Enhanced Monitoring and Planning of Network Infrastructure with Remote Data Collection

Contract DTOS59-10-H-00002

Final Report

Submitted by:

Dr. Mark Hickman
University of Arizona
Department of Civil Engineering and Engineering Mechanics
1209 E. Second Street
PO Box 210072
Tucson, AZ 85721-0072

Dr. Pitu Mirchandani
Arizona State University
School of Computing, Informatics, and Decision Systems Engineering
PO Box 878809
Tempe, AZ 85287-8809

Submitted to:

Caesar Singh
Program Manager
Research and Innovative Technology Administration
US DOT

Executive Summary

Introduction

The goal of the federal program supporting this project is to provide for commercial opportunities to develop and apply remote sensing technologies in the transportation industry. In support of this goal, this project developed a set of tools to for sensing of traffic from airborne imagery. The intent was to develop these tools to the point of demonstrating their practicality and utility in real-world situations.

Specifically, the project goal was to produce: (1) a completed software tool, called TRAVIS (Tracking and Registration of Airborne Video Image Sequences), which enhances the ability to collect macroscopic traffic flow data (speeds, densities, flows, etc.) and microscopic data (individual vehicle trajectories); (2) a set of methods to integrate ground-based and airborne sensor data into more accurate and more precise estimates of traffic speeds and densities and vehicle origin-destination behavior; and, (3) a set of methods that can be used to route and schedule airborne sensors to survey the road network and to detect and analyze traffic congestion, allowing the new capability for agencies to plan for traffic data collection using airborne sensors.

In producing these tools, one of the critical elements in the program was to establish their viability using practical examples. Hence, for each of these products, a formal and practical demonstration of their value was planned and executed: TRAVIS in Tucson and Tempe, Arizona; data integration methods in Phoenix, Arizona; and routing and scheduling of airborne sensors in Beaverton, Oregon. Through this development and demonstration, the project has met the desired goal of the federal program. More specific details of the project, and its execution, are described briefly in this executive summary, and in the full report.

Scope and Technical Objectives

Within the goals given above, the objectives of this project were to investigate the enhancement of current data sources with data from new, commercially- or otherwise publicly-available data from remote sensors for better monitoring of transportation infrastructure. In order to achieve such improvements, we undertook the development of methods and tools for data collection, analysis, and data fusion, and the validation of those methods and tools, through this project.

This project built on a wide variety of previous research by the lead investigators to design a transportation network monitoring program that would enhance traditional data sources with data from new remote sources such as cell-phone signals, aerial images and video, and vehicle-based GPS signals when available. The areas of research and development in this project included:

- ***Enhancing infrastructure monitoring and management with airborne imagery***.
  This activity has expanded previous work in several areas:

  (1) fusing airborne and other remotely sensed data with ground-based but location-specific traffic sensor data, to improve monitoring of the transportation network;

  (2) using the fused data for better traffic prediction, infrastructure use and condition information, and transportation planning;

  (3) furthering the use of airborne surveillance during emergencies, major incidents, evacuations and roadway construction activities; and

  (4) with collaborators from the DLR, enhancing methods for special event traffic management (e.g., Super Bowl, World Cup Soccer) using airborne imagery and data.

- ***Developing and enhancing "enabling" technologies for airborne data collection and model calibration***.
  A new imaging platform and image processing tools were developed in a proof-of-concept phase. There are still a few technical concepts that deserve further attention:

  (1) enhancing the recently developed prototype software at the UA to track individual vehicles from video sequences, and extract traffic data that is not otherwise available; and,

  (2) conducting statistical evaluation of remote measurements, such as error rates, frequency of type 1 and type 2 errors, and measurement noise statistics.

Technical Approach and Literature Review

Previous research within the National Consortium on Remote Sensing in Transportation – Flows (NCRST-F) has shown that location-based sensor data can be supplemented, or even in some cases replaced, by cost-effective use of remote sensing tools such as airborne and satellite imagery. To achieve these improvements, however, it is necessary to develop methods and tools to exploit the remotely sensed data.

The project team has developed prototype software to extract individual vehicle trajectories from aerial video. This software, called TRAVIS, can be used with aerial video to identify individual vehicles and their movement across consecutive images. By knowing the pixel coordinates and the approximate scale of the image, vehicle trajectories (in distance and time) can easily be determined. An underlying goal within this program was to continue refinements of this software, specifically to make it viable as a software tool on the Windows platform, and, to make the software more useful, to refine many of the vehicle detection and tracking tools in the software. We have also shown (in other parts of this project) that even partial trajectories taken from TRAVIS can be used in real time to do short-term forecasts of traffic conditions.

Ultimately, the completion of TRAVIS greatly expands the usefulness of airborne traffic data, and the utility of collecting both ground-based and remote sensors. Specifically for remotely sensed data, the completed TRAVIS software allows more traffic information to be extracted more quickly and at lower cost, and can now generate “products” that will be useful for transportation management and infrastructure planning.

TRAVIS and its Enhancements

TRAVIS Features

In the course of the activities under NCRST-F, the UA developed software, called TRAVIS (for Tracking and Registration of Airborne Video Image Sequences), which allows for the registration of aerial video and the tracking of individual vehicle movements within that video. The primary activity of TRAVIS involves the video image processing and trajectory processing. The process of data collection and processing in TRAVIS is depicted in Figure ES-1. The process begins with data collection over a roadway or traffic facility of interest. Once collected the TRAVIS software registers the video to a single image, identifies interesting features (vehicles) in the image sequence, and tracks these vehicles through the sequence. Once these pixel coordinates are known, these pixels can then be converted into “ground coordinates”, giving specific locations and times of vehicle locations.

