent ambiguity in the process of analyzing the logs. As shown in [10], depending on the specific timings of the vehicles, some vehicle trajectories would be mis-associated. Second, the major limitation of the system is that it can only estimate traffic performance measurements like vehicle turning volume. Other performance measurements, such as vehicle speed, waiting time, accepted and rejected gaps, lane use and origin-destination pairs for roundabouts, may not be handled. A very similar project and developed system based on detection using sensors is also reported in [11].

The alternative approach to traffic performance measurements is based on tracking of vehicles using video processing techniques. Compared to detection based approach, tracking-based approach is more like a direct measurement approach, which is more versatile in terms of traffic performance measurements. In tracking-based approach, what is ultimately desired is the moving trajectory of each individual vehicle from its entrance to the scene until its exit, as this gives comprehensive information that can be used for all types of traffic performance measurements. Video-Based approach has another advantage that it can also be manually inspected to collect ground-truth data for comparison purposes as well. Some commercial camera-based vision systems such as Autoscope [12] and Miovision [13] are available today for automatic traffic performance measurements. Autoscope can be used in different traffic scenes, however its main purpose is to set up a video-based detection or monitoring system for traffic engineers while producing limited performance measurements such as vehicle count and vehicle speed. Miovision is restricted to vehicle turning count for intersections though continuing research work is ongoing. Also, Miovision does not sell the software, but sells time to process submitted video clips.

1.3 Overview of a Multiple-Camera Tracking System

In tracking-based systems using camera-based vision systems, the accuracy of traffic performance measurement is directly proportional to vehicle tracking accuracy, the improvement of which is of most importance. In the above-mentioned video-based systems and also in our

past project [14], vehicle tracking is achieved by processing a video from a single camera. Although it is found that the accuracy of traffic performance measurements is encouraging (on average 70-90%), there are some problems not adequately resolved, mainly due to limited view of a single camera. When combining multiple camera views to obtain a joint tracking, it is typically much better than single-camera tracking as they can provide adequate view coverage of the traffic scene of interest and also multiple views of the same vehicle provide additional evidence of the vehicle which allows robust vehicle tracking and possibly significant alleviation of vehicle occlusion. Therefore, recently there are attempts to design systems to track vehicles using videos from multiple cameras, called multiple-camera tracking systems. Developing such a multiple-camera tracking system is also the main target of this project. Note that here, we do not strictly differentiate whether the multiple-camera tracking system is to be operating in real-time or offline, as the conceptual design in either case is mostly the same, except that the real-time operation of a multiple-camera tracking system would typically has much more stringent requirement on execution time of the system. In the following, we review some previous work on multiple-camera tracking system and motivate our proposed system developed in this project.

In general, reported multiple-camera tracking systems differ mainly in terms of what measurements each camera produces and how the overall measurements from multiple cameras are fused for a joint tracking. Dockstader presented a multiple-camera tracking system for multiple interacting persons in motion [15]. Each camera independently tracks persons using 2-dimensional (2D) semantic features, and then these features are collectively integrated using a Bayesian belief network with a topology that varies as a function of scene content and feature confidence. The network fuses measurements from multiple cameras by resolving independency relationships and confidence levels with the graph, thereby producing the most likely 3D state estimate of persons. Tyagi proposed an interesting method for direct 3D tracking using multiple cameras in smart rooms [16]. However, the tracking algorithm relies on color histogram of the objects, which works well in indoor environments. Our experiments show that it is not well extended to vehicle tracking in Intelligent Transportation Systems (ITS) as colors of vehicles are not apparent depending on weather and size of the vehicles. Bhuyan's multiple-camera tracking system uses a shape-based tracking algorithm for each camera and then fuses the tracking results from multiple cameras by a simple linear weighting scheme [17]. The shaped-based tracking algorithm may not be suitable to outdoor vehicle tracking, as vehicle sizes and shapes may change significantly during the course. Another multiple-camera tracking system recently proposed in [18] tracks vehicles using a region-based algorithm and then fuses the region measurements from three cameras in real 3D world. This system has been applied only to highway scenes. An alternative multiple-camera tracking system proposed in [19] uses multiple cues (such as color, shape and region) for vehicle tracking and then fuses the region measurements from multiple cameras to jointly estimate the 3D state of the vehicle. This system has been applied to outdoor tracking and reported reasonably accurate results.

All the above-mentioned systems set up their multiple cameras to provide significantly overlapped view of the targeted traffic scene so that a vehicle at any time is very likely to appear in the views of multiple cameras and not only vehicle occlusion may be significantly alleviated, but also multiple views of the same vehicle gives more evidence for robust vehicle tracking. There are also many multiple-camera tracking system targeting covering a large area of traffic scene (such as arterial roads) with each camera taking care of part of the scene. Not any two cameras have overlapped views, but has two consecutive cameras placed along the traffic scene.

For example, Dixon's multiple-camera system does so [20]. It tracks vehicles using a set of space-time sheets and the data fusion problem is expressed as a graph optimization problem using network flows [20]. Finally, it should be noted that most systems discussed above do not involve traffic performance measurements, but only tracking itself. Another tracking system for people counting is proposed in [21], however only multiple positions of a single camera are explored, which is essentially a single-camera system.

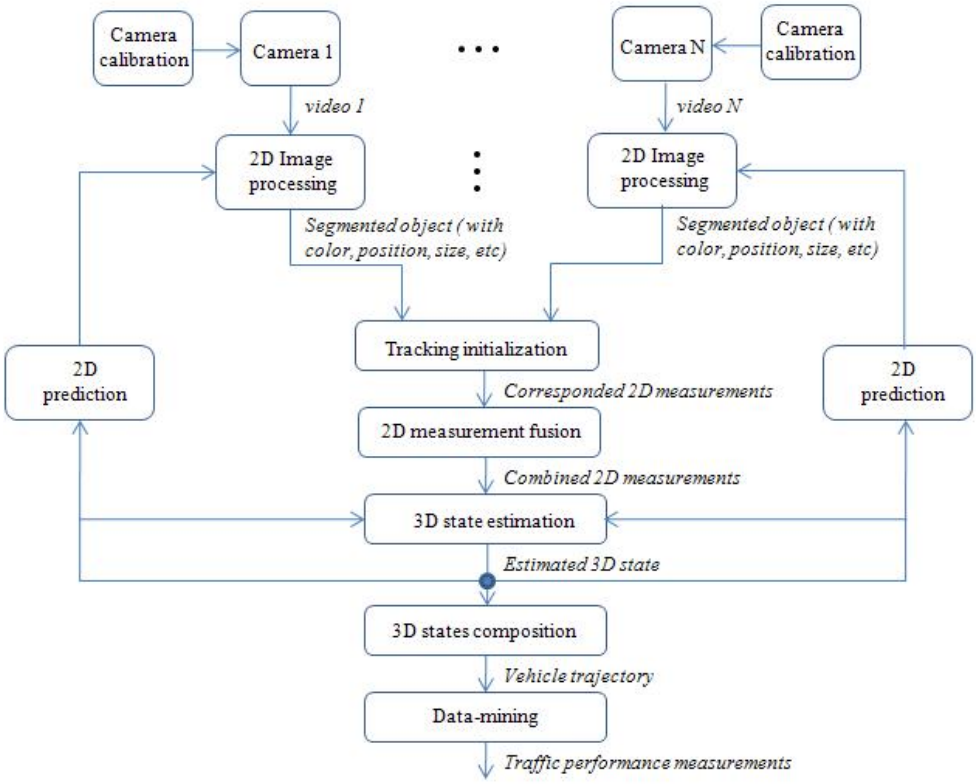


Figure 1.1: Processing flow of a multiple-camera tracking system (result from each step is shown in italics).

For most of the multiple-camera tracking systems discussed above, the processing flow is shown in Figure 1.1. In the following, for simplicity, we suppose there are two fixed-location cameras set up for the traffic scene of interest. The two-camera tracking system tracks vehicles in basically four steps. First, each camera needs to be calibrated. The transformation from the real-world position (x_w, y_w, z_w) to the image position (Q_x, Q_y) is formulated as

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} T_{11} & T_{21} & T_{31} \\ T_{12} & T_{22} & T_{32} \\ T_{13} & T_{23} & T_{33} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \begin{bmatrix} T_{41} \\ T_{42} \\ T_{43} \end{bmatrix}, \qquad Q_x = \frac{x_c}{z_c} f, \;\; Q_y = \frac{y_c}{z_c} f \tag{1}$$

where the rotation matrix and the translation vector are called the homogeneous transformation *T* and *f* is the effective focal length. Camera calibration is to find out *T* and *f*. Suppose the transformation and focal length is respectively *T* and *ft* for camera I, and *S* and *fs* for camera II.

Then, the following equations are obtained between the image position (Q_{1x}, Q_{1y}) from camera I, (Q_{2x}, Q_{2y}) from camera II and the real-world position (x_w, y_w, z_w)

$$Q_{1x} = \frac{x_w T_{11} + y_w T_{21} + z_w T_{31} + T_{41}}{x_w T_{13} + y_w T_{23} + z_w T_{33} + T_{43}} f_T = \frac{x_{c,1}}{z_{c,1}} f_T \text{ , } Q_{1y} = \frac{x_w T_{12} + y_w T_{22} + z_w T_{32} + T_{42}}{x_w T_{13} + y_w T_{23} + z_w T_{33} + T_{43}} f_T = \frac{y_{c,1}}{z_{c,1}} f_T$$

$$Q_{2x} = \frac{x_w S_{11} + y_w S_{21} + z_w S_{31} + S_{41}}{x_w S_{13} + y_w S_{23} + z_w S_{33} + S_{43}} f_S = \frac{x_{c,2}}{z_{c,2}} f_S \text{ , } Q_{2y} = \frac{x_w S_{12} + y_w S_{22} + z_w S_{32} + S_{42}}{x_w S_{13} + y_w S_{23} + z_w S_{33} + S_{43}} f_T = \frac{y_{c,2}}{z_{c,2}} f_S \quad (2)$$

Second, the 2D measurement from each camera is extracted, for instance, the position measurement (Q_{1x}, Q_{1y}) or (Q_{2x}, Q_{2y}) in equation (2). Typically, object segmentation is performed first before extraction.

Third, the 2D measurements from two cameras corresponding to the same vehicles need to be associated. For example, the problem is to determine whether position (Q_{1x}, Q_{1y}) from camera I and (Q_{2x}, Q_{2y}) from camera II correspond to the same vehicle. This correspondence initialization or tracking initialization is an important step of a multiple-camera tracking system. One can achieve tracking initialization by solving a constrained linear least-squared problem formulated as follows [19]

$$min_{[x_w, y_w, z_w]} \left\| \begin{matrix} f_T x_{c,1} - Q_{1x} z_{c,1} \\ f_T y_{c,1} - Q_{1y} z_{c,1} \\ f_s x_{c,2} - Q_{2x} z_{c,2} \\ f_s y_{c,2} - Q_{2y} z_{c,2} \end{matrix} \right\| \text{ subject to } z_w > 0 \quad (3)$$

where the symbols are defined in equation (1) and (2) above. The sum of errors is minimum when all vehicles in the view of camera I correctly find their correspondences in camera II. Note that a constraint is added since the moving object in real 3D world always has a non-zero height and this constraint helps remove the vehicle shadow on the ground plane whose height is zero.

Fourth, after the measurements from both cameras are properly corresponded from the previous step, the 2D measurement results from two cameras are fused to estimate 3D states of vehicles. Suppose the 3D state of a vehicle is denoted by X_k=$[x_w, y_w, z_w, x_w', y_w', z_w']$, where k is the time index and $[x_w', y_w', z_w']$ the speed of the vehicle. Then, the dynamic equation related to the state vector X_k is $X_{k+1} = \emptyset_k X_k + W_k$, where $\emptyset_k$ is the transition matrix and W_k the white noise. The combined measurements are denoted by $Z_k = [Q_{1x,k}, Q_{1y,k}, Q_{2x,k}, Q_{2y,k}]$ (assuming the two position measurements are corresponded) where k again is the time index. Z_k is related to X_k with the following equation

$$Z_k = \left[Q_{1x,k}, Q_{1y,k}, Q_{2x,k}, Q_{2y,k}\right] = h_k(X_k) + V_k \quad (4)$$

where V_k is assumed white noise with known covariance matrix and h_k a non-linear function that projects the 3D state X_k to the image measurements Z_k. This dynamic equation is non-linear, therefore Extended Kalman Filtering (EKF) can be used to estimate X_k. The EKF uses a linearization of the equations about the current best estimate of the sate to produce minimum mean-squared estimates of the next state. Other advanced filters, such as unscented Kalman filter and particle filter, can be used as well [15-19].

Subsequently, the 3D states of a vehicle at all-time moments are composed to derive the vehicle trajectory. Note that the accuracy of these vehicle trajectories ultimately determined the accuracy of collected traffic data. Accuracy of vehicle trajectories mainly depends on the intelligence of the vehicle tracking algorithms, which is a hot research area in ITS.

After all vehicle trajectories are available from vehicle tracking, they are finally processed to extract traffic performance measurements. This step does not depend on the number of cameras used in the tracking system. Also, this step does not incur any accuracy loss.

It can be seen that the two-camera tracking system provides additional evidence on the state of a vehicle, therefore allowing more robust tracking of the vehicle. Also, multiple cameras can significantly alleviate the problem of vehicle occlusion and hence improve tracking accuracy. The accuracy could improve from on average 80% for the conventional single-camera system to 95% as reported in some literature work [15-20]. In addition, the multiple-camera tracking approach allows 3D tracking of the vehicles, therefore 3D measurements of vehicles such as width, length and height.

1.4 The Proposed Multiple-Camera Tracking System

In this project, we propose the development of a multiple-camera tracking system for automatic traffic data collection system for roundabouts and intersections. The system targets roundabouts and intersections because no mature data collection systems exist for these traffic scenes yet in contrast to highway scenes. With the proposed multiple-camera tracking system, limited view and vehicle occlusion can be addressed to an adequate extent compared to conventional single-camera-based video systems.

The processing flow of the proposed multiple-camera tracking system is similar to the one shown in Figure 1.1, however details differ from previous systems at each step. Some major differences are as follows. First, we propose to extend the traditional calibration method for a single-camera system to accommodate multiple-camera system by using a square template from intersections. Second, we process each video from an individual camera in the world coordinate instead of the image coordinate as in previous systems, so that vehicle correspondence initialization can be simplified based on overlap of objects from different cameras. Third, the proposed multiple-camera tracking systems includes a data mining module to extract traffic data after vehicle trajectories are obtained from vehicle tracking, which makes the proposed system more useful to traffic engineers. Finally, the proposed multiple-camera tracking system targets intersections, which is not addressed in previous works. Also, the proposed system can be extended to roundabouts though intersection is the main focus of this project.

1.5 Organization of the Report

In the following chapters, we describe in detail the developed multiple-camera tracking system for automatic and accurate traffic performance measurements of roundabouts and intersections. We consider that the proposed system consists of three modules, the camera calibration module, the vehicle tracking module and the data mining module.

- In Chapter two, we describe the camera calibration module and present an extended approach for efficient calibration of multiple cameras for intersection traffic scenes.
- In Chapter three, we describe the vehicle tracking module and present the vehicle tracking algorithm in detail.
- In Chapter four, we describe the data mining module and the algorithms for traffic data collection.
- Finally, conclusions are given in chapter five.

Chapter 2. Camera Calibration of Multiple Cameras

As introduced and motivated in Chapter 1, the goal of this project is to develop a multiple camera-based vehicle tracking system for automated collection of traffic performance measurements. For any camera-based video system in ITS, whether there is only a single camera or multiple cameras, camera calibration is always the pre-requisite. The purpose of camera calibration is to establish the projection relationship between the vehicle in the real world and those in the images so that a distance in the 2D image space can be mapped to a real-world distance in the 3D world. Then, vehicle speed and vehicle sizes can be identified.

While there are well established methods for camera calibration of a single camera, camera calibration of multiple cameras could be more challenging. When developing a calibration method for multiple cameras, one desired feature is that the method should be simple to use yet accurate in terms of distance estimation. In this project, we extend the simple parallel-line-based method previously used for camera calibration of a single camera to accommodate multiple cameras. In the extended method, a rectangular region in the road is used as a simple calibration pattern, which specifies the common unique world coordinate system. Then, each camera is first separately calibrated using the traditional parallel line based method and the two parallel lines should be from the rectangular region. The parallel line based method would give the focal length, tilt angle and pan angle of each camera, however the assumed world-coordinate in the parallel line method does not align with specified common world coordinate defined in the rectangular region . Therefore, we then perform a post-optimization by minimizing the projection error for the selected rectangular region so that the coordinate shifts between the assumed world coordinates for each separate camera and the common unique world coordinate for all cameras are computed. Such a post-optimization is done for each camera. Finally, all cameras will have the projection relationship between their image coordinates and the common unique world coordinate.

The rest of the Chapter is organized as follows. Section 2.1 gives an introduction of previous methods for camera calibration of single and multiple cameras. Section 2.2 describes the extended method for camera calibration of multiple cameras in detail. Section 2.3 presents experimental results of the extended method tested on real-world intersection scenes. Finally, conclusions are drawn in Section 2.4.

2.1 Introduction and Previous Work

Camera calibration is to estimate the camera's perspective information from images of real-world objects. It is a well-grown field in computer vision and many techniques for camera calibration have been proposed [22]. According to the dimension of calibration objects, they can be classified into four categories, 3D reference object based calibration (using orthogonal planes) [23], 2D-plane based calibration (using planar objects) [24][25], 1D-line based calibration (using a set of collinear points) [26], and 0D calibration (or self-calibration) that does not use any calibration object [27].

Due to widespread use of camera-based vision systems in ITS for video recording, traffic monitoring and surveillance, camera calibration has also become an important topic in ITS. The

extracted camera parameters are typically necessary for estimation of vehicle speed [28], object classification and tracking [29], etc. In [30], camera calibration is also used for resolving vehicle occlusions. However, camera calibration in ITS is different from that in computer vision. In computer vision, accurate camera calibration, which is to fully extract both the intrinsic and extrinsic parameters, mostly requires the use of well-designed patterns (such as a 2D check board [24-25]) to be placed in the field of view of the cameras and multiple images of the pattern from different orientations [31]. However, such patterns are seldom available for camera calibration of traffic scenes in ITS. Therefore, camera calibration in ITS applications usually will not (and can not) calibrate all intrinsic parameters such as focal length, principal point, skew, aspect ratio and radial distortion [31]. In ITS applications, camera calibration of practical traffic scenes mostly considers only one intrinsic parameter, focal length. As shown in [31], with the restriction to consider only one intrinsic parameter (focal length) and to have constant aspect ratio, known principal point, no skew and distortion, the re-projection error is still acceptable. If an elaborate intrinsic model of all intrinsic parameters is indeed necessary, it can be calibrated a priori as in computer vision applications [22-27][31].

On the other hand, in recent years Pan-Tilt-Zoom (PTZ) cameras [28] have been widely used in ITS applications for monitoring and surveillance purposes and these cameras are usually mounted high above the ground to overlook the traffic scene so that no rotation angle is necessary. As shown in [28], the reduced two-angle model (with pan and tilt angle) has less than 10% bias in distance measurements compared to the full three-angle model even for road grades of 2% .