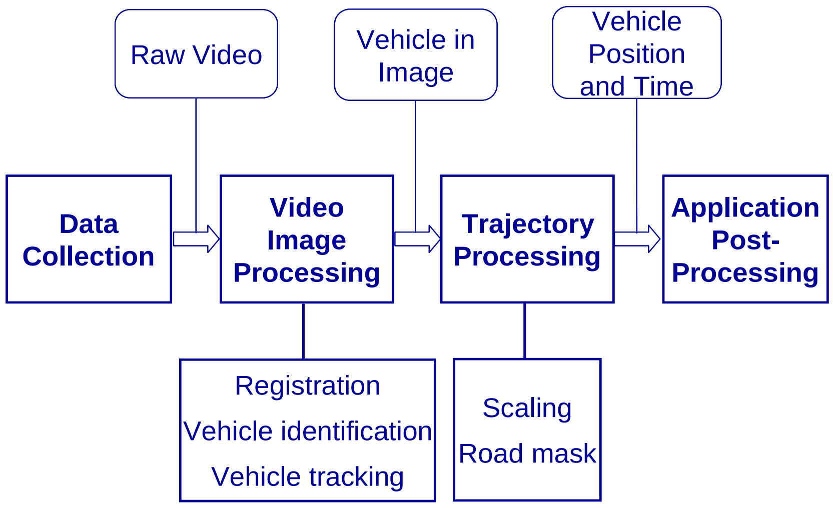


Figure ES-2: Process of Traffic Data Collection and Analysis in TRAVIS [3]

Scope of TRAVIS Enhancements

Several major enhancements of TRAVIS were enabled by this project:

- We enhanced this software with a new graphical user interface (GUI) which may be more flexible for input and operator control of TRAVIS. This includes: (1) a feature for manual identification of vehicles to be tracked in a set of imagery; (2) a window to specify parameters for the algorithms that detect and track vehicles in the imagery; and, (3) a help feature to assist users with the software. In addition to modifying the interface in Unix (the native platform for TRAVIS), we also made a Windows™ version of the code.
- Second, we have developed effective techniques of applying a "road mask" (i.e., defining the actual road boundaries within an image) to speed up the image processing. This mask is generated by noting the location of tracked features across a set of images. We then determine the most prominent direction of motion between these features, and identify the minimum and maximum of these features. Based on an estimated lane width, the road mask is placed outside the minimum and maximum lane locations. From this method, the TRAVIS software can focus on identifying and tracking vehicles in a specific "area of interest" (i.e., within the road boundaries), and the computational burden associated with the image processing is significantly reduced.
- Third, the older version of the TRAVIS software was effective at identifying only 60-80% of vehicles in an image. While this might be acceptable for some applications, it is clearly not sufficient if detailed vehicle trajectories are desired. We have developed more effective thresholds for the vehicle identification and tracking algorithms. These algorithms allow darker vehicles (which might otherwise blend with the pavement color), and also carefully

filter for noise in the imagery, to obtain identification and tracking rates of over 90%. These methods have been implemented without increasing the number or rate of incorrectly tracking other features.

- Fourth, a "dynamic" vehicle identification and tracking feature has been added to TRAVIS. This automatically detects new vehicles as they enter the image, and as the helicopter moves between different geographic areas. A single point of reference is used, but the dimensions of motion are continually updated to allow continuous tracking in the *x*- and *y*-directions. This feature means that vehicle trajectory processing can occur automatically across much longer video sequences. This can be very useful across long segments of freeways and arterials.
- Finally, we also experimented with software solution that allows us to register the helicopter imagery with existing geo-referenced imagery. That is, with satellite or ortho-rectified airborne imagery, we have shown that we can match this geo-referenced imagery to the helicopter imagery. This allows us to translate points in the vehicle trajectory into true ground coordinates. While the prototype has been developed in this project, it has not yet been integrated into the TRAVIS software. Work in this area is ongoing in a separate project, with the intent of finalizing its integration into TRAVIS in 2013.

With the completion of these enhancements, TRAVIS has become a viable tool for traffic agencies and transportation planners and engineers to analyze airborne imagery. The software now can output vehicle coordinates and trajectories over extended video image sequences. In turn, these data can then be used for detecting congestion, understanding driver behavior through vehicle interactions, and creating other data sources to support the calibration and validation of traffic simulation models.

Sensor Fusion of Airborne with In-ground Sensor Data

Another major task of the project was to develop methods to fuse data from airborne sensors with ground data. The initial focus was on freeway data. The traffic system is a highly dynamic and complex nonlinear system, even for a non-controlled freeway stretch. Traditionally, the data used for freeway traffic state estimation are collected through limited number of ground sensors located at some specific positions along the freeway stretch. This type of data has their intrinsic noise which would deteriorate the estimated results. In this research we focus on the exploration of how to utilize the high accuracy imagery for a freeway stretch collected by a surveillance helicopter to enhance traffic state estimation.