From the discussion in the above two paragraphs, it can be seen that camera calibration of practical traffic scenes in ITS applications has its own restrictions on available scene patterns, accordingly on intrinsic and extrinsic parameters. Therefore, most camera calibration methods in computer vision are not suitable to practical traffic scenes in ITS applications. Besides, most of those methods involve complex calculations and optimizations to fully extract intrinsic and/or extrinsic parameters. As a result, in the past years simplified methods to perform camera calibration in ITS applications have been proposed. In the following, we briefly review some previous work for camera calibration of traffic scenes in ITS applications and then motivate our work.

Most previous works on camera calibration in ITS applications have targeted roadway scenes such as highways, in which vehicle moving directions are known. For roadway scenes, the geometry setup of the camera for calibration is typically described as shown in Figure 2.1 [32-37]. Note that from the side view, the camera is installed above the ground plane with height h and has a tilt angle of φ with respect the ground plane which is assumed flat. Also, the Cz axis defined in the camera coordinate starts from the focal point, points from the camera's focal point to the ground, and intersects the ground plane at a point called world origin, denoted in yellow in the Figure. The Cy axis points downward to the ground with an angle of $(\frac{\pi}{2} - \varphi)$. The world axes are defined to start from the world origin. The Wz axis obviously is perpendicular to the ground plane and the Wy axis points along the line of projection of Cz axis. The top view in the upper-right of Figure 2.1 shows the definition of other parameters and axes. The Cx axis in the camera coordinate is parallel to the ground plane and in the side view it would point to the paper. Wx axis would be parallel to the ground plane as well. However, notice that the Wx and Wy

axes in the world coordinate rotates counter-clockwise with an angle of θ with respect to the line of projection of the Cz axis such that the Wy axis is parallel to the traffic lanes. This is not necessary but usually preferred as measuring distances and computing speeds of vehicles in the world coordinate would be simplified. Also note that two lines that are perpendicular to the traffic lanes are defined. They are shown in virtual lines because in practice these lines are hardly drawn in roads. The lane width is denoted as *w*. The lower plot of Figure 2.1 shows the projection of the traffic lanes and the world origin in an image. The world origin as defined above would be called the principal point, which is usually the center point of the image. Note that the parallel road lanes do not appear parallel in the image due to perspective projection and therefore the lines in the images intersect at a point usually named a vanishing point, denoted as (u0,v0). Similarly, the two perpendicular lines shown in the top view do not appear parallel either but intersects at a point named a vanishing point as well, denoted as (u1,v0). It should be noted that it can be proved that these two vanishing points share the same iy-axis coordinate v0 [28]. These parallel lines and vanishing points are the key to camera calibration in traditional methods.

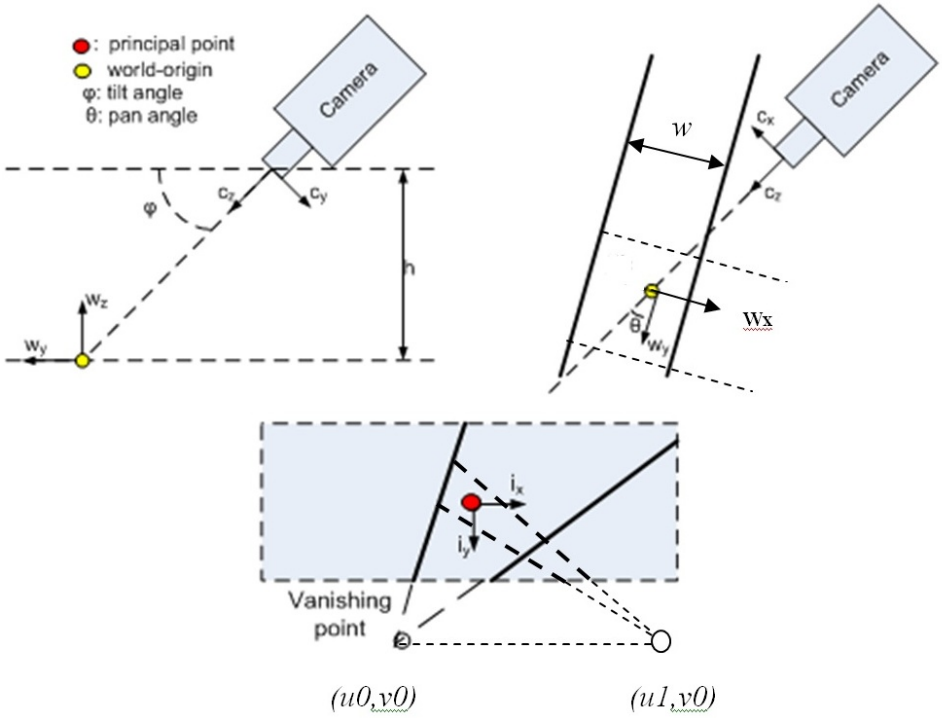


Figure 2.1: Side view and top view of the camera setup for roadway scenes and projection of real-world traffic lanes in the image coordinate.

With the above definitions, the main idea underlying traditional methods for camera calibration is to use one or two vanishing points from parallel lines to derive closed-form solutions of one intrinsic parameter (focal length) and two or three extrinsic parameters (pan angle, tilt angle and camera height). Some works obtain parallel lines from landmarks like points, lines, poles, or objects with known shape, while other works from moving vehicle trajectories.

The landmarks used most as in [33-34] are traffic lanes in the roads, which are supposed to be parallel in the real world but generally do not appear parallel in the image frames. In most cases,

these parallel lines can be manually picked with inspection. In some cases, these traffic lanes can be automatically extracted from the images by various image processing algorithms such as the Hough-transform technique [33-34]. Once the traffic lanes are extracted, their intersection coordinate in the image is computed, which gives one vanishing point, denoted as (u0,v0) here.

The camera's intrinsic parameters in general can be represented by the following matrix *K* [38-39]

$$K = \begin{bmatrix} f & s & p_x \\ 0 & \tau f & p_y \\ 0 & 0 & 1 \end{bmatrix} \quad (1)$$

where parameter *f* refers to the camera's focal length with unit in pixels. The point (p_x, p_y) is the principal point in the image coordinate, τ the aspect ratio and *s* the skew factor. In most cases in traffic surveillance applications, we can assume that $(p_x, p_y) = (0,0)$ and $\tau=1$ and the error incurred is typically small [28][34]. The skew factor *s* is the amount by which the angle between the horizontal axis and vertical axis differs from 90 degrees. This parameter is often admitted to zero because true CCD cameras have *x* and *y* axes perpendicular [39].

The rotation matrix, representing the orientation of the camera coordinate with respect to the world coordinate, can be written as

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & sin\varphi & cos\varphi \\ 0 & -cos\varphi & sin\varphi \end{bmatrix} \quad (2)$$

Note that in (2), pan angle θ does not appear as we did not make Wy axis align the line of projection of camera Zx axis. But it will be shown later that pan angle is easily found.

With $C = \begin{bmatrix} w_{cx0} \\ w_{cy0} \\ w_{cz0} \end{bmatrix} = \begin{bmatrix} 0 \\ -h/tan\varphi \\ h \end{bmatrix}$ representing the camera's optical center in the world coordinate and the transformation vector $t = -RC$, perspective transformation from the world coordinate to the image planar coordinate can be described as

$$\begin{bmatrix} u \\ v \\ g \end{bmatrix} = K[R|t] \begin{bmatrix} w_x \\ w_y \\ w_z \\ 1 \end{bmatrix}; \quad \begin{bmatrix} i_x \\ i_y \end{bmatrix} = \frac{1}{g} \begin{bmatrix} u \\ v \end{bmatrix} \quad (3)$$

where $[u\ v\ g]$ is the homogenous vector. Substituting *K*, *R* and *t*, (3) can be rewritten as:

$$\begin{bmatrix} u \\ v \\ g \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f.sin\varphi & f.cos\varphi & 0 \\ 0 & -cos\varphi & sin\varphi & -h/sin\varphi \end{bmatrix} \begin{bmatrix} w_x \\ w_y \\ w_z \\ 1 \end{bmatrix} \quad (4)$$

In typical traffic scenes, the depth of images is large because the camera is often mounted high above the ground to have a wide view. For that reason, the height of objects (vehicles or some

landmarks) can be neglected to simplify the calculation. With Wz = 0, from (3) and (4) we can obtain the transformation equations between the image coordinate and the world coordinate as follows

$$i_x = -\frac{f w_x}{w_y.cos\varphi + \frac{h}{sin\varphi}} \tag{5}$$

$$i_y = -\frac{f sin\varphi . w_y}{w_y.cos\varphi + \frac{h}{sin\varphi}} \tag{6}$$

Solving (w_x, w_y) from (5) and (6), the image coordinate can be mapped back to the world coordinate as follows

$$w_x = -\frac{h.i_x}{f.sin\varphi + i_y.cos\varphi} \tag{7}$$

$$w_y = -\frac{h.i_y}{\sin\varphi.(f.sin\varphi + i_y.cos\varphi)} \tag{8}$$

Equation (7) and (8) reveal three important parameters for camera calibration, which are focal length *f*, tilt angle φ, and camera height *h*. As camera height *h* is usually available or measured, many calibration methods assume known *h*, but some other methods can calibrate *h* as well. In Table 2.1, we make a brief comparison of these methods in terms of what intrinsic parameter and extrinsic parameter are calibrated and what parameters are assumed known. As can be seen, in ITS applications, camera calibration usually considers only one intrinsic parameter, which is the focal length *f*. Regarding extrinsic parameters, pan angle θ and tilt angle φ are considered as PTZ cameras are mostly used in ITS today [28] and the rotation angle *r* is not necessary. In Table 2.1, only the method presented in [31] computes the rotation angle *r*. One improvement of that method is that more geometric primitives (such as normal to plane lines from traffic poles, building edges, and point-to-point distances) are used to allow post-optimization of the camera parameters in contrast to those methods that mostly rely on closed-form equations. The equations are shown as follows [28]:

$$f = \sqrt{-(v_0{}^2 + u_0 u_1)} \tag{9}$$

$$\varphi = \text{atan}(\frac{v_0}{f}) \tag{10}$$

$$\theta = \text{atan}(\frac{u_0 cos(\varphi)}{f}) \tag{11}$$

where u_0, $v_0$and u_1 are the coordinates of the vanishing points in the image coordinates (i_x, i_y) shown in Figure 2.1. Note that camera height *h* is in fact not involved in the above equations (9)-(11), but will be used later.

Table 2.1: A comparison of previous methods for camera calibration in ITS applications.

	Intrinsic parameter computed	Extrinsic parameters computed	Known parameters assumed	Number of vanishing points needed
[33]	f	φ, θ	h, w	1
[34]	f	φ, θ	h, w	1
[32]	f	θ	φ, h, w	1
[28]	f	φ, θ, h	w	2
[31]	f	φ, θ, h, r	w	2
[36]	f	φ, θ, h	w	2

It should be noted here that equation (10) to (12) requires two vanishing points and the first one comes from the intersection of the two parallel road lines, which are usually available in the real road and visible in the image of the road. But how to compute the other vanishing point? The second vanishing point is the intersection of lines perpendicular to the traffic lanes (the virtual lines denoted in Figure 2.1), but these lines are seldom available in the traffic scene. There are also various techniques to estimate the second vanishing point depending on what primitive features are used. In [28], bottom edges of vehicles are used with the assumption that the camera is oriented properly. In this case, note that the bottom edges are ideally perpendicular to the vehicle traveling direction which should be aligned with the road lanes. This technique may not work well if bottom edges of vehicles are distorted or of abnormal shape in case of shadow. In [36], edges of vehicle windows are used, which seem to be more homogenous in vehicles. However, in both cases, high resolution images of roads are preferred in order to have reasonable accuracy. In this project, since we target intersections, the perpendicular lines can simply come from the two perpendicular roads forming the intersection. If the two roads are not perpendicular or multiple roads form the intersection, then similar methods from [28][31] can be applied. In fact, some other primitive features, such as the pedestrian walk lanes, can be used as well.

One more note is about the selection of the parallel lines. In most previous works, manual inspection is more popular, but it does require manual intervention. To allow more automation, the alternative is to extract traffic lanes using image processing techniques, which may require expensive computational processing. Besides, in practice, traffic lanes may not appear clearly in images depending on the specific traffic scene or road condition. To address this concern, some works propose to track vehicles to find vehicle trajectories [28][35-36], which can be regarded as the equivalents of traffic lanes, and then estimate the vanishing point from the intersection of multiple vehicle trajectories. In [35], focal length, pan angle, tilt angle, and camera height can be found, but mean dimensions of vehicles (width, length) to calculate scale factors on vertical and horizontal axes are required. These techniques can be automated but need to collect data from a large number of image frames. In this project, we use the manual inspection method.

In summary, previous works on camera calibration of roadway scenes are mostly based on the parallel line method in which vanishing points are estimated. Using the vanishing points, these camera parameters can be efficiently derived from closed-form formulas, which is one very important advantage for ITS applications compared to more complex methods in the computer vision area. Finally note that such a method based on vanishing points are well applicable to general roadways, such as highways and intersections, however it does not to some other

roadways such as roundabouts as there are no parallel lines. In that case, a different method needs to be used. For example, methods for camera calibration of roundabouts in the case of a single camera are proposed in [14][31], which can be extended to accommodate multiple cameras. We will not go into detail for roundabouts.

Google Maps - ©2012 Google

Figure 2.2: Bird's eye view of the W Arrowhead and Sawyer Avenue Intersection.

2.2 An Extended Method for Camera Calibration of Multiple Cameras

In this section, we present an extended method for camera calibration of multiple cameras in the case of intersection traffic scenes. We adopted the basic method introduced in [28] for camera calibration of a single camera and extend it to accommodate multiple cameras. To start, we first show the intersection for which a local test laboratory with two camera systems has been setup. The intersection is located at the W Arrowhead Rd and Sawyer Avenue in Duluth, MN. A bird's eye view of the intersection is shown below in Figure 2.2. It can be seen that the two roads forming the intersection are basically perpendicular. Also, an earth view of the intersection is shown in Figure 2.3. As denoted by the two red arrows, two separate cameras are installed at the two traffic light poles, one at the southwest corner and the other at the northeast corner.

Google Maps - ©2012 Google

Figure 2.3: An earth view of the W Arrowhead and Sawyer Avenue Intersection.

Figure 2.4: Two images of an intersection traffic scene from two cameras (Images are captured from local cameras).

Two images of the intersection captured by the two camera systems are shown in Figure 2.4. Each image has a resolution of 352 by 240 pixels. The left figure is from the camera at the NE corner and the right one from the camera at the SW corner. Note that the two images in Figure 2.4 do not seem to match the intersection views in Figure 2.2 and 2.3 very well and this is mainly because the two cameras are installed in the two traffic light poles which are too close to the center of the intersection and therefore even in the most zoomed-out case the two cameras cannot cover the complete intersection. This is an ongoing issue that needs to be addressed.

The processing flow of the proposed extended method for calibration of multiple cameras are detailed as follows.

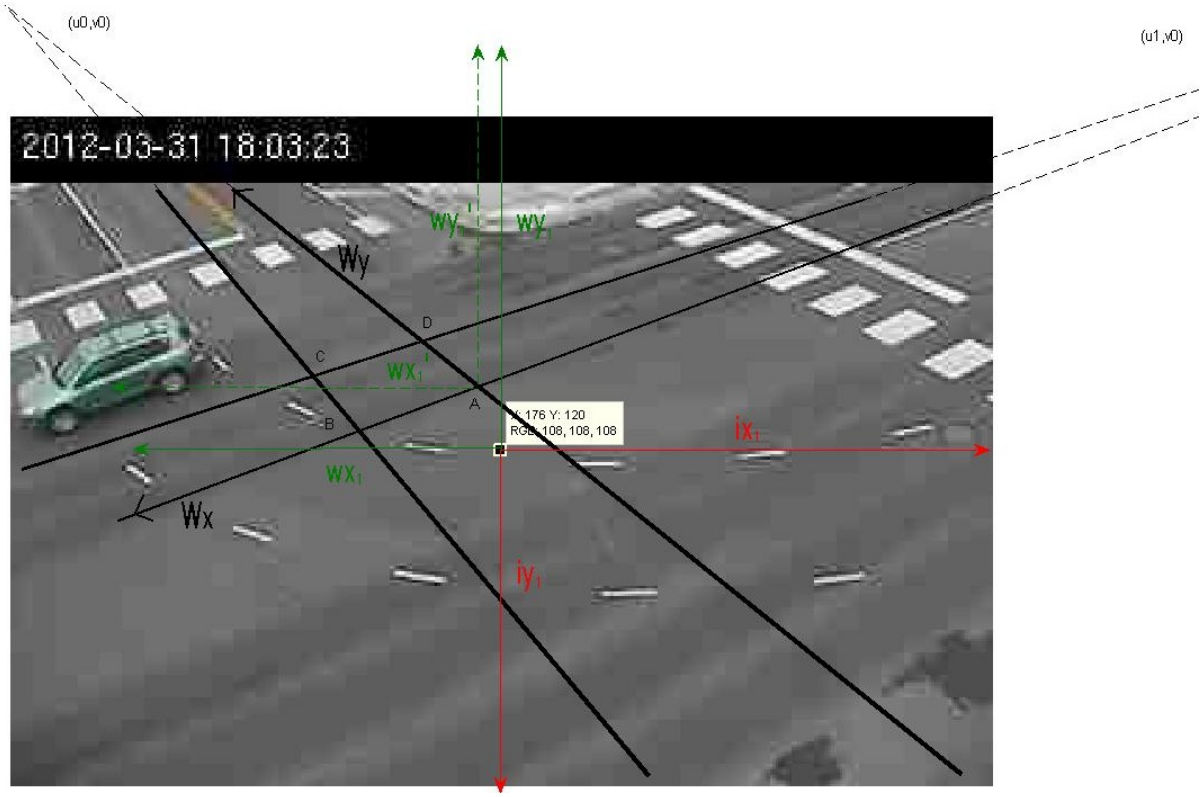


Figure 2.5: An enlarged view of the W Arrowhead and Sawyer Avenue Intersection from the first camera at the NE corner.

Step 1: the first step of the calibration process is to construct a common square or rectangular pattern that appears in both images and the purpose is to define a common world coordinate for both cameras. The common world axis is convenient for both cameras instead of each camera having its own world axis. A simple way to construct such a pattern is to select two parallel lines from each road and extend them until they cross each other to form the pattern. As shown in Figure 2.4, we select lines from the blocks of pedestrian crosswalk and the size of blocks and distance between blocks are known. For example, each block has a size of 3 by 6 feet and the distance between two blocks is 3 feet. Therefore, a square 9 by 9 feet pattern is constructed. Note that it would be even better to construct a pattern from the main road lanes, such as the yellow lanes in Sawyer Avenue. However, the reason we selected the block edges for the pattern is mainly because the limited camera view cannot provide reliable traffic lane selection in the W Arrowhead Rd (as inspected from the images). With such a square pattern that has four vertex, we can then arbitrarily select one as the origin of the world coordinate. In our case, we just select the vertices denoted as "A" as the origin, as can be seen in a larger view in Figure 2.5. Then, we can define the world coordinates. We define the W_Y axis as the one connecting vertex "A" and "D", and W_X axis connecting "A" and "B". Note that the way to define the two axes here is not arbitrary, but to follow the original definitions of W_Y and W_X axes as shown in Figure 2.1 in order to use the calibration method reviewed in Section 2.1. Referring to Figure 2.1, it can be seen that W_X axis should be at the left-hand side of W_Y axis instead of the right-hand side.