In this research we focus on the freeway traffic state estimation problem, regarding both macroscopic and mesoscopic levels, using data collected by both ground sensors and remote sensors. For our purposes, the evolution of a freeway traffic system is continuous in time and space, but researchers have found it very helpful to discretize it into segments and into discrete time. Matching the vehicles in a series of images would yield partial trajectories or sparse points along their trajectories. Optimally filling the gaps between these measurements is the objective of our mesoscopic estimator. [By *mesoscopic*, we mean that we can capture macroscopic properties such as speed, density, and flow, but without losing the information from individual vehicle trajectories.]

In the *mesoscopic* level defined in our research we focus on the temporal-spatial relationship for each individual vehicle while all microscopic behaviors like lane changing are neglected. From this view one only cares about at each time step where the vehicle is located in the freeway stretch. We have modeled the vehicle position as a state estimation problem, integrating the ground-based measures of vehicle location and speed with the actual vehicle locations observed sporadically by an airborne platform. These measures can be integrated using a Kalman Filter (KF), a common method for traffic state estimation. The parameters of the Kalman Filter are determined using least squares estimation.

An experimental design was generated, using vehicle trajectory data from the traffic simulation tool VISSIM. The estimation algorithm was realized in MATLAB and the least-squares model in IBM-ILOG OPL software. We find that the performance of the state estimation method is good in general, although it is of course sensitive to the time increment in the state estimation and to the spatial resolution of the vehicle positions.

Observation Logistics for Airborne Vehicles

A separate research initiative within the scope of this program dealt with logistics of the airborne vehicles (AV), such as helicopters, airplanes or unmanned aerial vehicles (UAVs), operating cameras to remotely monitor traffic. In particular, logistics problems were defined for two cases: (1) when the fleet of AVs has the capacity to monitor all required locations to be observed and the objective is to minimize the cost of monitoring the locations; and, (2) when the fleet has more points to monitor than it is capable of monitoring, in which case the objective considered was to maximize the number of observation locations.

These problems were formulated as optimization models. Their solution complexity was evaluated and heuristic solution approaches were studied. In general, these problems are variants of existing vehicle routing problems, which are notoriously complicated and require heuristics for their solution.

In this context, a simple case study was addressed to demonstrate the framework for solving these logistics problems. A prototypical version of the software, named SIM-AIR, was implemented to simulate and visualize the routing/scheduling of a fleet of AVs for the case study network. The research team was fortunate to have a small network preliminarily developed by team member Yi-Chang Chiu and simulated using DynusT, a traffic simulation software developed at the University of Arizona. The network was for traffic flow for Beaverton, Oregon. After the traffic simulation was run, traffic congestion times and locations were obtained, where congestion was defined using traffic density for each given roadway segment. Then, based on the locations and times of congestion, the logistics of the helicopter routing were determined, using a routing heuristic implemented in Google Earth.

Potential Future Applications

The project has also sought to solve practical problems for the industry, in the sense that these products in turn could feed future applications. This has been achieved through feedback from the Technical Advisory Panel for the project, which was sought both through a kickoff meeting with the panel (held April 27, 2010) and in soliciting feedback from panel members (electronically) on the value of airborne data products (conducted in summer 2011). The feedback from the technical advisory group was that the airborne traffic data would have value, inasmuch as we are able to capture extended sequences (5-10 min) over reasonable lengths of roadway (at least 0.5-mile) in the imagery field of view. From the view of our technical advisory panel, such imagery allows for more sufficient traffic data to be of use for: (1) microscopic traffic simulation modeling; (2) for understanding patterns of traffic congestion over larger geographic areas where aircraft may be able to travel quickly; and (3) estimating traffic states (levels of congestion) on specific facilities, when integrated with other ground-collected traffic data.

Table of Contents

Introduction

The goal of the federal program supporting this project is to provide for commercial opportunities to develop and apply remote sensing technologies in the transportation industry. In support of this goal, this project developed a set of tools to for sensing of traffic from airborne imagery. The intent was to develop these tools to the point of demonstrating their practicality and utility in real-world situations.

Specifically, the project goal was to produce: (1) a completed software tool, called TRAVIS (Tracking and Registration of Airborne Video Image Sequences), which enhances the ability to collect macroscopic traffic flow data (speeds, densities, flows, etc.) and microscopic data (individual vehicle trajectories); (2) a set of methods to integrate ground-based and airborne sensor data into more accurate and more precise estimates of traffic speeds and densities and vehicle origin-destination behavior; and, (3) a set of methods that can be used to route and schedule airborne sensors to survey the road network and to detect and analyze traffic congestion, allowing the new capability for agencies to plan for traffic data collection using airborne sensors.

In producing these tools, one of the critical elements in the program was to establish their viability using practical examples. Hence, for each of these products, a formal and practical demonstration of their value was planned and executed: TRAVIS in Tucson and Tempe, Arizona; data integration methods in Phoenix, Arizona; and routing and scheduling of airborne sensors in Beaverton, Oregon. Through this development and demonstration, the project has met the desired goal of the federal program. More specific details of the project, and its execution, are described in this full report.