Step 2: The next step of the calibration process is to individually calibrate each camera using the method reviewed in Section 2.1. We explain this in detail as follows. Let us first calibrate the camera that captures the image shown in left-hand side of Figure 2.4, replicated here in Figure 2.5 for a larger view. First of all, we define the image coordinate axes, i_{x1} and i_{y1}, as in red in the Figure. Then, according to the method introduced in Section 2.1, we need to select two pairs of parallel lines that are perpendicular to each other to drive the two vanishing points. In fact, these two pair of lines are already used in the above step to define the square pattern. We select the two parallel lines in the Sawyer Avenue, lines formed by points "A" and "D", and "B" and "C" to derive the first vanishing point (u_{01}, v_{01}) as denoted in Figure 2.5 and then the two parallel lines from the W Arrowhead Rd that are perpendicular to the previous two, lines formed by points "A" and "B", and "D" and "C" to drive the second vanishing point (u_{11}, v_{01}). Note again that the v_{01} coordinate of the two vanishing points in the image coordinates (i_{x1}, i_{y1}) should be the same. Then, it is very important to define correctly the world coordinates in this case. According to Figure 2.1, the origin of the world coordinate should be the point in the road that is projected to the principal point in the image, which is typically the image center as explained in Section 2.1 before. So, for display purpose, in Figure 2.5, the w_{y1} axis would simply point from the center up and w_{x1} axis from the center left in the image. The corresponding top view of $w_{y1}$and w_{x1} axis in the world coordinate is given in Figure 2.6 and it can be compared to Figure 2.5.

The resolutions of the images shown in Figure 2.4 are 352 by 240. In computer vision areas, the origin of the image coordinate is defined at the upper-left corner of the image by default and this is the case of in the MATLAB [40] software used in our project. This explains why the principal point in Figure 2.5 has image coordinates of (176,120). Thus, it is important that we shift the default image axes to the center of the image to become image axes i_{x1} and i_{y1} as defined in the image. This means that the default image center coordinates of (176,120) would become (0,0) in the i_{x1} and i_{y1} axis.

Then, we applied the calibration method in Section 2.1. The calibration results are shown below using equations (9)-(11):

$$f_1=310,\ \varphi_1=0.773,\ \theta_1=0.61$$

Of these three calibration parameters, the pan angle θ_1 is shown in Figure 2.6. Note that the pan angle is defined between the w_{y1} axis and the $W_Y$axis (the W_Y axis is the one along which the parallel lines are selected to derive the first vanishing point u_0, v_0). Note that while the calibration using equations (10)-(12) has the advantage of quick solving, it does have a dis-advantage that the results may not be very stable, since the selection of the parallel lines is subject to human error. There are a few ways to alleviate this problem. One way is to have multiple trials which hopefully average out the error. The other way is to use an optimization-based approach instead of direct solving from equations. As it will be shown later, a post-optimization will be applied to further improve the calibration accuracy. The key point here is that these parameters obtained from the equations do provide a reasonably good initial solution for the post-optimization, which does not have to be extremely accurate.

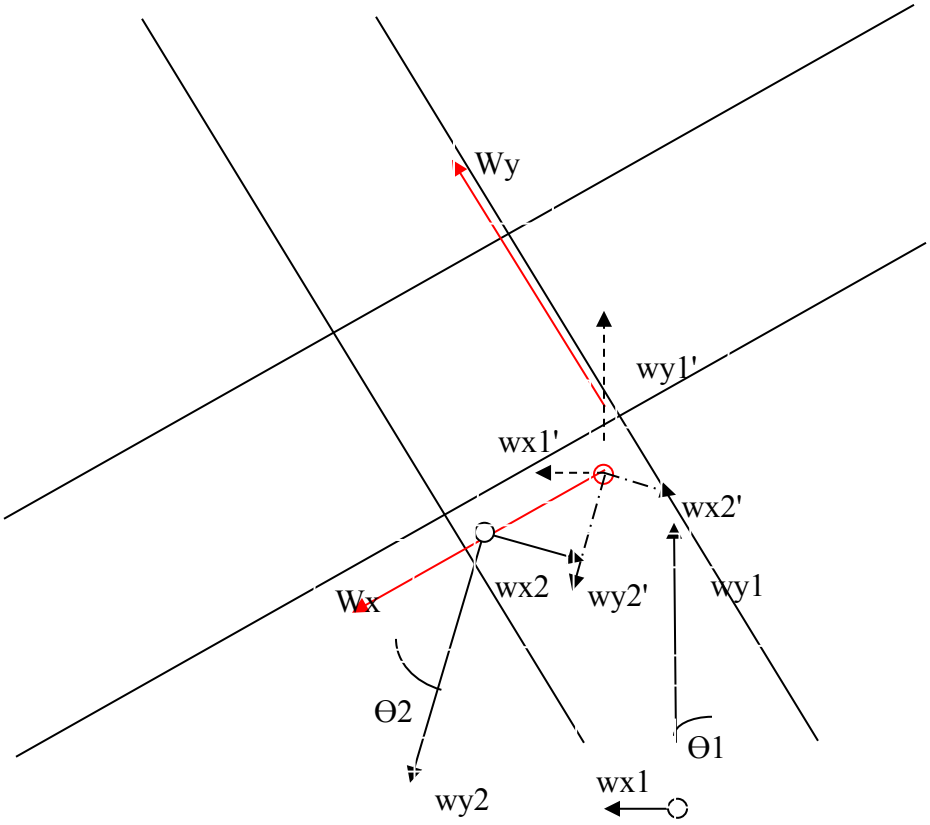


Figure 2.6: Bird's eye view of the W Arrowhead and Sawyer Avenue Intersection.

Similarly, we calibrate the other camera that captures the image on the right hand side of Figure 2.4, replicated here in a larger view in Figure 2.7. Note that i_{x2} and i_{y2} are defined to originate at the center of the image (or the principal point) and the world coordinate w_{x2} and w_{y2} to originate at the corresponding point in the road that projects to the principal point. The corresponding top view of w_{y2} and w_{x2} axis in the world coordinate is given in Figure 2.6 and it can be compared to the 2D image view in Figure 2.7. Note that in general case, the w_{y1} axis and w_{y2} axis for the two cameras will not be parallel to each other unless they are configured to be (refer to Figure 2.6). In Figure 2.7, we again select lines that are previously used to define the square pattern. In this case, we select the two parallel lines (formed by points "A" and "B", and "C" and "D") from the W Arrowhead Rd to derive the first vanishing point (u_{02}, v_{02}). The other two parallel lines (by points "A" and "D", and "B" and "C") from the Sawyer Avenue are used to derive the second vanishing point (u_{12}, v_{02}). The calibration results using equations (9)-(11) are shown below:

$$f_2=410,\ \varphi_2=0.66,\ \theta_2=0.85$$

Of these three calibration parameters, the pan angle θ_2 is shown in the Figure 2.6 and 2.7. Note that the pan angle is defined between the w_{y2} axis and the W_X axis (since the W_X axis is the one along which the parallel lines are selected to derive the first vanishing point u_{02}, v_{02}).

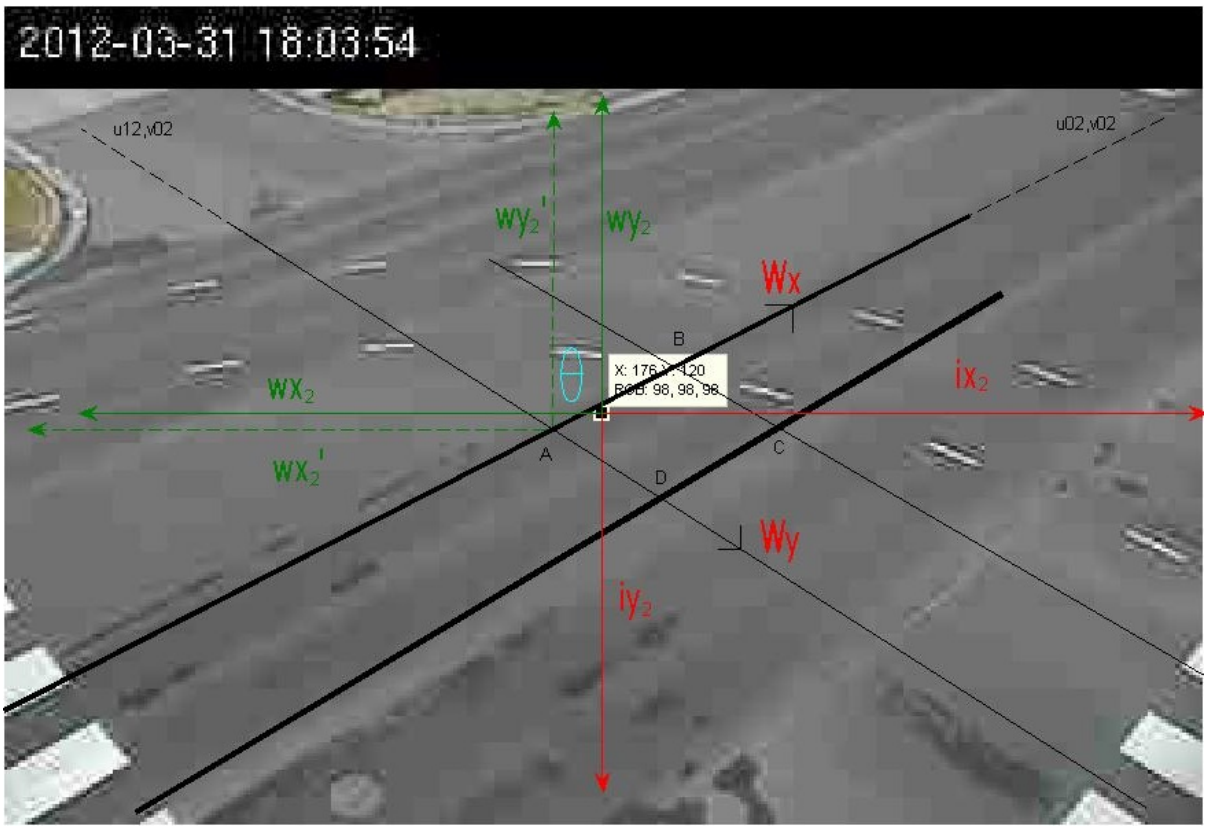


Figure 2.7: An enlarged view of the W Arrowhead and Sawyer Avenue Intersection from the second camera at SW corner.

Step 3: now we have separately calibrated each camera and the next step is to find the distance of coordinate shift from the current world origin of each camera to the common world origin defined from the square pattern in the first step. This is clearly seen from Figure 2.5 to Figure 2.7. Let us consider the first camera, for which the world axis w_{x1} and w_{y1}originates at the point in Figure 2.6 that projects to the principal point in Figure 2.5. Its current world origin needs to be shifted to the defined common world origin (denoted by point "A" in Figure 2.5 to 2.7) by a certain amount of distance on both axes. The shifted world coordinates are denoted by w'_{x1} and w'_{y1}. For the first camera, the coordinates (w_{x1} =0, w_{y1}=0) at its original world origin which are projected to its image coordinates (i_{x1}=0, i_{y1}=0) now become ($w'_{x1} = -sx_1, w'_{y1} = -sy_1$) in the shifted world coordinate, where sx_1 and sy_1 are the absolute shift amount. So the new equations that relates the shifted world coordinate (w'_{x1}, w'_{y1}) to the image coordinates (i_{x1}, i_{y1}) can be simply modified from the equation (5) and (6) to as follows:

$$i_{x1} = -\frac{f_1(w'_{x1}+sx_1)}{(w'_{y1}+sy_1).cos\varphi+\frac{h}{sin\varphi}} \quad (12)$$

$$i_{y1} = -\frac{f_1 sin\varphi.(w'_{y1}+sy_1)}{(w'_{y1}+sy_1).cos\varphi+\frac{h}{sin\varphi}} \quad (13)$$

We need to find out the shifts $sx_1$and sy_1 now. We propose to use a nonlinear optimization method for this purpose for two reasons. On one hand, we have at least 4 vertex from the square pattern to allow least-squared optimization based on projection errors of world coordinates to image coordinates. Second, such an optimization would not only solve sx_1 and sy_1, but also optimize focal length, pan angle and tilt angle for improved accuracy (as the later three parameters have been directly solved from equations before). Minimizing the project errors is usually used in this optimization[31]. To formulate the optimization problem, we first need to find the coordinates of the 4 vertex (i.e. points "A", "B", "C" and "D" in Figure 2.5 and 2.6) from the square pattern in the shifted world coordinate $w_{x1}^{'}$ and $w_{y1}^{'}$. Note that the four vertex going clockwise have well defined coordinates (0,0), (9,0),(9,9) and (0,9) in the common world coordinate W_X and W_Y denoted in red in Figure 2.5 and 2.6. To transform these coordinates to those in the shifted world coordinate w_{x1}' and w_{y1}' for the first camera, we simply rotate the common world axes clockwise by an angle θ_1, which is actually one of the variables to be post-optimized, as an initial value is already available from above. Such a transformation can be easily solved as shown below:

$$\begin{bmatrix} w_{y1}' \\ w_{x1}' \end{bmatrix} = \begin{bmatrix} \cos(\frac{\pi}{2} - \theta_1) & \sin(\frac{\pi}{2} - \theta_1) \\ -sin(\frac{\pi}{2} - \theta_1) & \cos(\frac{\pi}{2} - \theta_1) \end{bmatrix} \begin{bmatrix} -W_X \\ W_Y \end{bmatrix} \tag{14}$$

Then, we obtain

$$w_{x1}^{'} = W_X \cos\theta_1 + W_Y sin\theta_1 \tag{15}$$

$$w_{y1}^{'} = -W_X sin\theta_1 + W_Y \cos\theta_1 \tag{16}$$

Note that in the above equation w_{y1}' appears in the upper row is due to the rotation on the right. With coordinates of the 4 vertex in the shifted world coordinate $w_{x1}^{'}$ and $w_{y1}^{'}$, we then find the coordinates of the 4 vertex in the image coordinate i_{x1} and i_{y1}. This can be simply obtained by clicking the data cursor on the vertex in the image. However, just be careful that these image coordinates obtained in the MATLAB software [40] need to be shifted to be those in the i_{x1} and i_{y1} axis by subtracting 352/2=176 and 240/2=120. Now, the post-optimization problem can be formulated as follows:

$$minimize\ G(f, \theta, \varphi, sx, sy) =$$

$$\sum_{K=1}^{4} (i_{xK} - C_{xK})^2 + \left(i_{yK} - C_{yK}\right)^2$$

$$subject\ to\ f_{LB} \le f \le f_{UB}$$
$$\boldsymbol{\theta_{LB}} \le \theta \le \boldsymbol{\theta_{UB}}$$
$$\boldsymbol{\varphi_{LB}} \le \varphi \le \boldsymbol{\varphi_{UB}}$$
$$\boldsymbol{sx_{LB}} \le sx \le \boldsymbol{sx_{UB}}$$
$$\boldsymbol{sy_{LB}} \le sy \le \boldsymbol{sy_{UB}} \tag{17}$$

where $\boldsymbol{i_{xK}}$, $\boldsymbol{i_{yK}}$ are from equation (13) and (14), $\boldsymbol{C_{xK}}$, $\boldsymbol{C_{xK}}$ are the coordinates of the four vertex in the image coordinate $\boldsymbol{i_{x1}}$ and $\boldsymbol{i_{y1}}$, and *K* the index for the number of vertex used. Finally note that each variable to be optimized has an upper and lower bound. In our project, we are giving a very large range for optimization and we have found the optimization has been very robust and fast.

The above described process is then repeated for the second camera. Much of the processing steps remain the same, except that we need now to transform the coordinates of the 4 vertex in the common world coordinate W_X and W_Y denoted in red in Figure 2.6 and 2.7 to the shifted coordinate for the second camera w'_{x2} and w'_{y2} (note that w'_{x2} and w'_{y2} are obtained by shifting the original world coordinate w_{x2} and w_{y2} for the second camera by distances sx_2 and sy_2) for the second camera and then we need to find the image coordinates for the 4 vertex appearing in the image taken by the second camera as shown in Figure 2.7. The axis rotation in this case is

$$\begin{bmatrix} w'_{x2} \\ -w'_{y2} \end{bmatrix} = \begin{bmatrix} \cos[-\left(\frac{\pi}{2} - \theta_2\right)] & \sin[-\left(\frac{\pi}{2} - \theta_2\right)] \\ -\sin[-\left(\frac{\pi}{2} - \theta_2\right)] & \cos[-\left(\frac{\pi}{2} - \theta_2\right)] \end{bmatrix} \begin{bmatrix} -W_X \\ W_Y \end{bmatrix} \tag{18}$$

Then, we obtain

$$w'_{x2} = -W_X sin\theta_2 - W_Y cos\theta_2 \tag{19}$$

$$w'_{y2} = W_X \cos\theta_2 - W_Y sin\theta_2 \tag{20}$$

Now, a similar post-optimization as in (18) could be formulated for the second camera, which gives optimized f_2, θ_2, φ_2, sx_2, and sy_2. Note that in the problem formulation in (18), only 4 vertex from the square pattern is used for minimum projection error. If more primitives are available, such as more vertex or distances [31], are available and used, then more robust results could be expected.

2.3 Experiment Results

In this Section, we report the experimental results for the real-world intersection considered in this project. We applied the extended calibration method discussed in Section 2.2. In Section 2.2, we reported the initial calibration results for focal length, pan angle and tilt angle for the first camera as f_1=310, φ_1=0.773, θ_1=0.61. Then, the following lower and upper bounds were given for these parameters:

$$\begin{gathered} \boldsymbol{100 \le f_1 \le 800} \\ \boldsymbol{0.3 \le \theta_1 \le 0.9} \\ \boldsymbol{0.5 \le \varphi_1 \le 1.1} \\ \boldsymbol{0 \le sx_1 \le 20} \\ \boldsymbol{0 \le sy_1 \le 20} \end{gathered}$$

Note that we did not specify here a rule for setting the lower and upper bounds. A rule of thumb is to give a very large range so that there is a large design space to be explored. For example, for

f_1, φ_1, θ_1, we could simply set the lower bound to be 50% of the nominal value and upper bound be twice the nominal value. For sx_1, sy_1, note that it can be inspected from Figure 2.5 and 2.6 that it should be positive, and we simply set a large enough number for them, 20 feet in this case. Note that in this project, all real-world distances are measured in feet, such as the coordinates of the 4 vertex from the square pattern and the camera height *h*=39 feet. However, the bounds for sx_1, sy_1 can be simply set larger and the optimization results are very stable. With the camera height at 39 feet, the optimization gives the following results:

$$f_1 = \mathbf{360.50}$$
$$\theta_1 = \mathbf{0.626}$$
$$\varphi_1 = \mathbf{0.547}$$
$$sx_1 = \mathbf{1.61}$$
$$sy_1 = \mathbf{7.14}$$

The cost function value, which is the minimized projection error, was only 9.86 (in squared pixels), which means that on average each coordinate out of the 8 coordinates for the 4 vertex of the square pattern has only a projection error of 1.1 pixel in the image. A brief analysis of the optimization results shows that the tilt angle has been changed much from its initial value φ_1. Finally, note that the optimization was performed in MATLAB [40] and each runs took less than 10 seconds.