Scope and Technical Objectives

The objectives of this project were to investigate the enhancement of current data sources with data from new, commercially- or otherwise publicly-available data from remote sensors for better monitoring of transportation infrastructure. Current data sources to monitor traffic and its associated infrastructure include census data, manually collected infrastructure maintenance data, vehicle detector data, signage, traffic controls, incident reports and other information commonly used by transportation planners and managers. While these data are useful, they can be supplemented by a wide variety of data from new remote sensors, such as cell-phone signals, aerial images and videos, and vehicle-based GPS signals when available. Such data can significantly enhance our understanding of infrastructure condition and the traffic patterns that lead to congestion on that infrastructure.

In order to achieve such improvements, we undertook the development of methods and tools for data collection, analysis, and data fusion, and the validation of those methods and tools, through this project.

This work builds on preliminary research conducted as part of the National Consortium on Remote Sensing in Transportation – Flows (NCRST-F) between 2000 and 2004. Both lead investigators on this project participated directly in this consortium. This prior research demonstrated that remote sensing and data fusion methods are viable means of collecting both aggregate traffic measures (delay, density, flow, speed, etc.) as well as individual vehicle trajectories (see references [1-12]). Both satellite and airborne sensing platforms were shown to be useful; the particular airborne platforms have included airplanes, helicopters, and even Unmanned Aerial Vehicles (UAVs).

Similar research has integrated traffic flow data from airborne imagery into formal data collection programs, where the data from the imagery is fused with ground detector data to enhance the estimates of traffic flows. The Ohio State University, as part of the NCRST-F, has effectively shown the value of satellite imagery in collecting traffic flow data with no fixed ground-based data collection equipment (i.e., loop detectors or temporary tube counters). Also, Skycomp, Inc., an independent commercial company, has also shown that traffic data, such as queue lengths on arterials and densities on freeways, can be collected from aircraft and fused with ground sensor data to estimate the level of service on these roadways.

The University of Arizona (UA) and the Arizona State University (ASU), through the research conducted with the NCRST-F, has also shown that traffic data can be enhanced through real-time data from airborne imagery. This can include traffic counts and queue estimates at intersections, densities on freeways, and even vehicle speed estimates. Additional research conducted at the Advanced Transportation and Logistics Algorithms and Systems (ATLAS) Center/Laboratory at UA/ASU is investigating the value of airborne sensor data when ground-based sensor data are not available or difficult to interpret, especially during a major incident or during major evacuations.

This project built on this preliminary research to design a formal transportation network monitoring program that would enhance traditional data sources with data from new remote sources such as cell-phone signals, aerial images and video, and vehicle-based GPS signals when available. The areas of research and development in the proposed project include:

- ***Enhancing infrastructure monitoring and management with airborne imagery***.

  In the NCRST-F and ATLAS Center research, a helicopter-based platform was developed that can collect video and still imagery, and process the imagery in near real time. Outputs from this data processing include queue lengths at intersections and also individual and collective vehicle speeds. This activity has been expanded in this program by development and enhancement of these capabilities in several areas:

(1) to fuse the airborne and other remotely sensed data, which has greater spatial coverage, with ground-based but location-specific traffic sensor data, to improve monitoring of the transportation network;

(2) to use the fused data for better traffic prediction, infrastructure use and condition information, and transportation planning;

(3) to further the current preliminary research conducted at ATLAS on the use of airborne surveillance during emergencies, major incidents, evacuations and roadway construction activities; and

(4) with collaborators from DLR, to enhance methods for special event traffic management (e.g., Super Bowl, World Cup Soccer) using airborne imagery and data.

- ***Developing and enhancing "enabling" technologies for airborne data collection and model calibration*.**

In the NCRST-F, a new imaging platform and image processing tools were developed primarily to test their feasibility; this was a proof-of-concept phase. There are still a few technical concepts that deserve further attention. With the rapid development of image processing techniques, and the "miniaturization" of airborne platforms and sensors, the following objectives will be pursued in this project:

(1) to enhance the recently developed prototype software at the UA to track individual vehicles from video sequences, and extract traffic data that is not otherwise available; and,

(2) to conduct statistical evaluation of remote measurements, such as error rates, frequency of type 1 and type 2 errors, and measurement noise statistics.

Technical Approach and Literature Review

Imagery collected from airborne platforms has long been used to document the evolution of traffic congestion across extended areas. Since the late 1990's, many important ideas, new and old, on the use of airborne imagery for traffic analysis were investigated and tested in the field by two of the co-principal investigators through their membership in NCRST-F. Initially, the investigators developed methods to determine traffic performance and mobility measures. Traffic characteristics such as speed, density, intersection queue lengths and delays, average annual daily traffic, and other level-of-service measures could be determined directly from airborne imagery (see references [13-16]). Such spatial traffic measures can be used to determine the level of service of existing infrastructure and the locations of bottlenecks or other locations where modernizing and/or expanding roadway infrastructure may be needed. In addition, by using these data to calibrate and validate traffic simulation models, one can more directly analyze the performance of scenarios where there are changes in roadway infrastructure.

To date, traffic engineers and transportation planners have relied heavily on fixed location-based sensors to monitor infrastructure performance and traffic congestion. Previous research within the NCRST-F has shown that these location-based sensor data can be supplemented, or even in some cases replaced, by cost-effective use of remote sensing tools such as airborne and satellite imagery. To achieve these improvements, however, it is necessary to develop methods and tools to exploit the remotely sensed data. For example, while manual methods may still be used to analyze the airborne imagery, there have been considerable gains in computing power and image processing techniques over the last ten years.

This technical progress has led to greater interest in using airborne imagery for a variety of civilian purposes, including traffic monitoring. As part of the work within the NCRST-F, UA researchers developed prototype image processing techniques that could be used to automatically determine traffic flow measures from the imagery collected. Researchers at the UA developed software to automatically estimate queue lengths at intersections, to estimate vehicle speeds, and to estimate vehicle flows and densities. These activities are documented in the reports from the NCRST-F as well as peer-reviewed papers in academic journals (e.g., references [1-12]).

The Project Team has also postulated and “architectured” approaches to automatically, in real-time, geo-reference images from remote cameras for managing traffic, especially when other sensors are absent or disabled. This is done by integrating the imagery with information on the height, location, and orientation of the camera. Using these camera data, in combination with a geographic representation (latitude-longitude) of the area to be monitored, leads to an explicit geo-referencing of the road and vehicle locations. Absolute values of vehicle positions, speeds, accelerations, decelerations, and lane changes can be determined. This was proven in experiments conducted jointly with the Ohio State University, with a very high-resolution inertial measurement unit (IMU), geodetic-quality (high resolution) GPS receivers, and industrial high-resolution cameras (see [11]). However, initial experiments using lower-cost and lower-resolution equipment show some promise in geo-referencing the airborne imagery.

In addition, the project team has developed prototype software to extract individual vehicle trajectories from aerial video. From the video, we are able to identify individual vehicles and their movement across consecutive images. By knowing the pixel coordinates and the approximate scale of the image, vehicle trajectories (in distance and time) can easily be determined. We have demonstrated this technique using video from Tucson, Phoenix, and Seattle, and have presented these results at technical conferences (see references [1] and [3] as examples). An underlying goal within this proposed program is to continue refinements of this software and develop data sets of vehicle trajectories that can be used for calibration and validation of microscopic traffic simulation models. Such simulation models can then be used for investigating possible roadway infrastructure improvements or changes, to better explain existing and likely future traffic conditions.

In the meantime, research collaborators in Europe, namely Prof. Reinhart Kühne and Martin Ruhé from the German Aerospace Agency (DLR) in Berlin, have developed an integrated platform, referred to as ANTAR ("Airborne Traffic Analyzer"), that can be flown on fixed-wing planes and helicopters to acquire and transmit images at 5 frames per second. They have also developed associated software, called TrafficFinder, which can process the imagery from ANTAR, in real-time, providing traffic parameters such as speeds and densities, and individual vehicle trajectories (see [17-19]). ANTAR includes a GPS receiver (with a GPS antenna), an inertial measurement unit (IMU), a high-resolution camera, an infrared camera, and a microwave transmitter to transmit the acquired images along with the measurements from the GPS and IMU. The transmitted images are quickly referenced to a GIS-based street map, and TrafficFinder finds the vehicles in the images and obtains speeds and densities. The problem with this system is that it is expensive, especially because of the IMU (see Figure 1). An objective of this project was to incorporate some approaches of TrafficFinder in the cheaper (i.e., more affordable to traffic agencies) system being developed at UA that does not include an IMU but uses software approaches to geo-reference images, in order to obtain traffic measures in real-time.

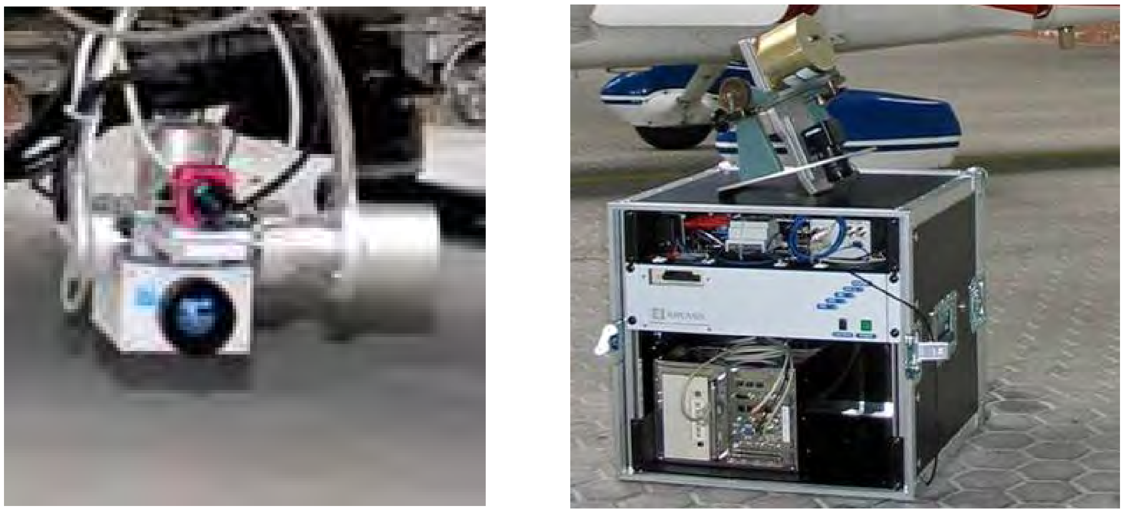


Figure 1: ANTAR Imaging System: Camera Platform and Processing Unit (DLR)

An example of the types of vehicle trajectories available from this system is shown below in Figure 2. In this case, the trajectories are relatively short because of the movement of the helicopter during the data collection. Nonetheless, even these partial trajectories can be used in real time to do short-term forecasts of traffic conditions. The local speed, density, and flow estimates are determined from the individual trajectories; these are then translated through a fundamental diagram to determine the movement of shock waves in time and space along the roadway. With the shock wave analysis, vehicle movements are then projected into the future, accounting for decelerations and accelerations due to the shock waves. Hence, a short-term traffic forecast can be created [17-19].