For the second camera, the initial values are f_2=410, φ_2=0.66, θ_2=0.85. Then, we gave the following bounds for optimization:

$$\mathbf{100} \le f_2 \le \mathbf{800}$$
$$\mathbf{0.5} \le \theta_2 \le \mathbf{1.3}$$
$$\mathbf{0.5} \le \varphi_2 \le \mathbf{0.9}$$
$$\mathbf{0} \le sx_2 \le \mathbf{20}$$
$$\mathbf{-10} \le sy_2 \le \mathbf{10}$$

Note that in this case, the coordinate shift sx_2 is positive, but it is hard to tell sy_2 in Figure 2.7 due to the limitation of the 2D view of a 3D scene, therefore a different bound was set for sy_2 than sx_2. Again, all the five parameters can be given a larger range. With camera height *h*=39 feet, the calibration results for the second camera are shown below:

$$f_2 = \mathbf{339.46}$$
$$\theta_2 = \mathbf{0.862}$$
$$\varphi_2 = \mathbf{0.626}$$
$$sx_2 = \mathbf{2.61}$$
$$sy_2 = \mathbf{-1.06}$$

The minimized cost function value was only 8.56 (in squared pixels), which means that on average the coordinate of each vertex has only a projection error of 1.0 pixel. An analysis of the results shows that the optimized f_2, φ_2, θ_2 are fairly close to their initial values in this case. Also, note that it turns out that the coordinate shift sy_2 is negative in this case. Besides, note that the optimized tilt angle φ_2 turns out to be the same as the optimized φ_1. Again, each optimization run takes less than a few seconds. Finally, note that there only needs to be one

optimization run instead of two and the only change is to combine all ten calibration variables into one formulation. In the latter case, the calibration results would not change because the five variables for the first camera are independent of those for the second camera.

In Table 2.2, we evaluate accuracy of the camera calibration results by comparing estimated distance from the calibration results to measured distance (or ground truth) for a few line segments from the square pattern that appears in both Figure 2.5 and 2.7. It can be found that the average accuracy is about 93.5%.

Table 2.2: Comparison of estimated distances from the calibration results to ground truth distances (coordinates of A=(168,95), B=(126,110), C=(109,90), D=(148,78) in Figure 2.5 and A=(161,125), B=(198,106), C=(228,124) and D=(193,146) in Figure 2.7) .

	Figure 2.5			**Figure 2.7**		
Distance	Estimated (ft)	Measured (ft)	Accuracy	Estimated (ft)	Measured (ft)	Accuracy
AB	9.64	9.0	93%	9.96	9.0	89%
BC	8.60	9.0	96%	8.33	9.0	93%
CD	9.32	9.0	96%	9.60	9.0	93%
DA	8.38	9.0	93%	8.58	9.0	95%

As will be shown in later Chapters, the accuracy of camera calibration may be one of the key factors for successful tracking of vehicles in a multiple-camera-based system. In the above calibration process, selecting more primitives for the purpose of optimization of calibration parameters would further improve accuracy. Also, based on our experience, we recommend the following empirical rules to achieve accurate calibration results: (1) the cameras are preferred to be configured such that the tilt angle of each camera is relatively large so that the images have enough depth; (2) the resolution of the images is preferred to be large, because rounding of the pixels of the selected points would statistically lead to less accuracy loss.

In the above experiments, the parallel lines have been manually selected for initial calibration of each camera. As in previous works for traffic lane extraction [35-36], these lines can also be automatically extracted. First, we convert the color image to grayscale and perform Sobel edge detection. The result is a binary image with edge points. With these edge points which may not all belong to the traffic lanes, we used a voting method to derive vanishing points. Once the initial solution is obtained from these vanishing points, the post-optimization can then follow. However, it should be noted that camera calibration in the automatic mode has the disadvantage that unclear (or poorly legible) traffic lanes may have significant negative effect on the accuracy of calibration. In case the traffic lanes are not clear in the image (as the landmark of the traffic lanes is not clear in real-world traffic scenes), one may track the vehicles to obtain the vehicle trajectories, which can then be regarded as the equivalents of traffic lanes. This idea has been also used in previous works on camera calibration for roadway scenes when parallel traffic lanes are not clear in the image [28][35-36]. In this case, the initial solution obtained from the close-form equations may not be very reliable and post-optimization could help improve the calibration accuracy.

In this project, we have target the intersection traffic scene. In general, such an intersection scene is cross-section one. If it is not in some cases, then the angle between the traffic lanes from both ways may be solved from post-optimization or simply assumed known by practical measurements. We will not go into detail for these special cases.

Finally, we need to note that the calibration results are obtained when considering projecting the pixels from the world coordinate to the corresponding image coordinate. Referring back to Figure 2.5 to 2.7, for the first camera, the image coordinate is (i_{x1}, i_{y1}) and the world coordinate (w'_{x1}, w'_{y1}) and for the second camera the image coordinate (i_{x2}, i_{y2}) and the world coordinate (w'_{x2}, w'_{y2}). To project the image coordinates (i_{x1}, i_{y1}) and (i_{x2}, i_{y2}) back to the common world coordinate (W_X, W_Y), a proper rotation of (w'_{x1}, w'_{y1}) and (w'_{x2}, w'_{y2}) to (W_X, W_Y) is required. But now the rotations can be easily solved from the above equations (16-17) and (20-21). In the next Chapter, we will be using these rotations to inverse-project objects in two images to the common world coordinate (W_X, W_Y).

2.4 Summary and Conclusion

In this Chapter, we considered calibration of multiple cameras for a typical intersection traffic scene. We first reviewed previous work on camera calibration of a single camera for regular roadway scenes, such as highways and intersections. A popular method for fast and convenient calibration is based on vanishing points derived from parallel lines. We then proposed to extend the previous method to accommodate calibration of multiple cameras. In the extended method, the first step is to define a square or rectangular pattern from two pair of parallel lines in the intersection such that a common world coordinate is defined for both cameras. Then, we calibrated each camera separately using the traditional method based on vanishing points. In this step, the image coordinates of each camera are related to the camera's own world coordinate, not the common world coordinate defined in the first step. In the next step, we proposed to post-optimize the camera parameters using the vertex of the square pattern. Such a post-optimization not only refines the solved focal length, pan angle and tilt angle from the previous step, but also find the coordinate shifts in order to transform each camera's world coordinate to the common world coordinate. Our experiment results have shown that such a post-optimization is very fast and robust.

The calibration results can then be used further to estimate vehicle speed and facilitate vehicle tracking to derive vehicle trajectory, which will be discussed in detail in the next Chapter.

Chapter 3. Multiple-Camera-Based Vehicle Tracking System

Following camera calibration, vehicle tracking is the next main module in the overall traffic performance measurement system developed in this project. As mentioned in Chapter 1, vision-based vehicle tracking has been widely used in recent years due to its advantage to provide the most comprehensive information about the vehicles compared to loop-detectors, radars or other sensor-based systems. Though videos captured could be manually inspected for traffic performance measurements, it is very costly and laborious. Recent research work focuses on automatic collection of traffic performance measurements from video processing. There are relatively mature vision-based vehicle tracking technologies available for highways, but vision-based vehicle tracking for intersections (and roundabouts) presents more challenges due to more complex vehicle behavior, such as turning, stopping, accelerating/de-accelerating at such traffic scenes and more complex traffic performance measurements, such as waiting time and accepted/rejected gaps. Also, most vision-based vehicle tracking systems reported in literature use a single camera, whereas multiple-camera-based vehicle tracking systems are emerging.

Vehicle tracking is probably the most important and also most challenging part of a multiple-camera tracking system. This Chapter focuses on the vehicle tracking module in our system as mentioned in Chapter 1. The designed vehicle tracking module takes the captured images from multiple cameras as inputs and outputs vehicle trajectories. It has two major steps of processing. First, the system uses an enhanced Mixture of Gaussian algorithm with shaking removal for video segmentation, which can cope with repeated and sudden camera displacements. Next, region based tracking is employed to track the vehicles and derive the complete vehicle trajectories. In the second step, many sub-processing steps are followed, which will be discussed in detail later in this Chapter. The resulting vehicle trajectory of each individual vehicle gives the position, size, shape and speed of the vehicle at each time moment. Finally, a data mining algorithm is used to automatically extract the interested traffic data from the vehicle trajectories and this is the topic of Chapter 4.

The rest of the Chapter is organized as follows. In Section 3.1, we give an overview of the designed vehicle tracking module. In Section 3.2, we discuss the vehicle segmentation method used in the designed vehicle tracking module, followed by the vehicle tracking algorithm in Section 3.3. Finally, some experimental results are shown in Section 3.4.

3.1 System Overview

The designed vehicle tracking module is implemented as software that runs on a regular PC. It incorporates powerful image processing algorithms to allow accurate vehicle tracking. The vehicle tracking system has mainly two processing steps, vehicle segmentation and vehicle tracking. Figure 3.1 shows the general processing flow for the multiple-camera vehicle tracking system. As can be seen, the second step can be further divided into many sub-steps, which will be all detailed in this Chapter.

Vehicle segmentation is the first processing step in the vehicle tracking system. The purpose of this step is to identify potential objects in the images from the multiple cameras. Then, these potential objects go through some image processing, such as filtering, dilation and hole filling, to

cope with poor detection in many cases. In the next step, we perform an inverse projection to project the segmented objects in the image coordinates to the world coordinates to facilitate vehicle tracking. Recall that this can be done since camera perspective information required for inverse projection has been obtained from camera calibration in Chapter 2. After this step, potential objects are extracted from the word coordinate and only valid objects that meet certain requirements are claimed. Subsequently, valid objects from each camera are checked for their position, size, shape and so on to see whether objects from different cameras may correspond to the same object in the world. This step is basically correspondence initialization as mentioned in Figure 1.1 in Chapter 1. The result of this check is a complex overlap relation matrix between objects from different cameras. In the following step, potential vehicles are created for the current image frame based on the derived overlap relation matrix. Then, potential vehicles are checked with current vehicles, which are those vehicles tracked up to the previous image frame, to create a vehicle overlap relation matrix, which is then used in the final step of vehicle updating to update current vehicles with potential vehicles such that each current vehicle is tracked to the current image frame.

The above described process goes iteratively until the last frame of images is reached. The final results of the vehicle tracking system are vehicle trajectories in the world coordinate for all tracked vehicles. In the following, we will detail on each processing step shown in Figure 3.1.

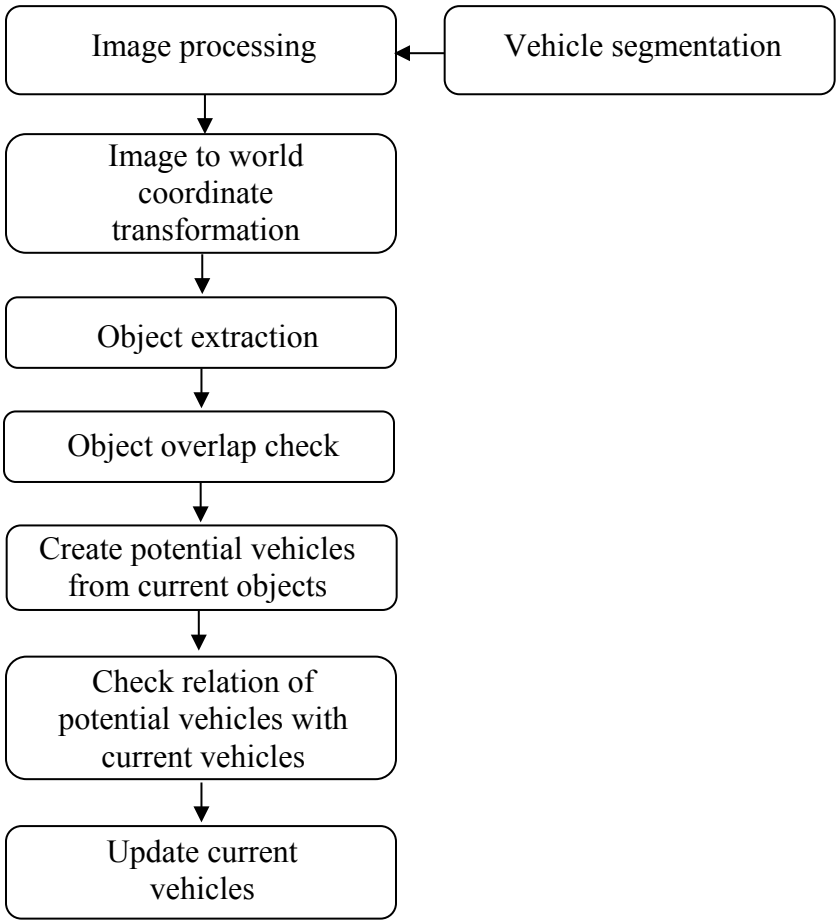


Figure 3.1: Processing flow of the designed vehicle tracking module.

3.2 Vehicle Segmentation

For vision-based traffic surveillance or vehicle tracking systems, vehicle segmentation from the captured images is the first step. The desired outputs of this step are all vehicles that appear in images as can be inspected by humans. But for an automatic vehicle tracking system that runs on personal computers, it is clear that such a system can never reach the intelligence of humans on identifying vehicles so that the accuracy of vehicle segmentation is always inferior to that from human inspection. In a vehicle tracking system, such as the one targeted in this project, vehicle segmentation is arguable the most important processing step in the whole vehicle tracking module. The better the result of this step is, the easier and more accurate vehicle tracking is.

We first briefly review some previous methods for vehicle segmentation and then introduce the Mixture-of-Gaussians method that is adopted and revised in this project.

3.1.1 Existing Approaches

There are many existing methods for vehicle segmentation, such as background subtraction, running average, texture-based method and Eigen-background method [41-47].

In [41-43], the main idea of vehicle segmentation is calculating background based on the history of pixel values. The average background simply takes the average of N previous frames, and when a new frame comes, it will replace the oldest frame. Other similar approaches take median or mode value instead of average. Background of those approaches can be defined as

$$B_N = \text{average, median or mode } \{I_k | \; k = 0, 1 \ldots N\}$$

where I_k is the intensity at frame k. Those methods run very fast, but they also require a lot of memories: N * size (frame). To reduce memory consumption, the running average method simply adapts background based on incoming frame with a certain learning rate:

$$B_N = (1 - \alpha) * B_{N-1} + \alpha * I_N$$

where B_{N-1} is the background at frame N-1, I_N is pixel intensity value at frame N, and α a learning rate. When a new frame comes, we take α percent of its intensity and (1-α) percent of existing background as the new background. A large value of α means that the model will update faster, typically α= 5% (0.05).

There is also a combined method like in [50], choosing between running mode and running average base on a scoreboard algorithm. The mechanism to update background can be modified from blind update to selective update:

$$B_N = (1 - \alpha) * B_{N-1} + \alpha * I_N * M$$

with M=1 if the pixel is classified as background, and 0 if foreground.

[47] uses another advanced approach, eigenspace, to form the background. M eigenvectors corresponding to M largest eigenvalues are kept when applying Principle Component Analysis (PCA) on a sequence of N images. Background pixels, which appear frequently in frames, will significantly contribute to this model while moving objects do not.

Texture property of the image could also be used for segmentation. [45] uses autocorrelation difference between two image blocks to compare their similarities. This method also requires an empirical threshold which is not explicit and varies through videos. Moreover, the computational workload for autocorrelation is very high [44-45].

The above-discussed methods have some common disadvantages. One main disadvantage of these methods is that they do not cope well with camera shaking and background movements, which happen quite often in practice.

3.1.2 The Mixture-of-Gaussian Approach

Another method to build background models is to model each pixel with a Gaussian distribution (MoG) [48-49]. This method is adopted in our project and the main advantage of the MoG method is its capability to cope with repeated camera shaking. To further suppress false detections from sudden camera shaking, we also proposed a shaking-removal algorithm to refine the segmented objects.

In MoG, the variation of each pixel across time is modeled by a mixture of 5 Gaussian distributions for each color (R, G, B) and each distribution has its own mean, standard deviation, and weight [48-49]. The MoG actually models both foreground and background. The first B (B<=5) distributions are chosen as the background model. A threshold T is defined to represent the portion of the data that should be accounted for the background model:

$$Background = min_B\left(\sum_{k=1}^{B} w_k > T\right)$$

where w_k is the weight of distribution k (k=1,2,3,4,5). We normalize the weights such that:

$$\sum_{k=1}^{5} w_k = 1$$

For a new image frame, each incoming pixel will be subject to a matching test to see whether it belongs to any existing distribution of that pixel or not. We apply confidence intervals for matching. We choose the confidence interval to be 98%, so an incoming pixel X will belong to the distribution D(μ,σ) if:

$$p = \frac{|X - \mu|}{\sigma} \leq 2.5$$

Background model update is an important step to keep the background models up-to-date with environment changes such as illumination and scene changes. If none of 5 Gaussian distributions matches the incoming pixel, the update will simply replace the distribution with the lowest weight by a new distribution with the same weight, a large standard deviation and the mean equal to the incoming pixel. If there is one distribution matching the pixel, its weight, mean and standard deviation will be updated as follows [48]:

$$w_k = w_{k-1} + \alpha * (1 - w_{k-1})$$

$$\mu_t = (1 - \alpha)\mu_{t-1} + \alpha * X_t$$

$$\sigma_t^2 = (1 - \alpha)\sigma_{t-1}^2 + \alpha * (X_t - \mu_t)^2$$

whereas the mean and standard deviation for other distributions are kept the same except their weights which are reduced as follows:

$$w_k = w_{k-1} + \alpha * w_{k-1}$$

where α is a learning rate. In case there are more than one distributions that match the pixel, the distribution which has smallest p will be chosen for update.

MoG can model fast illumination changes in the scene. Another advantage of MoG is its robustness against repeated camera shaking due to its multi-model modeling capability. Repeated camera shaking is one very practical problem that affects tracking accuracy. As the obtained images capture slightly different world views due to camera shaking, large regions of noisy strips could be segmented from the image using traditional background modeling approaches [41-47], which could severely affect the accuracy of object extraction. MoG can fight with repeated camera shaking very well, as the pixel changes caused by repeated camera shaking are recorded in the background distributions and are therefore not identified as false segmentation regions any more. Experiment results on this can be found in a previous report [14].

On the other hand, when the camera shakes to specific positions which have not yet been observed before, many background pixels may get pixel values not modeled before and be falsely classified as foreground while they are actually from nearby background. Those false detections could also happen if the camera shakes only occasionally after some time of being still. In that case, even the pixel values are modeled in some distributions before, but their rare observations make their weights too small to be considered in background models. Therefore, to suppress these false detections and further refine the detection results, we propose a shaking-removal algorithm after MoG segmentation. This is based on the observation that the false detection regions will have a high probability to be a part of the background distributions at its original locations. Hence, we can decide whether a detected region is caused by a real foreground object or the background by considering the background distributions in a small neighborhood of the detection region. If the detection pixel matches with the background distributions of pixels in the neighborhood, it is highly possible that this detection is false caused by camera shaking. When considering color images with the RGB (Red, Green, Blue) space, this technique proves to be effective as the probability of mismatching is small [49]. In our implementation, we choose a square 5x5 window for shaking removal.