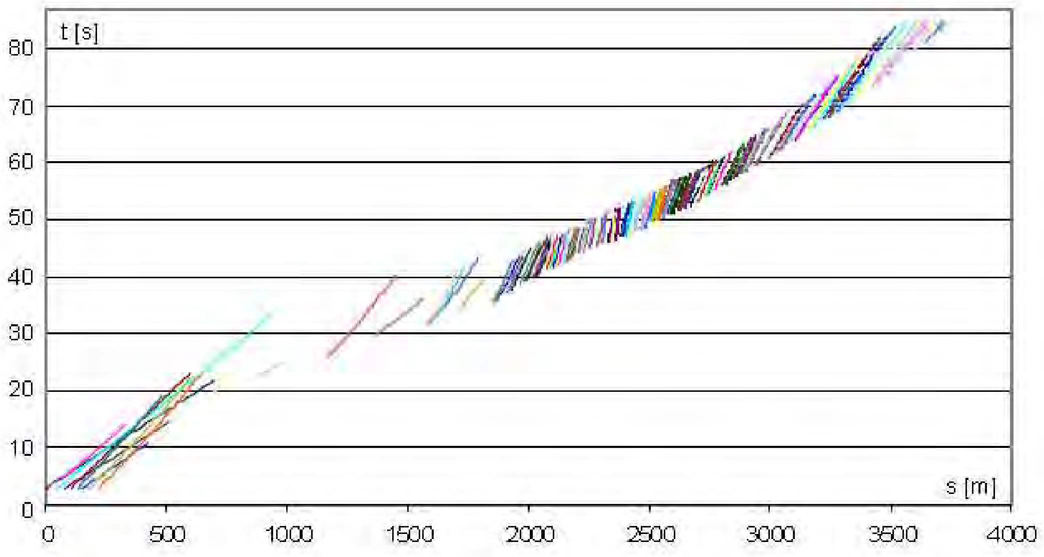


Figure 2: Vehicle Trajectories [19]

A team of researchers at the Technical University of Delft has developed a similar system [20-23]. Their technique likewise uses digital video collected from a helicopter, with the objective of capturing microscopic traffic data. Their method differs from the Arizona approach in that it specifically uses an automated technique to generate and match ground control points in the imagery, in order to register the imagery. Vehicle detection is done by first generating a "background" image (the roadway without vehicle traffic), and vehicles are detected as differences from this reference image. Correspondence of vehicles from one image to the next is also achieved through a coarse matching algorithm, made easier since the imagery is taken at relatively high frequency. Finally, the conversion of vehicle coordinates in the image to vehicle coordinates in the real world appears to have been done manually, through specific lateral and longitudinal references on the roadway. More recent research by this same team has examined finer resolution of the vehicle trajectories through the use of a Kalman filtering technique.

It is important to note here that most research and practical traffic studies have used fixed-wing airplanes, helicopters, or blimps as the means to carry personnel and photographic equipment, which in itself can be quite expensive. Only a few small studies to date have examined the use of smaller, lightweight, and low-cost unmanned aircraft systems (UAS's, formerly unmanned aerial vehicles or UAV's) to record traffic imagery (see [24],[25]). These studies have shown the potential use of UAS's to collect data for freeway density measurement, queue measurement at intersections, origin-destination studies, and even parking lot utilization.

Ultimately, our research results could greatly expand the usefulness of the traffic data that is presently being collected from both ground-based and remote sensors. Specifically for remotely sensed data, this research will allow more traffic information to be extracted more quickly and at lower cost, and will lead to "products" that will be useful for transportation management and infrastructure planning.

Primary Project Activities and Feedback

In addition to the work required to generate the "products" described above, several critical activities feeding all of the work in this project were required. First, a Technical Advisory Board (TAB) was formed to advise the work. Members of the TAB included:

- Scott Nodes, Arizona Department of Transportation
- Tom Buick, Independent Consultant
- Sarath Joshua, Maricopa Association of Governments
- Aichong Sun, Pima Association of Governments
- David Gibson, Federal Highway Administration
- Jim Schoen, Kittelson and Associates