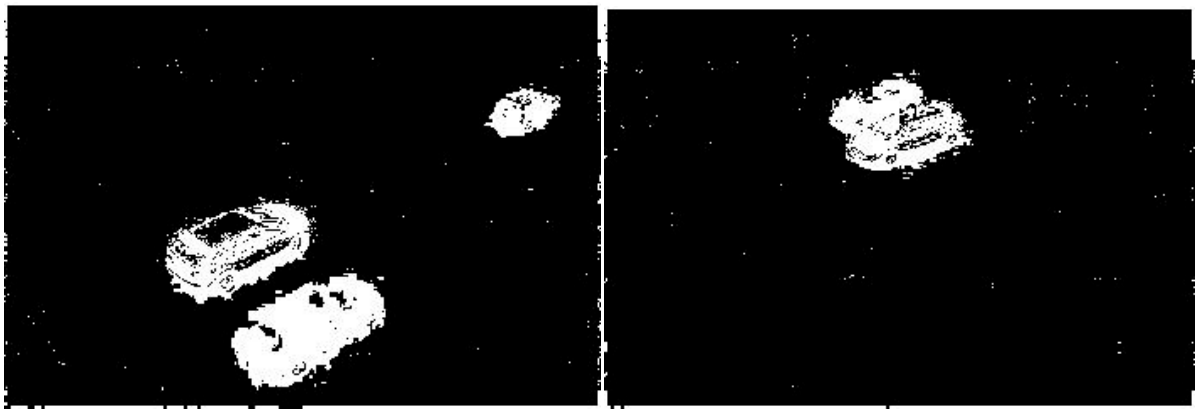
The combined MoG modeling of the background and the shaking-removal algorithm turn out to be very valuable as video cameras are subject to repeated and occasional shakings in practice, which would significantly affect segmentation quality if not treated. The proposed method is very general and does not involve manual intervention or rely on any prior scene knowledge. The pseudo code for the MoG method could be found in Table 3.1.

Table 3.1: Pseudo code for the MoG algorithm for vehicle segmentation.

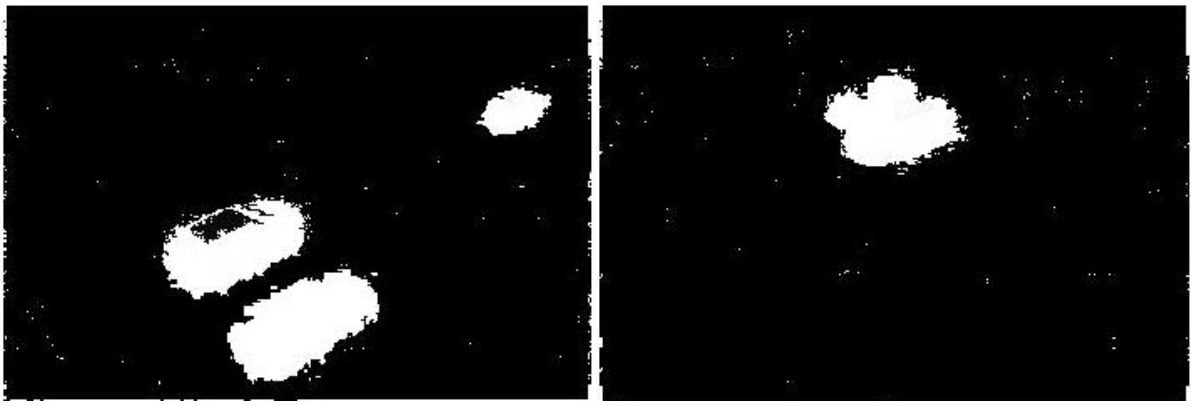
```
%Initialize
read the first frame from the video to get the size of images
create a matrix M with the same size as the frame
for all elements in M
   for all distributions
      mean=initial mean;
      variance=initial variance;
   end for
end for
%Processing
for all frames of the video
   %segmenting
   for all pixel in the frame
      for all distributions
         t=sum((mean-newcoming_pixel)^2-(2.5*var)^2);%matching
         if t<0
            the new pixel matches one or more distributions;
         end if
      end for
      if there is no match
         the pixel is recognize as foreground;
      end if
      if there is one or more matches
         find the match that has smallest t value;
         check whether that match belongs to the background model;
         if no
            set pixel as foreground;
         end if
      end if
   end for
   %shaking remove
   for all pixel in the frame
      if foreground
         for neighbor pixels
            take the distribution that has largest weight;
            t=sum((mean-newcoming_pixel)^2-(2.5*var)^2);%matching
            if t<0
               match found, reset pixel as background;
            end if
         end for
      end if
   end for
   %updating
   for all pixel in the frame
      update mean, variance and weight;
   end for
end for
```

The resulting outputs from vehicle segmentation are binary pixels that could potentially form valid objects (or sometimes referred to as blobs in image processing terminology). Figure 3.2(a) shows two images from two cameras respectively for the traffic intersection mentioned in Chapter 2 and 3.2(b) shows outputs of the segmented images. Note that it seems that two cameras are not synchronized from the time stamp, but it was found that there was an internal 2-second mismatch between the two cameras.

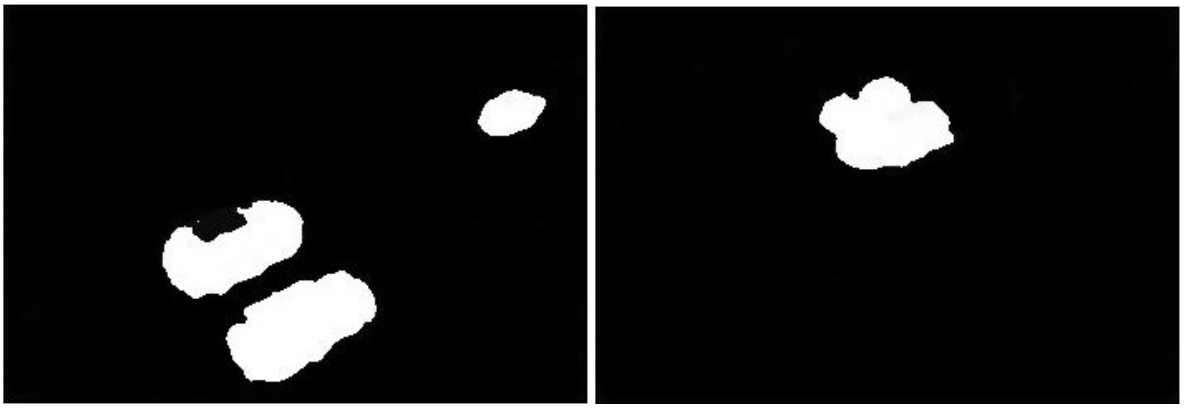
(a)



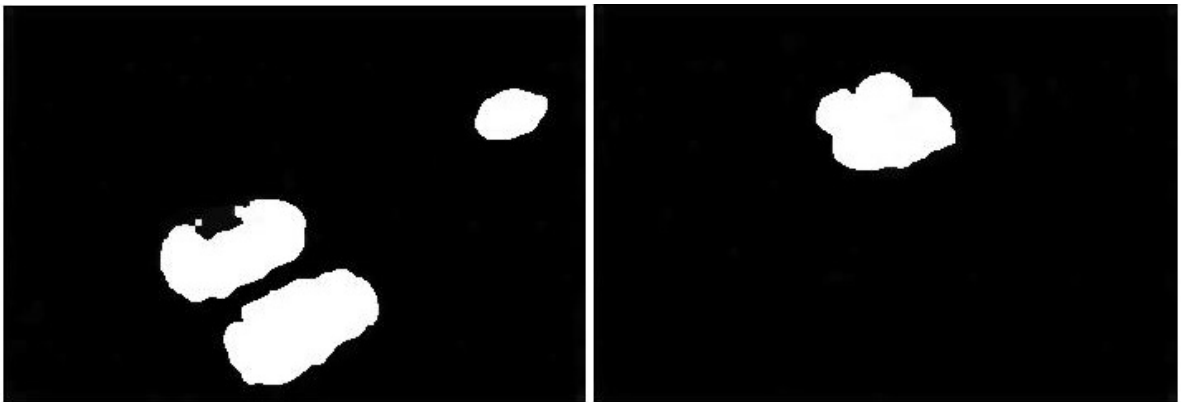
(b)



(c)



(d)



(e)

Figure 3.2: Two captured images from two cameras and their segmented outputs.

3.3 Vehicle Tracking

In this Section, we describe the second processing step of the vehicle tracking module as shown in Figure 3.1 in detail. After segmentation, tracking is the next step to detect and track vehicles in each time frame. The goal is to obtain the complete vehicle trajectory, as it gives comprehensive information about the state of the vehicle at each time moment, which is more accurate for traffic performance measurement than without trajectories. A number of methods exist for tracking. Some popular approaches are region-based tracking [5], contour-based tracking [50], model-based tracking [51], and feature-based tracking [52]. Our approach is region-based tracking. In the following, each sub-section presents the algorithm and implementation details of each sub-step in Figure 3.1.

3.3.1 Image Processing for Object Refinement

As it can be seen from Figure 3.1(a), the vehicles from both images are clearly identified with human inspection. If given only Figure 3.1(b) and knowing that vehicles are the interested objects, then human inspection would be able to identify vehicles easily in most cases. Note that a pixel in black would mean a background pixel and a pixel in white mean a foreground pixel, in

other words, an object pixel. While a human glance over the pictures gives a qualitative sense that even the black and white pixels represent the actual vehicles reasonably well in this particular example, however, they do not in many other cases. Also, even they do, it may still be a challenge for computes to correctly recognize the vehicles. For best recognition accuracy, at least one of the factors would be how to refine the black and white pixels for best representation of the objects. Therefore, this sub-step of processing is to apply some image processing techniques to refine the segmented images, from which valid objects need to be extracted and claimed.

While there are hardly rigorous mathematical proofs that these following techniques would help refine the segmented images and there is no guarantee that they will not make negative effect, they are used mostly based on common sense observation. One image processing technique is hole filling [39], which is to fill the black holes in an enclosed region of white pixels. The underlying idea behind this action is to make up missing foreground detections and make real objects more solid. As can be seen in Figure 3.2(b), the vehicle in grey in Figure 3.2(a) loses its middle and right white pixels (from the left camera view) and middle and left white pixels (from the right camera view). These missing white pixels obviously make the object's representation of a vehicle incomplete and this action of hole filling can help in this case. The result of hole filling can be seen in Figure 3.2(c).

The next action of image processing is image filtering [39], which is used to filter out noisy detections, for instance the so called salt and pepper noise that may be widespread all over the segmented image. Referring to Figure 3.2(b), there are quite some white pixels that should have been segmented to be black ones because the world locations of those white pixels are actually background. These noisy detections can be easily removed using the filtering technique. The result of filtering can be seen in Figure 3.2(d). One of the control parameters here is the threshold size for filtering, which has a tradeoff. If it is set too large, then there is some risk that valid regions of white pixels that actually represent valid objects or part of the valid objects may be removed. On the other hand, if too small, some relatively large regions of noisy detection may be retained, which may be mis-claimed as valid objects. In either case, the negative impact is that it may complicate the vehicle tracking later.

After filtering, another useful action of image processing is image dilation [39], which is used to dilate white regions of potential objects around their contours. Dilation mainly helps when for example a single vehicle is represented by an overall collection of several or many dis-connected regions of white pixels due to poor detection or segmentation, and in some cases image dilation may re-join these separate regions to simplify vehicle tracking later. The result of dilation can be seen in Figure 3.2(e). In this specific case, the dilation simply make the blob fatter. Similarly, there may be negative impact associated with image dilation as well, as it may accidentally join regions that belong to different vehicles, creating difficulty for vehicle tracking.

It should be noted the order of the actions discussed above could be important, but the specific order of actions discussed above is not the only way. It should also be noted that there may be many other actions of image processing that can help improve refine the objects and facilitate vehicle tracking. For example, there are techniques to remove shadow, which can be added into the processing flow if shadow exists. Besides, more human intelligence can be added as well, for example some intelligent algorithms for proper merging and splitting of blobs considering the

nature of the objects of interest, such as size, shape, color etc. Finally, it should be emphasized again that segmentation results play a very important role in the accuracy of the overall vehicle tracking module.

3.3.2 Image to World Coordinate Transformation

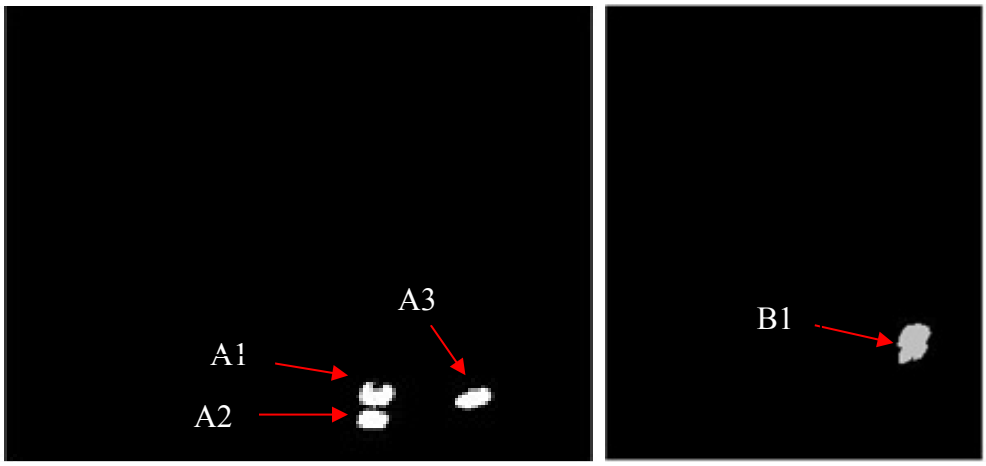
After object refinement discussed in the last sub-section, we now proceed to the next sub-step of processing in vehicle tracking, which is image to world coordinate transformation. Recall in Chapter 2, we find out the projection relation between the common world coordinates (wx,wy) and image coordinates (ix1,iy1) and (ix2,iy2) for both cameras. Now, this sub-step of processing basically inversely projects the potential objects from the 2D image coordinate to the 3D world coordinate, which would allow size and shape analysis of the object in the real world. Obviously, the sub-step requires the camera calibration information, which has been already obtained from Chapter 2. So, given the focal length, pan angle, tilt angle, camera height and coordinate shifts, we transform (ix1,iy1) back to (wx,wy) and if the segmentation results at pixel (ix1,iy1) is foreground of a binary value of 1 (or shown as white in Figure 3.2), it would be foreground of a binary value of 1 at the corresponding world coordinate (wx,wy). Similarly, if the segmentation results is background of a binary value of 0 (or shown as black in Figure 3.2), it would be background at the corresponding world coordinate.

Also, a special action here is that we can find out the view boundary in the world coordinate by inverse projection of all edge pixels of the images. There are two practical issues here that are worth notes. First, when projecting, the projected world coordinate (wx,wy) corresponding to the image coordinate (ix1,iy1) may become negative (one of them or both), which can not be used for index of arrays in MATLAB [40], therefore a positive coordinate offset could be added to address this issue. Second, a similar issue is the projected world coordinate may not be integers to be index of arrays, and rounding may be used to make them integers, however this may raise a new potential concern that the projected object in the world coordinate may appear dis-connected or broken even though it is a solid object in the image coordinate. This later issue is only occasionally observed in our project, and we currently use hole filling and image dilation (as discussed in previous sub-section) to re-solidify the object in the world coordinates if it happens.

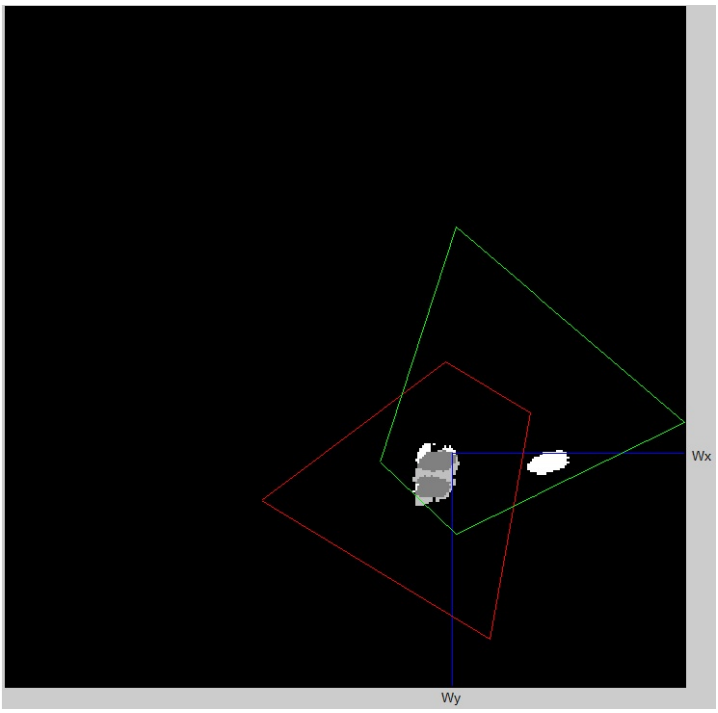
Table 3.2: Pseudo code for image to world coordinate transformation.

```
%Initialize
read in calibration parameters for all cameras
set coordinate offset
for all cameras
   for all pixels
      transform image coordinates to world coordinates;
      if background at current pixel
         set background in the corresponding world coordinates;
      else
         set foreground in the corresponding world coordinates;
      end if
   end for
end for
%Processing
define a matrix;
for all pixels
   copy the transformed results from both cameras for that pixel;
   if both results are background
      this is the overlap between two camera views;
   end if
end for
```

A pseudo-code for the image to world coordinate transformation is shown in Table 3.2. Figure 3.3(a) and Figure 3.3(b) shows the transformed objects from Figure 3.2(e). Note that in Figure 3.3(a), the objects from the left hand side of Figure 3.2(e) are shown in white and in Figure 3.3(b), those from right hand side of Figure 3.2(e) shown in light gray. Figure 3.3(c) shows the overlaid view of the objects from both cameras. The dark gray are used to show the overlap between the objects from both cameras. Also, note that the red and green lines are used to indicate the view boundary of the two cameras.



(a) (b)



(c)

Figure 3.3: The transformed objects in the world coordinate.

3.3.3 Object Extraction

After objects appearing initially in the images are projected back to the world coordinate, objects can now be extracted by the connected component algorithm [5]. To well represent objects, each object is characterized by both a rectangular box with width and length to bind the object, and a contour with the detailed shape of the object. An object with the above description would be possibly part of a vehicle, one vehicle or more vehicles. When creating an object, there are a number of conditions to check to validate the object. First of all, we set up a size threshold for the object, and the object is not validated if its size is smaller than the threshold. Such a step is empirical, as it may have both positive and negative consequences. On the positive side, it helps filter out truly invalid objects such as objects due to relatively large noise and poor detection and reduce the number of objects, which may greatly simplify vehicle tracking in later steps. On the negative side, those smaller-size objects might be part of a vehicle or multiple vehicles, and removing them may impair complete and correct characterization of vehicles to be created later. In our project, we used the threshold size of 24 to 28 $feet^2$.

Another condition that we currently check in our project is whether an object is just entering the view of one of the cameras. In this case, we would not count that object as a valid one to simply vehicle tracking in later steps. In other words, only after an object or vehicle is completely inside the camera view can we claim a valid object. Such a check can be performed by checking whether some pixels of the object intersect with the view lines (refer to Figure 3.3 for the view lines). Like the previous check, this one is empirical in order to achieve a good tradeoff between vehicle tracking complexity and accuracy. It is worth more research on whether there should be

more or less intelligent checks and their impact on the performance of the whole vehicle tracking system.

In our project, once the object of interest passes the above two checks, then it is claimed as a valid object. Subsequently, the object is characterized by a number of parameters, namely, the object's center in the world coordinate, the object size in the world coordinate (or the width and length of a rectangular box bounding the object), the object's detailed contour or shape and its upper-left coordinate from the rectangular box. One improvement to be done is to use rotated rectangular box to bound the object as the object (or vehicle) may make turning movements. In this processing step, the last action is to store a valid object in a list of objects for further processing.