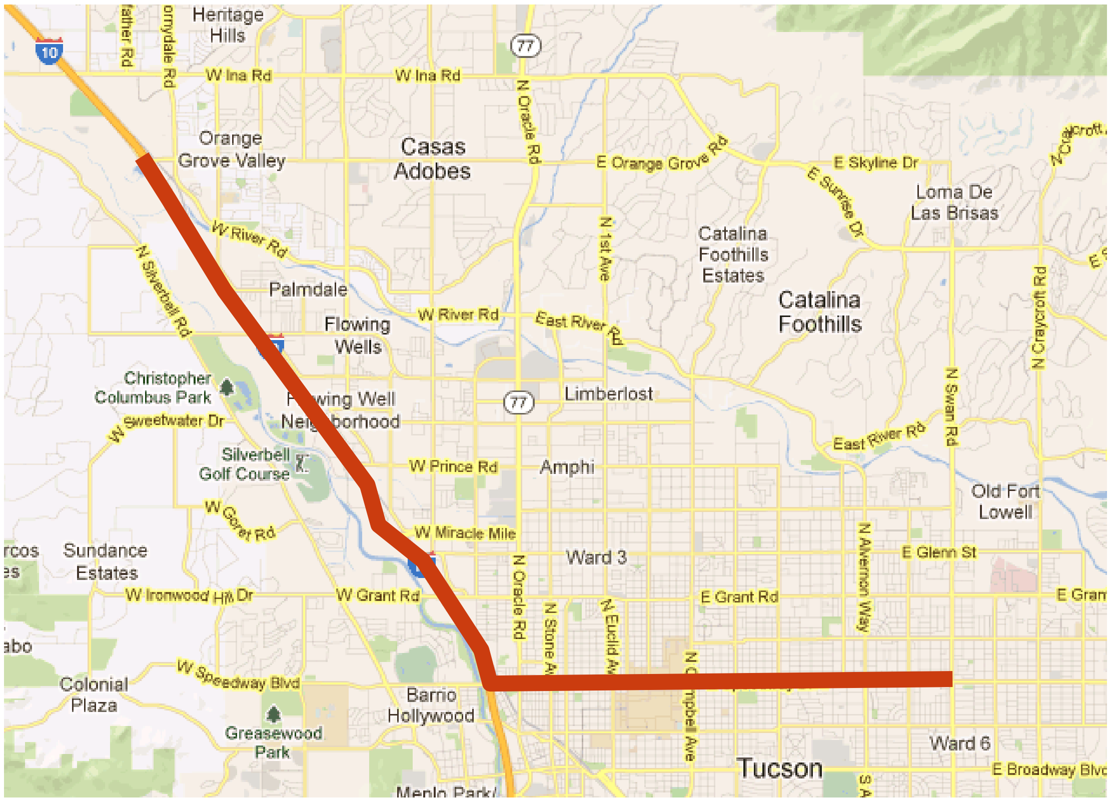
An initial kickoff meeting with the TAB was conducted on April 27, 2010. At this meeting, the scope of the project was discussed, and the relevance of the research and development work to transportation practice was investigated. A separate set of minutes, given in Appendix A, describes the results of that kickoff meeting. Essentially, the primary feedback from the TAB included:

- A review of the project deliverables, to ensure that products would be relevant to the larger transportation community. Major deliverables include data sets from the first major work activity, software tools, and other project documentation and journal papers.
- A discussion of special events that could possibly serve as opportunities to collect data on atypical traffic patterns.
- The question of value in measuring emissions from an airborne platform, compared to existing ground-based "sniffers". [Subsequently, this item was dropped from the scope of the project.]

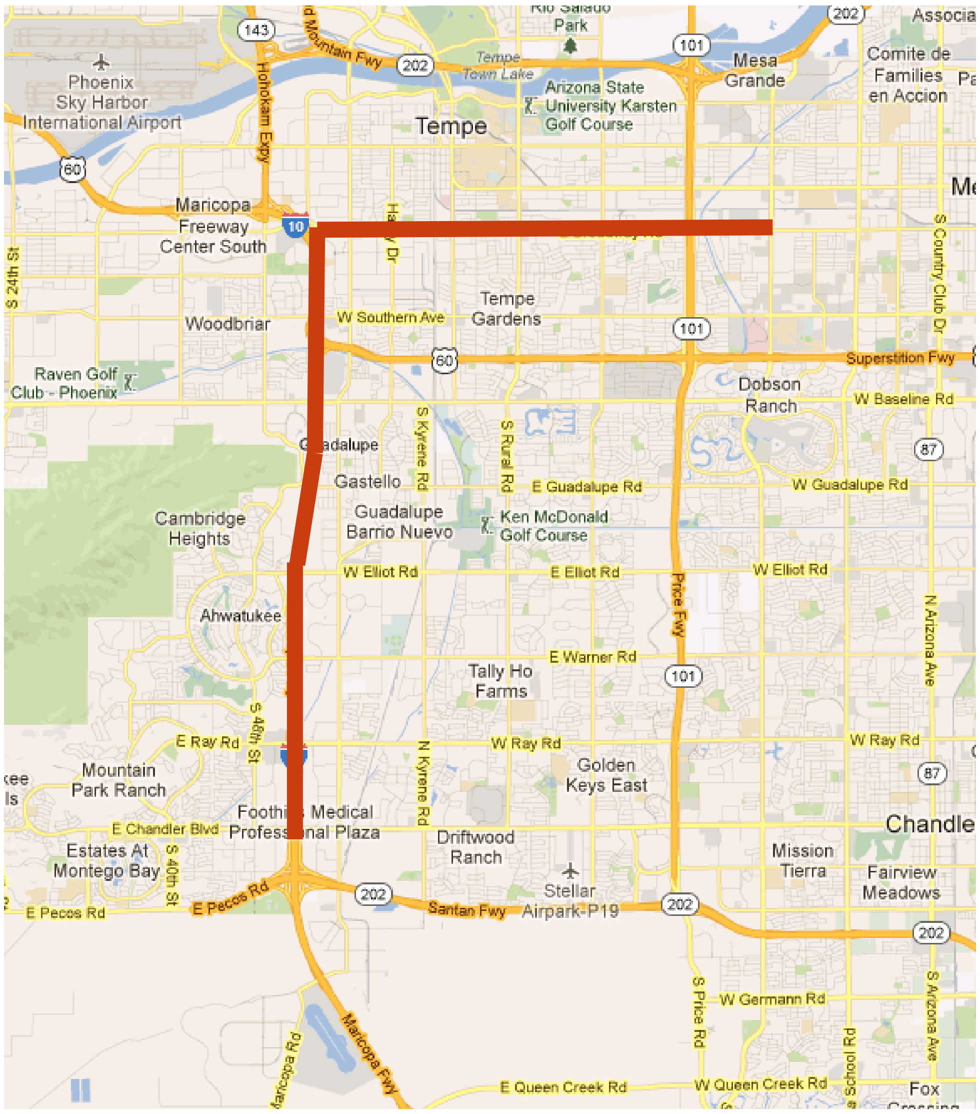
- The role of the second main activity in generating a real-time traffic management capability. [This topic was not directly investigated in this project.]
- The possible coordination of traffic data collection with the next regional traffic bottleneck study. While this is possible, our emphasis is more on the generation of traffic data from imagery, and the automated analysis of imagery to generate traffic data. We have less emphasis on producing specific performance measures.