Figure 3.4 shows the objects from Figure 3.3 bounded by the rectangular box and the characterization of the object. Note that each object is characterized by center, size, shape detail and upper-left coordinate (the 'used' parameter is for coding purpose only). Note that the object, A2, in Figure 3.3(b) is not claimed valid as it is too close to the view boundary, therefore is not shown in Figure 3.4.

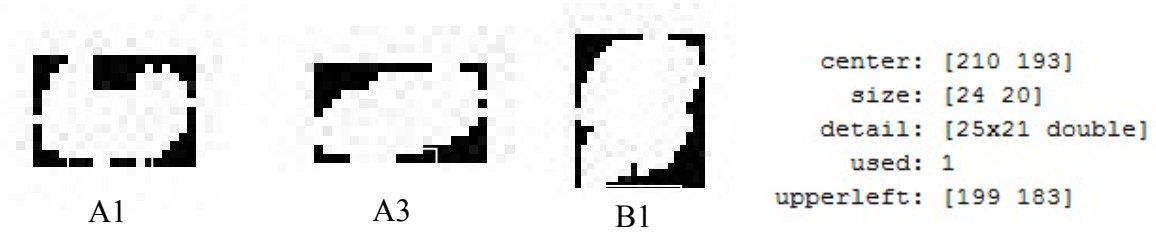


Figure 3.4: The extracted object from Figure 3.2.

3.3.4 Object Overlap Check

Once the objects are extracted from the above step, they are ready to be considered for potential vehicles. However, objects from different cameras may correspond to the same one, and this should be found out. To do that, a correspondence check needs to be performed so that objects from both cameras corresponding to the same object/vehicle are identified. Such a correspondence check is done by comparing the overlap between objects from both cameras. This idea is based on the observation that in the ideal situation two objects from two cameras would have 100% overlap in the world coordinate if camera calibration, object detection etc are all perfect. In reality, less than 100% overlap would still give valuable information on the correspondence of the objects.

We differentiate four cases here. Case 1 is that when both cameras have valid objects, case 2 and 3 that only one camera has valid objects and case 4 none has valid objects. Case 1 is the typical case, for which we describe in a little more detail. Basically, any object from camera A is compared against all objects in camera B to compute their overlap percentages, which are measured as the amount of pixels in overlap over the number of pixels of the objects. This gives two overlap percentages and the smaller one is taken as the overlap percentage of the two objects. The overlap percentage ranges from 0 to 100% and at which threshold the two objects are taken as the corresponding object/vehicle may not be clear. In our current design, we

consider a very small threshold of 5 to 10% to simplify the vehicle tracking later. For example, referring to Figure 3.4, object B1 from camera B is compared against object A1 and A3 from camera A and their overlap percentages are shown below in Table 3.3.

Table 3.3: Overlaps between objects from two cameras.

	A1	A3
B1	0.3	0

3.3.5 Create Potential Vehicles

In previous steps, we are dealing with objects only. Note that at the current image frame, we extract the objects and we now need to create potential vehicles from these objects at the current image frame. As explained before, in general case, an object is not equivalent to a vehicle; rather due to vehicle occlusion and noisy detection, an object may be part of a vehicle, a vehicle or multiple vehicles. For example, Object B1 in Figure 3.4 from camera B is actually two occluded vehicles (see the original Figure 3.2). Also, note that we probably should not create vehicles directly from objects from either camera, as different objects from both cameras may correspond to the same vehicle.

Let us consider the general case that both cameras have objects, therefore a matrix giving overlap percentages between any two objects from both cameras is available from the above processing step, which must be used to create potential vehicles. We start creating vehicles from the list of objects from camera A. Similar to creating an object, there are a number of conditions to check to create a valid potential vehicle. First, the object from camera A must be larger than a size threshold in order to filter out noise and also to simplify vehicle tracking later. In our project, we set the size threshold to create a vehicle to be slightly larger than that to create an object. For example, the size threshold is 28 feet^2, while that for an object is 24 feet^2. Such a threshold is empirical as well, as it is known that the smallest vehicle, such as a compact two-person only vehicle, has a bottom area size of around 30 feet^2. Again, this threshold can be adjusted to evaluate the overall impact on the vehicle tracking module.

Once the size check is passed, we turn eyes to the matrix giving overlaps between the objects from camera B with respect to all other objects from camera A. There are three cases to consider here.

- The first case is that the current object Bk (the kth object from camera B) overlaps with more than one object from camera A, then those objects from camera A are recorded. Supposing those objects are labeled as Am and An, then object Bk and Am are used to create one potential vehicle and object Bk and An to create another potential vehicle. Such actions are based on the vehicle correspondence idea mentioned in previous step that two objects having overlap would correspond to the same vehicle or multiple vehicles. For example, in Table 3.3, had the other object A2 claimed valid, then B1 would overlap with both A1 and A2.
- In the second case, current object Bk overlaps with only one object Am and they are used to create one potential vehicle. For example, in Table 3.3, the object B1 and A1 are used to create one potential vehicle.

- The last case is that current object Bk does not overlap with any object from camera A, and in this case only object Bk itself is used to create a potential vehicle.

After all objects Bk from Camera B are considered, the left objects from Camera A are then considered to create potential vehicles. For example, for the above Table 3.3, the sequence of actions would be first to create a potential vehicle from B1 and A1, and then another one from A3 only.

Next, we briefly discuss how to characterize the created potential vehicle. In fact, it is very similar to characterizing an object. However, it should be noted that the overlap of two objects is actually a vehicle or multiple vehicles. In case both objects are actually the same single vehicle, then theoretically, the overlap of these two objects would be the bottom area of the single vehicle, from which one can extract the width and length of the vehicle. Except size, a potential vehicle is also characterized by states, predicted states, Kalman parameters (to be detailed later), a start frame (i.e. the frame when the potential vehicle is created, which in this case would always be the current image frame), an end frame (i.e. the last frame when vehicle could still be detected, which is not really meaningful or used for potential vehicles), lifetime (this one is not used for potential vehicles), and shape detail. The state vector, denoted as [x y v_x v_y] is associated with each potential vehicle, where (x,y) gives the center position of the vehicle in the world coordinate and (v_x v_y) the velocity along the world coordinates. Note that a potential vehicle is a claimed vehicle for the current image frame only, not a tracked vehicle since its detection, therefore its staring velocity is always declared as 0 (in fact the velocity for a potential vehicle is not important at all). Similarly, the predicted state takes the same as the state. Also, for convenience of overlap computation in later processing steps, we also record the upper-left coordinate of a potential vehicle similar to the case of an object. Note that the shape of a potential vehicle is computed from overlap of the two involved objects. When a potential vehicle is created from a single object, then its shape directly takes that of the object, similarly the size and upper-left coordinate. Figure 3.5 shows how a potential vehicle is represented in our vehicle tracking module. The right of Figure 3.5 also shows two created potential vehicles following the above example in Figure 3.4. Note that the potential vehicle on the left is obtained from overlap of A1 and B1, and that on the right is from A3 alone (therefore one can notice that potential vehicle has the same shape and size as the object A3 shown in Figure 3.4).

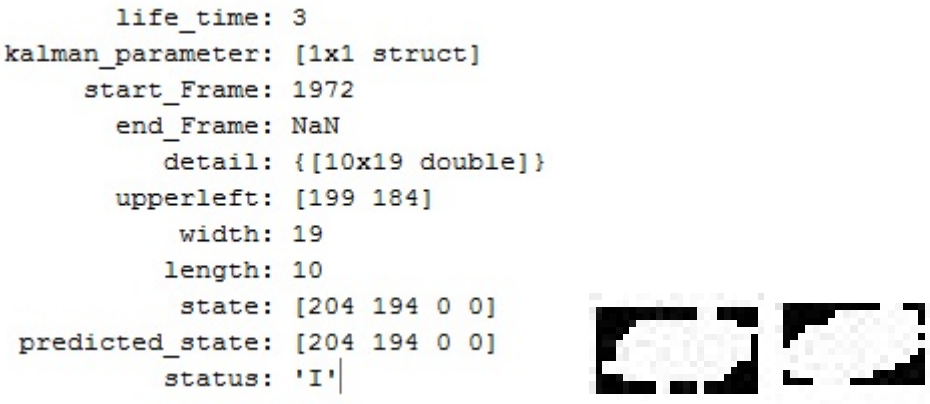


Figure 3.5: The created potential vehicles from Figure 3.4 and Table 3.3 and its characterization.

3.3.6 Check Relations of Potential Vehicles with Current Vehicles

In the previous processing step, potential vehicles are created at each image frame from overlap relations between valid objects from both cameras. As discussed, the created potential vehicles only represent possible vehicles at current image frame that are to be associated with current valid vehicles, which is another list of vehicles that have been tracked up to the previous frame, so that current vehicles can be tracked up to the current image frame using potential vehicles. (Note that the list of current vehicles are eventually the desired outputs of the vehicle tracking module.) Therefore, in this subsection, we discuss the next processing step, which is to find out the relation between potential vehicles at current image frame, say N, and current valid vehicles that have been tracked up to the previous frame, N-1. Once the relations are found, we can associate potential vehicles to current vehicles so that current vehicles can be updated and tracked to frame N, which is to be discussed in the next subsection.

The basic principle that we use to find the relations between potential vehicles and current valid vehicles is based on the overlap again, and this idea has been used previously to create potential vehicles from overlap relations of objects. Such an idea is valid here as well to relate potential vehicles and current vehicles, because ideally a potential vehicle at the current image frame would overlap exactly with a current vehicle (provided that the current vehicle is well predicted at the current image frame) if they are the same vehicle. In practice, the percentage of overlap is affected by several factors even if they are the same vehicle, such as the prediction error of the current vehicle at current image frame, vehicle size and shape changes between frames due to poor detection.

When checking for overlap between a potential vehicle with a current vehicle, it should be noted that the current vehicle needs to be predicted so that its position, shape, size etc can be propagated to the current image frame (this is because the current vehicle list contains all current vehicles in the system that have been tracked starting from the frame of their detections to the previous frame N-1). Accordingly, the upper-left coordinate of the vehicle would be updated as well. However, so far we have chosen not to update shape, size etc to simplify the vehicle tracking. Then, we compute the overlap between a potential vehicle with a current vehicle using the same approach as in the above step of computing overlap between objects. And again, the percentage of overlap is computed by dividing the overlap size in terms of the number of overlapped pixels over the potential vehicle size and current vehicle size and eventually the minimum of the two is recorded as the overlap of the potential vehicle and current vehicle. Therefore, the overlap check results in a matrix just like that for objects. Following the above example in Figure 3.5, there are two potential vehicles created at current image frame N and suppose that there are two current vehicles tracked up to the image frame N-1, then an example overlap matrix may be shown in Table 3.4 below. The first column of the matrix shows the list of potential vehicles (named as P_v1, P_v2 here) and the first row the list of current vehicles (named as C_v1, C_v2). So, the matrix shows that the potential vehicle P_v1 has 72% overlap with the current vehicle C_v1 and similarly the P_v2 has 75% overlap with C_v2. Such relations derived from overlap check is very important because it implies that P_v1 at frame N may be associated with a tracked vehicle C_v1 up to frame N-1, similarly P_v2 may be associated with C_v2, and the confidence of the association depends on the percentage of overlap. Therefore, with the associations, the potential vehicle P_v1 can be used to update the current vehicle C_v1

such that the vehicle C_v1 can now be tracked all the way to the current image frame N. The update procedure will be discussed in detail later.

Table 3.4: Overlap relations between potential vehicles and current vehicles (Left: overlap percentage matrix; Middle: strong match matrix; Right: loose match matrix).

	C_v1	C_v2		C_v1	C_v2		C_v1	C_v2
P_v1	0.72	0	P_v1	1	0	P_v1	0	0
P_v2	0	0.75	P_v2	0	1	P_v2	0	0

Referring to Table 3.4 again, P_v1 does not overlap with C_v2 and P_v2 does not overlap with C_v1 either. It should be noted that ideally one potential vehicle should be uniquely related to a current vehicle, the so called one-to-one association. If one potential vehicle is related to multiple current vehicles (called one-to-many association), or multiple potential vehicles to one current vehicle (called many-to-one association), or multiple potential vehicles to multiple current vehicles (many-to-many association), then the update procedure is much more complicated than the one-to-one association. In our system design, we consider most of these cases. However, to simplify vehicle tracking, we simplify the relations using two simple techniques. One technique is to create two binary matrix, one called a strong match matrix and the other called a loose match matrix. Both matrix have the data structure as shown in Table 3.4 (the middle one and the right one). The strong match matrix indicates that a potential vehicle has significant overlap with a current vehicle and a threshold defines the significant overlap. For example, we use 80% as the threshold. A strong match also indicates a loose match but not vice versa. So, when a potential vehicle has a strong match with a current vehicle, then it is declared as an one-to-one association, though it may not originally be in the loose match matrix (in other words, the potential vehicle may have small overlaps with other current vehicles in the loose match matrix). Therefore, a strong match supersedes the loose match matrix in terms of priority of associations. Only if the strong match is 0 for an entry in the strong match matrix, do we then go to define vehicle associations from the loose match matrix.

The second technique is to remove some entries of very small overlap percentages for the case of many-to-many associations. For example, if in Table 3.4, the entry for P_v2 and C_v1 is 0.05, which indicates there is a very small overlap between them and there is a 2-to-2 association (because C_v1 has overlap with both P_v1 and P_v2, and P_v2 has overlap with both C_v1 and C_v2). As it is a very small overlap percentage, it may be set to 0 so as to simplify the association between potential vehicles and current vehicles. One should note that such a technique does not necessarily reduce the many-to-many association to a many-to-one, one-to-many, or one-to-one association. In fact, a many-to-many association may still exists, but simplified.

All the cases of associations need to be considered when updating current vehicles with potential vehicles due to complex vehicle behavior, such as merging, splitting and vehicle occlusion. This will be discussed in the next Section.

```
       life_time: 3
kalman_parameter: [1x1 struct]
    order_number: 1
     start_Frame: 1971
       end_Frame: NaN
          detail: {[9x18 double]  [10x19 double]}
       upperleft: [199 184]
           width: [2x1 double]
          length: [2x1 double]
           state: [2x4 double]
 predicted_state: [2x4 double]
          status: 'I'
```

Figure 3.6: The characterization of a current vehicle.

As discussed above, to check the overlap between potential vehicles and current vehicles, the current vehicles that have been tracked up to frame N-1 needs to be predicted to current image frame N in order to have a reasonable overlap check. If without the prediction, the potential vehicle could possibly have no overlap with the current vehicle at all even if they are actually the same vehicle but in different image frames. In literature, Kalman filtering techniques have been widely used for prediction and proved useful [53] to estimate the state of vehicles in the next frame based on their current states and Kalman parameters. The Kalman filtering is a set of mathematical equations that provides a recursive solution of estimations of past, present, and future states. We use the Kalman filtering to predict the state of a current vehicle in the next image frame and save it in the predicted state vector (see Figure 3.6). After association in the current image frame, the current vehicle's state vector and predicted state vector are both updated and then tracking is repeated in the next image frame. The state vectors at all times are recorded to derive the complete vehicle trajectory since its detection until exit.

The characterization of a current vehicle, shown in Figure 3.6, is very similar to that of a potential vehicle shown in Figure 3.5. Note that some parameters are not really used in potential vehicles, but all are used for current vehicles. The lifetime parameter is used to track vehicles under vehicle misses. If a vehicle is not detected for one image frame, its lifetime is reduced by one and the vehicle continues to be tracked with Kalman filtering techniques. However, the vehicle is declared a dead vehicle (the vehicle is either completely lost in the camera view or it truly exits the camera view) if lifetime is reduced to 0. Other parameters have been explained in the above Sub-section. Finally, it should be noted that many parameters of a current vehicle, such as shape, width, length and state, have all their values saved at each image frame as long as it is not a dead vehicle yet. This is important, as all states of a current vehicle throughout its life in the camera view gives the vehicle trajectory. All shapes throughout its life gives powerful information to more accurately analyze the vehicle's size.

3.3.7 Update Current Vehicles

Recall from the previous Sections, the list of potential vehicles contain all potential vehicles extracted from the current image frame N and list of current vehicles contain all vehicles that have been tracked up to image frame N-1. And the potential vehicles and current vehicles have

been associated, so now potential vehicles can be used to update current vehicles so that current vehicles can now be tracked to the current image frame N. We detail this step as follows. We discuss the several cases of update that are considered in our design and then how to perform the update.

First, we consider one-to-one associations. We update current vehicles by first checking the strong match matrix discussed in the previous Section, because a strong match means a large percentage of overlap, thus it has higher priority in terms of updates. So, if a current vehicle, C_vx has a strong match with a potential vehicle P_vy, then P_vy is solely paired with C_vx. In other words, only P_vy is used to update C_vx and at the same time P_vy is not to update any other current vehicle. After all one-to-one associations from strong match matrix have been considered, we then check the loose match matrix. Similarly, if a current C_vx has a loose match with a potential vehicle P_vy, then P_vy is solely paired with C_vx. For detailed actions in this update, see update 1 below.

Second, we consider one-to-many associations (that is one current vehicle is associated with multiple potential vehicles). Note that such a case could be very complicated including many possible scenarios, while we focus on two of them. One scenario we consider is that the current vehicle involved in this one-to-many association is indeed a single vehicle (based on size check) splitting into multiple potential vehicles, and typically in this case the multiple potential vehicles are just pieces of the single current vehicle (due to noisy detection for example) and should empirically have smaller sizes than the current vehicle. However, note that in another scenario, the current vehicle involved in this association is not necessarily a single vehicle, but possibly joint multiple vehicles (for example, when multiple vehicles enter the camera view occluded, therefore have been jointly tracked as just one current vehicle), and later due to better camera view or different vehicle speeds, the multiple vehicles may not be occluded anymore and therefore this can causes a one-to-many association as well. To differentiate these different scenarios, we could perform an empirical size check on the current vehicle and potential vehicles involved in this association. In the first scenario, we would update the current vehicle with the multiple potential vehicles joined (see update 2). In the later scenario, we would update the current vehicle with one of the multiple potential vehicles (see update 1) and create new vehicles from the rest of potential vehicles (see update 4).

Third, we consider many-to-one association (that is a potential vehicle is associated with multiple current vehicles). Again, like the above one-to-many association, this case may have many possible scenarios. In our system design, we focus on one scenario that multiple current vehicles merge to a potential vehicle. We first perform a size check that the current vehicles have smaller sizes than the potential vehicle and at the same time check that the current vehicles have large enough sizes to split or merge (a threshold is set for this purpose). If both conditions are met, then multiple current vehicles are considered to merge into one larger-size potential vehicle. As all the involved current vehicles are associated with the single potential vehicle, and it may not be so clear how to update each current vehicle. In our system design, we perform an overlap-based optimization to update each current vehicle with the single potential vehicle (see update 3). If the above conditions are not met, especially some of the current vehicles do not have large enough sizes to split or merge, then those current vehicles are deleted (see update 6) and only the rest of current vehicles are associated with the potential vehicle for update (see update 1).

Fourth, the other case left is the many-to-many associations, which is even more challenging than the above two cases that could involve even more scenarios. Currently, in our system design, we have not found that such a case has happened. On one hand, the frame rate of video is relatively high, 10 to 30 frames a second, therefore the vehicle motion between two consecutive frames is small, which is less likely to cause complex associations. On the other hand, the simplification we have used in Section 3.4.6 has helped get rid of these cases.

Fifth, another case is that a current vehicle does not associate with any potential vehicle. The main scenario to consider in this case is that the current vehicle has a miss in the current image frame N due to noise for example. In this scenario, the current vehicle is updated with a prediction by Kalman filtering (see update 5).

Sixth, another case is that a potential vehicle does not associate with any current vehicle. The main scenario to consider is that the potential vehicle in current image frame N may be a new vehicle to be added into the list of current vehicles. Therefore, the update is basically a new vehicle creation and the routines to create a new vehicle are practiced such as the size check (see update 4).

After discussing all cases of associations considered in our vehicle tracking module, we next detail the update procedures as cited in the above six cases.

Update 1: this is used in one-to-one associations. In this update, Kalman parameters, current vehicle's state and predicted state are updated, and current vehicle's shape detail, width, length and upper-left coordinate for current image frame N are all set to those of the potential vehicle. Note that current vehicle's lifetime is always reset as the vehicle has an association at current image frame.

Update 2: in this update, the only difference compared to that in update 1 is what is used to update the current vehicle is not just one of the potential vehicles, but all of them joined. Therefore, we first create a single temporary vehicle combining all those potential vehicles and use that single vehicle's shaped detail, width, length and upper-left coordinate to update those for the current vehicle.

Update 3: in this update, the major action is overlap-based optimization and the purpose of this optimization is to maximize coverage of the potential vehicle's shape detail by the merging current vehicles. For example, considering a simple case of two merging vehicles A and B associated with the potential vehicle C, we perform the following operations:

$$F = (A \cap C) \cup (B \cap C) \; and \; D = F \oplus C$$

where D represents the area of the potential vehicle that is not covered by the current vehicles. We compute the center of D, and then iteratively move the current vehicles toward that center to cover the overall shape detail of the potential vehicle as much as possible until no improvement is achieved.

Update 4: this update is very similar to creating a new vehicle. The new current vehicle is directly added to the end of the current vehicle list. Note that it needs to be properly numbered to reflect the total number of current vehicles that have been tracked so far.

Update 5: this update basically updates those current vehicles that does not have any association. In this update, we continue to update the Kalman parameters, state and predicted state and the upper-left coordinate. But the shape detail, width and length are kept the same. An important action in this update is to reduce the lifetime by one, as the current vehicle is not detected at current image frame. If the lifetime is reduced to 0, then this vehicle is declared dead.

Update 6: in this update, we delete invalid current vehicles. To do that, we simply set the lifetime of those vehicles to 0 and record their end frames to declare dead vehicles.

At the end of the updates, we check the list of current vehicles for dead vehicles and remove them from the list (and store them in a dead vehicle list) such that the list of current vehicles contains only live vehicles, which helps reduce the memory requirements for the software design. The pseudo code for the vehicle tracking module can be found in Table 3.5.

Table 3.5: Pseudo code for the tracking algorithm.

```
%Initialize
set size and percentage thresholds
initialize the list of objects, potential vehicles and current vehicles

for all frames
   segment foreground using MoG; (section 3.3)
   image processing to refine foreground; (section 3.4.1)
   image to world coordinate transformation; (section 3.4.2)
   object extraction from all cameras; (section 3.4.3)

%object overlap check (section 3.4.4)
for all objects from camera A
   for all objects from camera B
      check overlap between objects;
   endfor
endfor

%create potential vehicles (section 3.4.5)
for all objects from camera A
   if object size is large enough
      if object has overlap with more than one from camera B
         create a potential vehicle from each overlap
      elseif object has overlap with a single one from camera B
         create a potential vehicle from the overlap
      elseif object has no overlap with anyone from camera B
         create a potential vehicle from the object itself
      endif
```

```
   endif
endfor
for all objects from camera B
   create potential vehicles from the rest of objects in B;
endfor

%check relations of p_vehicles with c_vehicles (section 3.4.6)
for all c_vehicles
  for all p_vehicles
    check overlap between current and potential vehicles;
    create strong match and loose match matrix;
    simplify the matrix (delete small overlap percentages and
            the smaller percentage in a many-to-many association)
  endfor
endfor

%update current vehicles; (section 3.4.7)
Update current vehicles from strong match matrix first (update 1);
%Update current vehicles from loose match matrix then;
for all current vehicles
 for all potential vehicles
  if there is a loose match at current entry
    if one-to-one association
      update 1;
    elseif one-to-many association (c_vehicle splits to p_vehicles)
      perform size check that c_vehicle is larger than p_vehicles;
      if all potential vehicles have large enough size
      update c_vehicle with one p_vehicles (see update 1);
      create new current vehicles from the rest of p_vehicles
                                    (see update 4);
    else
      update c_vehicle with all p_vehicles combined
                                    (see update 2);
    endif
  elseif many-to-one association (c_vehicles merge to p_vehicle)
    perform size check that c_vehicles smaller than p_vehicle;
    if all current vehicles have large enough size
      overlap-based optimization (update 3);
    else
      update c_vehicles with large-size one and delete rest;
    endif
  endif
 endif
endfor
endfor
```

```
for all current vehicles
  if no match for a current vehicle
    current vehicle missing (update 5)
  endif
endfor

for all potential vehicle
  if no match for a potential vehicle
    create a new current vehicle (update 6)
  endif
endfor

for all current vehicles
  if a current vehicle is dead
    move it to a dead vehicle list;
  endif
endfor
```

3.4 Vehicle Trajectories

In the above Sections, we have described in detail how the vehicle tracking module track vehicles. The designed module is coded in C/C++ and MATLAB software [40].

Note that the output of this vehicle tracking module is vehicle trajectories, which is achieved by concatenating vehicles' states at each time moment since their first detections to exits. As mentioned before, a vehicle's state includes its position (x and y coordinate in the world coordinate) and speed (x and y coordinate speed in the world coordinate). In our software design, a vehicle's all states are saved, therefore giving a vehicle's position and speed at each time moment. Figure 3.7 shows a sample vehicle trajectory for the two images from both cameras and all states of that vehicle in the software. Note that the x, y position and speed are measured in feet and feet/second respectively. In this example, the estimated vehicle speed on average during turning is 15 miles-per-hour, which seems to be very reasonable for intersections.

x,y position		x,y speed	
203	191	0	0
203	193	0	2
203	194	0	1
204	197	1	3
205	198	1	1
205	200	0	2
205	201	0	1
205	202	0	1
206	204	1	2
206	205	0	1
206	207	0	2
206	210	0	3
207	211	1	1
207	213	0	2
207	214	0	1
207	216	0	2
206	218	-1	2
206	220	0	2
206	221	0	1
205	223	-1	2
204	225	-1	2
204	226	0	1
204	227	0	1
204	229	0	2
205	229	1	0
205	229	0	0
203	229	-2	0
202	231	-1	2
202	231	0	0
201	233	-1	2
201	234	0	1
200	236	-1	2
199	238	-1	2
198	240	-1	2
197	242	-1	2

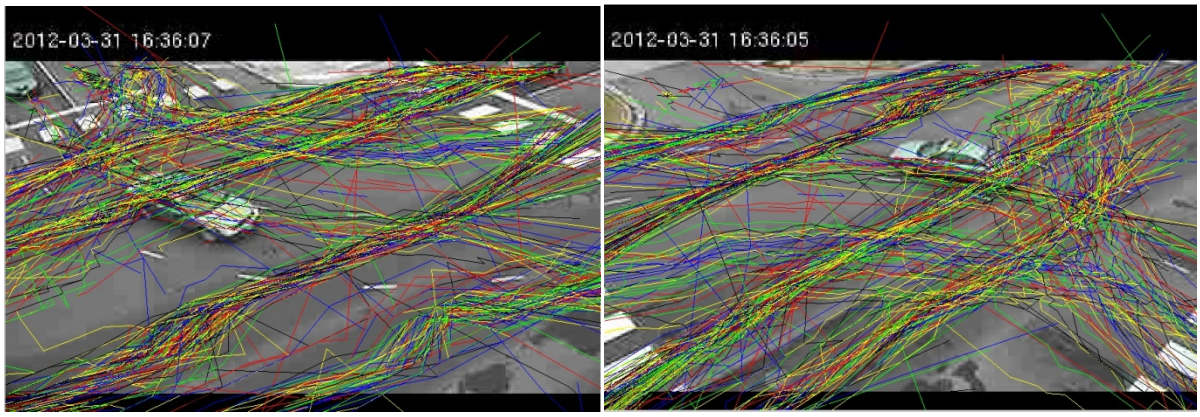
Figure 3.7: A sample vehicle trajectory and its states in the software.

Ideally, each vehicle that enters the camera view should be tracked to derive its trajectory. However, in practice, this is rarely possible due to many factors, such as image noise or poor detection. Besides, the vehicle tracking algorithm described in Section 3.4 does not take all cases of vehicle associations into account. Therefore, the output of the vehicle tracking module is vehicle trajectories that have been tracked by the system, not by human inspection, however it saves significant cost and labor as it automates the vehicle tracking process. In fact, the difference between human inspection and the system gives error of vehicle tracking. As the accuracy of traffic performance measurements directly depends on the accuracy of the obtained vehicle trajectories, it may be insightful to understand what the sources of error may be for the trajectories. We classify them into a few categories, but it should be noted by no means are they complete.

- First, there may be missing vehicle trajectories that do not exist in the obtained vehicle trajectories at all. In this case, a vehicle is simply not tracked by the system, probably due to poor detection and segmentation results.

- Second, one vehicle trajectory does not necessarily correspond to one vehicle, but combined several vehicles or just a fragment of a single vehicle. This is mainly caused by the vehicle tracking algorithm, as some complex cases of associations are not considered.
- Third, some of the vehicle trajectories are not complete in the sense that the vehicles are not tracked all the way from entrance to the camera view to the exit. Again, the vehicle tracking algorithm may mis-tracked vehicles in this case.
- Fourth, some vehicle trajectories may not be correct in the sense that the vehicles do not really move in the way as shown by the trajectory. Some of these can be seen in the following Figures. It is again the vehicle tracking algorithm that is mainly responsible for this.

(a)



(b)

Figure 3.8: (a) Vehicle trajectories overlaid in both images from the two cameras; (b) Five vehicle trajectories in red, blue, green, yellow and black (the red one does not seem right).

Figure 3.8(a) shows the overlay of all vehicle trajectories on the images from both cameras for a 35-minute video recorded from 3-4pm on March 31st, 2012 for the intersection traffic scene in Chapter 2. Note that the vehicle moving directions, including straight-through and turning, can be clearly recognized in spite of some wrong vehicle trajectories. For example, there are two

lanes for each straight-through direction. Also, there are two lanes of traffic for the South-to-West turning direction, but only one lane of traffic for the East-to-South turning direction (for the definition of direction, refer to Figure 4.1 in the next Chapter as well). In order to show more detail of a vehicle trajectory, we randomly pick five vehicle trajectories and show them in Figure 3.8(b) (again using five colors: red, green, blue, yellow and black). As can be seen, in both figures, the vehicle trajectories shown in red does not look correct in both Figures in 3.8(b), which may fall into one of the cases we discussed above. The main point here is that such a vehicle trajectory will not be used for traffic data collection in the next Chapter, because it does not seem a correct one.

3.5 Summary

In this Chapter, we present in detail the vehicle tracking module in the overall traffic performance measurement system. Vehicle tracking is the core of the traffic performance measurement system, as eventually the accuracy of collected traffic data depends on how well vehicle tracking is performed or the accuracy of the obtained vehicle trajectories in this step. Vehicle tracking includes two major steps, vehicle segmentation and the actual tracking. Vehicle segmentation is very critical and in this project we propose to use the MoG method for vehicle segmentation to cope with practical camera shaking issues. In the tracking part, we propose a sequence of steps to associate potential vehicles in current image frame to current vehicles that have been tracked up to the previous frame. The output of vehicle tracking is vehicle trajectories, which are to be processed in the next module to extract traffic performance measurements.

Chapter 4. Traffic Data Collection

The proposed traffic performance measurement system has mainly three modules, camera calibration, vehicle tracking and data mining as mentioned in Chapter 1. Camera calibration has been discussed in detail in Chapter 2, followed by the vehicle tracking module discussed in detail in Chapter 3. As the outputs from the vehicle tracking module are vehicle trajectories of all vehicles (provided they are correctly detected and tracked), and from those the data mining module needs to mine the vast amount of data for useful traffic data, therefore we call this step data mining. Note that this step is very similar to that in the single-camera vehicle tracking system reported in [14] in spite of different traffic scenes (roundabout entrance in [14] versus intersection in this project), as data mining takes vehicle trajectories as inputs, which is independent of the number of cameras. Therefore, the reader could also refer to [14] for some traffic performance measurements for roundabouts and how to extract them.

In the rest of the Chapter, we briefly discuss how to automatically extract traffic data from the resulting vehicle trajectories provided by the vehicle tracking module. In Section 4.1, we discuss how to extract vehicle count and travel time. In Section 4.2, we focus on accepted and rejected gaps. Then, some experiment results are shown in Section 4.3. Finally, a summary of this Chapter is offered in Section 4.4.

4.1 Vehicle Count and Travel Time

In traffic data collection, vehicle count is a basic type of data to acquire, especially in our project targeting intersections. Here, we target vehicle count for all straight-through and turning directions. To do that, we check on each vehicle trajectory and detect how its x and y coordinates in the world coordinate change throughout its life in the camera view. Recall that back in Chapter 2, the x coordinate is defined to be parallel to one roadway (W Arrowhead Rd) while the y coordinate parallel to the other roadway (Rice Lake and Sawyer Avenue) which is perpendicular to the former one. For example, for a straight-through vehicle along the W Arrowhead Rd, the vehicle's y coordinate is supposed to be relatively constant while the x coordinate would change in its lifetime, though this change may not be predictive as it depends on traffic conditions and the vehicle's velocity, acceleration and de-acceleration behavior. Once the vehicle is found to be moving along the y coordinate, we could then further check whether it is moving West or East along y coordinate by checking out whether the x coordinate is going positive or negative. Another example, suppose we found that both x and y coordinates have significant changes along a vehicle's lifetime, which typically imply a turning vehicle, we could then determine its turning direction by checking out how x and y coordinate changes.

For travel time, a proper definition is necessary before we compute the exact time. There could be a few definitions of travel time. For example, travel time can be defined as the time it takes for a vehicle to move from location A to location B. However, such a definition is not well accommodated in our peculiar traffic scene. As mentioned in Chapter 2, the two cameras are too close to the intersection and does not offer a very wide view, therefore setting two locations with a certain distance away is not appropriate. Being aware of this, we simply define the travel time as the time difference between entrance to one of the camera views to exit, which could be measured directly from the vehicle trajectory by diving the total number of frames by the frame

rate of the video. However, the downside of this way of measuring travel time is that it is not consistent for all vehicles, as vehicles may travel unequal distances due to the view coverage of the two cameras.

The following Figure 4.1 defines all possible straight-through and turning directions for the intersection.

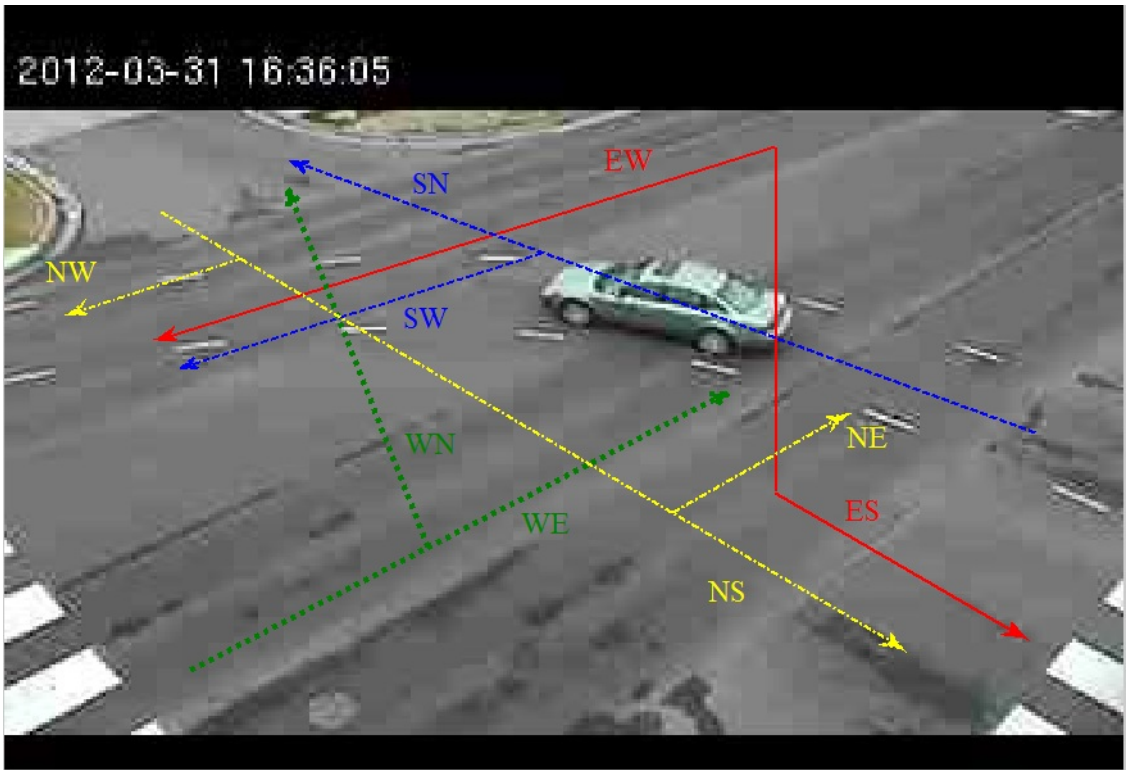


Figure 4.1: Illustration of all straight-through and turning vehicles.

4.2 Accepted and Rejected Gaps

For the case of intersections, we can also extract accepted and rejected gaps for turning vehicles. First, we check out all those turning vehicles from the vehicle trajectories and classify the turns into a few cases as discussed in 4.1. Then, for each turning case, for example the WS turn as shown in Figure 4.1, we check out for each WS turn vehicle whether there are vehicles moving WE because WS turn vehicles have to yield to WE vehicles and if there are, then the accepted gap can be computed as the time or the distance for the WE vehicle to reach the location of the WS vehicle from its current location at time of WS vehicle turning. Similarly, rejected gaps can be computed if a WS turn vehicle waits during the course of its vehicle trajectory and the time or the distance that it takes for the WE vehicle to reach the location of the WS vehicle from its current location at time of WS vehicle waiting is the rejected gap. Of course, eventually when the WS vehicle makes the turn, it may results in an accepted gap if there are WE vehicles. Gaps for other turning directions can be computed similarly as the above example.

In our project, since the camera views are very limited as mentioned in Chapter 2, when a vehicle makes a turn, it is not often to see straight-through vehicles to be yielded by the turning vehicles. Therefore, in our project we did not focus on these two performance measures.

4.3 Experiment Results

The overall data collection system, including the camera calibration module, vehicle tracking system module and the data mining module discussed in this Chapter, has been implemented in C and MATLAB and runs on a 2.33GHz Xeon PC.

As mentioned at the end of Chapter 3, we tested the vehicle tracking system with a 35-minute video and obtained the vehicle trajectories. These vehicles trajectories were then given as inputs to the data mining module and we can then obtain traffic performance measurements as discussed in Chapter 4.1 and 4.2. The processing time (mainly the execution time for the vehicle tracking module and data mining module) for a 35-minute video is currently 6 to 10 hours in the PC. One should note that given a video the processing time is related to the number of cameras in the multiple-camera system (in our case, two cameras), frame rate of the video (in our case, 10 frames per second), the complexity of the vehicle tracking algorithm and data mining algorithm.

We show some results below for the system with main focus on the vehicle counts for all directions of traffic. To make a comparison, we then did a manual vehicle count, which is extremely time-consuming and in fact this takes much more time than the proposed traffic performance measurement system. However, one should note that showing the overall vehicle count for the overall 35 minutes may not be a good measure in the sense that it may not reflect the matching accuracy between the extracted vehicle trajectories to the actual ones, but only the count accuracy. In other words, a vehicle trajectory obtained from the vehicle tracking system increase the vehicle count by one, but in reality this vehicle may not really exist or be mis-tracked. This number of mis-tracked vehicles and the number of missed vehicles (i.e. those vehicles that exists in reality but were not tracked) may offset each other and this may eventually end up giving a false overall vehicle count accuracy. One way to have a better accuracy measure is to divide the 35-minute duration to many segments of small intervals and evaluate the accuracy for each small interval. Therefore, in Table 4.1, we show the vehicle count for 7 segments of 5 minutes each with accuracy. It can be seen that on average the vehicle count accuracy is over 90% (except when statistics limits a good accuracy measure, such as in case (e) for the ES direction).

Table 4.1: Vehicle count for each direction of traffic for a 35-minute video.

	ES	EW	WE	WN	SN	SW	NS	NW	NE
Estimated	6	38	47	0	0	7	0	0	0
Ground truth	6	40	50	0	0	7	0	0	0
Accuracy (%)	100	95	94	-	-	100	-	-	-

(a) 03:36-3:41pm

	ES	EW	WE	WN	SN	SW	NS	NW	NE
Estimated	4	52	46	0	0	9	0	0	0
Ground truth	4	52	47	0	0	9	0	0	0
Accuracy (%)	100	100	98	-	-	100	-	-	-

(b) 3:41-3:46pm

	ES	EW	WE	WN	SN	SW	NS	NW	NE
Estimated	10	50	56	1	0	7	0	0	0
Ground truth	11	51	55	0	0	7	0	0	0
Accuracy (%)	91	98	98	-	-	100	-	-	-

(c) 3:46-3:51pm

	ES	EW	WE	WN	SN	SW	NS	NW	NE
Estimated	7	28	49	1	0	5	0	0	0
Ground truth	7	28	48	0	0	5	0	0	0
Accuracy (%)	100	100	98	-	-	100	-	-	-

(d) 3:51-3:56pm

	ES	EW	WE	WN	SN	SW	NS	NW	NE
Estimated	1	31	44	0	0	8	0	0	0
Ground truth	2	29	46	0	0	8	0	0	0
Accuracy (%)	50	94	96	-	-	100	-	-	-

(e) 3:56-4:01pm

	ES	EW	WE	WN	SN	SW	NS	NW	NE
Estimated	11	38	48	0	0	4	0	0	0
Ground truth	13	43	49	0	0	4	0	0	0
Accuracy (%)	85	88	98	-	-	100	-	-	-

(f) 4:01-4:06pm

	ES	EW	WE	WN	SN	SW	NS	NW	NE
Estimated	8	29	46	0	0	8	0	0	0
Ground truth	8	28	46	0	0	7	0	0	0
Accuracy (%)	100	97	100	-	-	88	-	-	-

(g) 4:06-4:11pm

4.4 Summary

In summary, this Chapter briefly discusses the last module of the overall traffic performance measurement system, which is the data mining module to extract the interested traffic data from the vehicle trajectories obtained from the previous vehicle tracking module. We also showed some results of traffic performance measurements with main focus on vehicle count for all directions of traffic. When compared to ground-truth measurements obtained from manual inspection of the videos, it is found that the accuracy on vehicle count is on average over 90%, which is improved compared 70% to 90% for a single-camera-based video system. As previous mentioned, the data mining module itself does not incur any accuracy loss given vehicle trajectories from the vehicle tracking module. Therefore, the error is strictly from vehicle tracking, for example, poor detection. For the data mining module, the main concern is its efficiency in terms of execution time.

Chapter 5. Summary and Conclusions

In this project, we have developed a multiple-camera tracking system for accurate traffic performance measurements for intersections and roundabouts. In Chapter 1, we review various methods for traffic performance measurements and conclude that camera based vision systems are preferred in this project. Traditionally, a single-camera-based vision system is used, but it suffers from limited camera view causing the vehicle occlusion problem. In this project, we propose to use a multiple-camera vision system, which has two main advantages compared to a single-camera system. On one hand, the multiple views from multiple cameras could help to significantly alleviate the vehicle occlusion problem and on the other hand, multiple views of the same vehicle give more evidence of the vehicle, which allows more robust tracking. In Chapter 1, we also describe the general processing flow for a multiple-camera tracking system.

We divide the proposed multiple-camera tracking system into three modules, the camera calibration module, vehicle tracking module and data mining module. In Chapter 2, we describe in detail the camera calibration module. In a multiple-camera tracking system, it is preferred to have a common world coordinate for all cameras. To achieve that, we define a square or a rectangular pattern for intersection traffic scenes, from which the common unique world coordinate is defined. We then extend the traditional method based on vanishing points for calibration of a single camera to multiple cameras. The extended method allows equation-based solving to derive initial camera parameters and then optimization to derive more robust camera parameters. Such an extended method is simple and efficient for camera calibration of intersection traffic scenes. With camera calibration, we can then estimate vehicle speeds and real-world distances from 2D images.

In Chapter 3, we describe in detail the vehicle tracking module, which has the most impact on the accuracy of traffic performance measurements. The vehicle tracking module has several steps of processing. First, the system takes pre-recorded videos from multiple cameras as inputs and applies the mixture-of-Gaussian background modeling method and a shaking-removal algorithm to segment objects. Then, the segmented objects are refined with image processing techniques. Subsequently, objects in the image coordinates are transformed to the world coordinates, followed by object extraction. The above several steps need to be repeated for each camera. Once objects are extracted, a correspondence check based on overlap is performed such that objects corresponding to the same one from different cameras are related. Giving the correspondence information, potential vehicles for the current image frame can be created. These potential vehicles are then associated with current vehicles that have been tracked up to the previous image frame based on overlap. There are several cases of association considered in our project. Based on these associations, current vehicles can finally be updated from potential vehicles so that current vehicles can now be tracked all the way to the current image frame. At each image frame, a current vehicle's state, including position, size, shape, etc., is recorded. By composing the states, each current vehicle's trajectory can be obtained, which is the output of the vehicle tracking module.

In Chapter 4, we present the data mining module of the developed system. The data mining module takes all vehicle trajectories from the vehicle tracking module as inputs and processes them to derive interested traffic data, such as vehicle count for all directions of traffic at

intersections, waiting time, accepted gap, rejected gap, etc. We discuss the algorithms to derive these data in Chapter 4 and also show practical experimental results. In this project, due to the limited camera view from both cameras, we mainly focus on the vehicle count. Extensive experiments on the real-world intersection have shown that the proposed data collection system can achieve, on average, 90% accuracy compared to ground truth, which is an improvement compared to the average 80% accuracy for a single-camera tracking system.

The proposed system can automatically collect traffic data with minimum requirements of manual inputs so that significant labor work and cost can be saved. Also, the system derives the complete vehicle trajectories, giving enough details to collect all types of interested traffic data. To the best of our knowledge, this project reports the first work on traffic performance measurement for intersections using a multiple-camera tracking system.

At the same time, we are also aware of the limitations of the developed system, which will be addressed in future work. First, the proposed system has not addressed night-time vehicle tracking, as vehicles are more difficult to detect at night. Second, vehicle shadow is also a difficult issue. Vehicle shadows could be recognized as part of the objects, which leads to tracking errors. Third, in general the vehicle segmentation results from the MoG algorithm can be further improved. This requires more sophisticated segmentation algorithms at the expense of processing time. However, better segmentation results could help to significantly simplify the vehicle tracking algorithm and improve tracking accuracy. Our analysis on the vehicle trajectories show that poor segmentation results cause significant trouble to the tracking algorithm. Fourth, with the current given segmentation results, the vehicle tracking algorithm can be improved as well to improve tracking accuracy. For example, more complex associations might need to be considered. Fifth, the current setup of the two cameras at the intersection gives very limited view even in the most zoom-out case as they are too close to the intersection; therefore, we have not evaluated other performance measures other than vehicle count so far.

Except addressing the above limitations, the other future work is to extend the proposed multiple-camera tracking system for traffic performance measurements at roundabouts. In this case, the three modules of the system all need some changes. For example, a new calibration method is required for multiple cameras at the roundabout. The vehicle tracking algorithm needs change, too, as the vehicle motion behavior is different.

References

1. S. Mandavilli, "Study of Operational Performance and Environmental Impacts of Modern Roundabouts in Kansas," *Transportation Research Board National Roundabout Conference*, 2005, Vail, CO.
2. K. Parma, *Going around the Neighborhood – a Roundabout Case Study in Texas*, http://www.leeengineering.com/roundabouts/, accessed Jan. 2011.
3. *Traffic Detector Handbook*, 3rd Edition, Vol. 1, pp 1-10, Oct. 2006, US DoT, Washington DC.
4. Intelligent Transportation System Unit Costs Database, Oct. 2007, US DoT, Washington DC.
5. C. Smith, C. Richards, S. Brandt, N. P. Papanikolopoulos, "Visual Tracking for Intelligent Vehicle Highway Systems," *IEEE Transactions on Vehicular Technology*, Vol. 45, No. 4, Nov. 1996, pp. 744-759.
6. D. Dailey, *CCTV Technical Report: Phase III*, Washington Department of Transportation, Technical report, WA/RD-2006-635.2, Washington, USA.
7. R. Ervin, C. MacAdam, J. Walker, S. Bogard, M. Hagan, A. Vayda, E. Anderson, *System for Assessment of the Vehicle Motion Environment (SAVME)*, UMTRI-2000-21-1, US Department of Transportation, Washington DC.
8. *Next Generation of Simulation Program (NGSIM)*, ngsim-community.org, accessed Jan. 2011.
9. http://www.trafficvision.com/, accessed Jan. 2012.
10. T. Kwon, Portable Cellular Wireless Mesh Sensor Network for Vehicle Tracking in an Intersection, Minnesota Department of Transportation, CTS 08-29, St Paul, MN.
11. A. Abdel-Rahim, B. Johnson, *An Intersection Traffic Data Collection Device Utilizing Logging Capabilities of Traffic Controllers and Current Traffic Sensors*, National Institute for Advanced Transportation Technology, University of Idaho, Moscow, ID, 2008.
12. Image Sensing Systems, Inc., Autoscope, http://autoscope.com/, accessed Jan. 2011.
13. *Automated Video Traffic Studies*, Miovision Technologies Inc, Kitchener, Ontario, 2011.
14. H. Tang, H. Dinh, Development of A Tracking-based Traffic Performance Measurement System for Roundabouts and Intersections, Minnesota Department of Transportation, CTS 12-10, St Paul MN.
15. S. Dockstader, A. Tekalp, "Multiple Camera Fusion for Multi-Object Tracking," *Proc. of IEEE Workshop on Multi-Object Tracking*, pp. 95-102, 2001, Vancouver, BC, Canada.
16. A. Tyagi, G. Potamianos, J. Davis, S. Chu, "Fusion of Multiple Camera Views for Kernel-based 3D Tracking," *Proc. of IEEE Workshop on Motion and Video Computing*, Feb. 2007, pp. 1-6, Austin TX.
17. M. Bhuyan, B. Lovell, A. Bigdeli, "Tracking with Multiple Cameras for Video Surveillance," *Proc. of Conference on Digital Image Computing Techniques and Applications,* Dec. 2007, pp. 592-599, Glenelg, Australia.

18. Z. Hu, C. Wang, K. Uchimura, "3D Vehicle Extraction and Tracking from Multiple Viewpoints for Traffic Monitoring by using Probability Fusion Map," *Proc. of IEEE Intelligent Transportation System Conference*, Oct. 2007, pp. 30-35, Seattle, WA.

19. Q. Zhou, J. Aggarwal, "Object Tracking in an Outdoor Environment using Fusion of Features and Cameras," *Image and Vision Computing*, 24 (2006), pp. 1244-1255.

20. M. Dixon, N. Jacobs, R. Pless, "An Efficient System for Vehicle Tracking in Multi-Camera Networks," *Proc. of ACM/IEEE International Conference on Distributed Smart Cameras*, Aug. 2009, pp. 1-8, Como, Italy.

21. E. Ribnick, A. J. Joshi, N. P. Papanikolopoulos, *Multi-Camera Monitoring of Human Activities at Critical Transportation Infrastructure Sites*, Minnesota Department of Transportation, CTS 08-08, St Paul, MN.

22. Z. Zhang, "Camera Calibration," *Emerging Topics in Computer Vision*, edited by G. Medioni and S.B. Kang, pp. 4-43, Prentice Hall, 2004, Upper Saddle River, NJ.

23. O. Faugeras, *Three-Dimensional Computer Vision: a Geometric Viewpoint*, MIT Press, 1993, Boston, MA.

24. P. Sturm, S. Maybank, "On plane-based camera calibration: A general algorithm, singularities, applications," *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 432–437, Jun 1999, Ft. Collins, CO.

25. Z. Zhang, "A flexible new technique for camera calibration," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 11, pp. 1330–1334, 2000.

26. Z. Zhang, "Camera calibration with one-dimensional objects," *Proc. of European Conf. on Computer Vision*, volume IV, pp. 161–174, May 2002, Copenhagen, Denmark

27. R. I. Hartley, "An algorithm for self-calibration from several views," *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 908–912, 1994, Seattle, WA.

28. T. N. Schoepflin, D. J. Dailey, "Dynamic Camera Calibration of Roadside Traffic Management Cameras for Vehicle Speed Estimation," *IEEE Trans. on Intelligent Transportation Systems*, Vol. 4, No. 2, pp. 90-98, Jun 2003.

29. S. Gupte, O. Masoud, R. F. K. Martin, N. P. Papanikolopoulos, "Detection and Classification of Vehicles," *IEEE Trans. Intelligent Transportation Systems*, Vol. 3, No. 1, pp. 37-47, Mar 2002.

30. C. C. C Pang, W.W.L Lam, N.H.C Yung, "A Novel Method for Resolving Vehicle Occlusion in a Monocular Traffic-Image Sequence," *IEEE Trans. Intelligent Transportation Systems*, Vol. 5, No. 3, pp. 129-141, Sep 2004.

31. O. Masoud, N. P. Papanikolopoulos, "Using Geometric Primitives to Calibrate Traffic Scenes," *Transportation Research Part C*, 15, pp. 361-379, 2007.

32. E. K. Bas, J. D. Crisman, "An Easy to Install Camera Calibration for Traffic Monitoring," *Proc. of IEEE Intell. Transp. Syst. Conf.*, Nov. 1997, Boston, MA.

33. Y. Li, F. Zhu, Y. Ai, F. Wang, "On Automatic and Dynamic Camera Calibration based on Traffic Visual Surveillance," *Proc. of IEEE Intell. Veh. Symp*, June 2007, Istanbul, Turkey.

34. K. Song, J. Tai, "Dynamic Calibration of Pan-Tilt-Zoom Cameras for Traffic Monitoring," *IEEE Trans. System, Man, and Cybernetics*, Vol. 36, No. 5, pp. 1091-1103, Oct 2006.

35. T. H. Thi, S. Lu, J. Zhang, "Self-Calibration of Traffic Surveillance Camera using Motion Tracking," *Proc. of IEEE Intell. Transp. Syst. Conf*., 2008, Beijing China.

36. N. K. Kanhere, S. T. Birchfield, W. A. Sarasua, "Automatic Camera Calibration Using Pattern Detection for Vision-Based Speed Sensing," *Transportation Research Record*, No. 2086, pp. 30-39, 2008.

37. S. Pumrin, D. J. Dailey, "Dynamic Camera Calibration in Support of Intelligent Transportation Systems," *Transportation Research Record*, No. 1804, pp. 77-84, 2002.

38. R. Hartley, A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2nd ed. 2003, Ch. 6, Cambridge, UK.

39. E.R. Davies, *Machine Vision: Theory, Algorithm, Practicalities*, Morgan Kaufmann, 3rd ed. Ch. 12, Waltham, Massachusetts, 2005.

40. MATLAB software version 2011a, Mathworks Inc., Matick MA.

41. S. Gupte, O. Masound, R. F. K Martin, N.P. Papanikolopoulos, "Detection and Classification of Vehicles," *IEEE Transaction on Intelligent Transportation Systems*, Vol.3 No.1, March 2002, pp. 37-47.

42. L. Wang, N.H.C. Yung, "Extraction of Moving Objects From Their Background Based on Multiple Adaptive Thresholds and Boundary Evaluation," *IEEE Transaction on Intelligent Transportation System*s, Vol. 11, No. 1, March 2010, pp. 40-51.

43. J.W. Hsieh, S.H. Yu, Y.S. Chen, W.F. Hu, "Automatic Traffic Surveillance System for Vehicle Tracking and Classification," *IEEE Transaction on Intelligent Transportation System*, Vol. 7, No. 2, June 2006, pp.175-187.

44. W. W. L. Lam, N.H.C. Hung, "Highly accurate texture-based vehicle segmentation method," *Optical Engineering*, Vol.43, No. 3, March 2004, pp. 591-603.

45. M. Heikkila, M. Pietikainen, "A texture-based method for modeling the background and detecting moving objects," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 28, No. 4, April 2006, pp. 657-662.

46. H.S Lai, N. H.C Yung, "A fast and Accurate Scoreboard Algorithm for Estimating Stationary Backgrounds in an Image Sequence," *Proc. IEEE Intl. Symp. Circuit Syst*., Vol. 4, 1998, pp.241-244, Monterey, CA.

47. N. M. Oliver, B. Rosario, A. P. Pentland, "A Bayesian Computer Vision System for Modeling Human Interactions," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 8, August 2000, pp.831-843.

48. C. Stauffer, W.E.L Grimson, "Adaptive background mixture models for real-time tracking," *IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 2, 1999, pp. 246-252, Ft. Collins, CO, USA.

49. I. Pavlidis, V. Morellas, P. Tsiamyrtzis, S. Harp, "Urban Surveillance Systems: From the Laboratory to the Commercial World," *Proc. IEEE*, Vol. 89, No. 10, pp. 1478-1496, Oct. 2001.

50. B. Yiu, K Wong, F. Chin, R. Chung, "Explicit Contour Model for Vehicle Tracking with Automatic Hypothesis Validation," *IEEE Intl. Conference on Image Processing*, 2005, Genoa, Italy.

51. T. N. Tan, G. D. Sullivan, K. D. Baker, "Model-based Localization and Recognition of Road Vehicles," *International Journal of Computer Vision*, Vol. 27, No.1, 1998, pp. 5-25.

52. D. Beymer, P. McLauchlan, B. Coifman, and J. Malik, "A real-time computer vision system for measuring traffic parameters," *IEEE Conference on Computer Vision and Pattern Recognition*, 1997, San Juan, Puerto Rico.

53. G. Welch, G. Bishop, *An Introduction to the Kalman filte*r, University of North Carolina, Chapel Hill, NC, http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html, accessed Feb. 2010.