During the work on the products, additional feedback from the TAB was solicited in the summer of 2011, primarily centered on the value of different traffic data products that could be generated by airborne imagery, as processed by the TRAVIS software and used in traffic congestion, state estimation, and traffic condition forecasting. This solicitation was intended to provide critical feedback to the team for the initial, baseline capabilities of TRAVIS that might make the software viable in practice. In summary, the feedback gave several useful suggestions. Succinctly, the TAB felt that airborne traffic data would have value inasmuch as we are able to capture extended sequences (5-10 min) over reasonable lengths of roadway (at least 0.5-mile) in the imagery field of view. Such imagery allows for more sufficient traffic data to be of use for: (1) microscopic traffic simulation modeling; (2) for understanding patterns of traffic congestion over larger geographic areas where aircraft may be able to travel quickly; and (3) estimating traffic states (levels of congestion) on specific facilities, when integrated with other ground-collected traffic data.

To this end, the major data collection activity for this project included an AM and a PM flight over major roadways in Tucson, Arizona, on January 11, 2011. In each case, about 2 hours of video (30 frames per second, 720x480 pixels) were collected using a standard off-the-shelf digital camcorder affixed to the skid of a helicopter. Simultaneously, high-resolution imagery (at 16 megapixels) was also collected, at an approximate frame rate of 4 frames per second, using a high-resolution surveillance camera. The major roadways included Speedway Boulevard, a critical east-west arterial serving both local and longer-distance traffic in Tucson, and Interstate 10, a critical east-west freeway through the center of Tucson. These roadways are highlighted in the following map.



A second data collection activity, including an AM and a PM flight over major roadways in Tempe and Chandler, Arizona, was conducted on February 1, 2011. Again, in each case, about 2 hours of video (30 fps) were collected using a digital camcorder on the skid of a helicopter. Simultaneously, high-resolution imagery (at 16 megapixels) was also collected, at 4 fps, using a high-resolution surveillance camera. The major roadways included Broadway Boulevard, a critical east-west arterial serving Tempe, and Interstate 10 between the Broadway curve and Chandler Boulevard, in Tempe and Chandler, Arizona. These roadways are highlighted in the following map.



Both data collection activities (Jan 11 and Feb 1) were used and analyzed as part of this project. The imagery from these data collection activities supported both the enhancements to TRAVIS and the estimation of traffic states using both airborne- and ground-based data sources.

Outline of Report

Following this introductory chapter, an extensive discussion of the enhancements to the TRAVIS software is provided in the second chapter. This chapter includes the improvements to the TRAVIS graphical user interface (GUI) to improve its utility and to make it compatible with the Windows operating system. Other enhancements to the algorithms in TRAVIS are also discussed in this chapter, including: (1) the application of a road mask to improve the speed of the image processing and vehicle identification; (2) the improvement of the vehicle identification algorithms to increase the fraction of vehicles correctly identified; (3) the use of dynamic image processing to allow vehicle tracking over extended sequences when the airborne platform is moving over larger special areas; and (4) the development of a software algorithm to georeference the airborne images to a georeferenced mapping image (e.g., from a satellite or airborne mapping platform.

The third chapter discusses the fusion of vehicle trajectory information from airborne platforms with other in-ground sensor data to estimate and to project the short-term evolution of traffic conditions. Finally, the fourth chapter contains a report on the routing and scheduling of airborne platforms to collect traffic information. This includes several formulations and solution methods for the routing and scheduling problem, depending on the objective of the airborne data collection. These chapters summarize the major extensions and proof of concept for airborne data collection methods.

TRAVIS and its Enhancements

In the course of the activities under the National Consortium on Remote Sensing of Transportation – Flows (NCRST-F), the UA developed software, called TRAVIS (for Tracking and Registration of Airborne Video Image Sequences), which allows for the registration of aerial video and the tracking of individual vehicle movements within that video. The TRAVIS software has been used to track individual vehicles in video segments up to 90 seconds long (~2700 images at 30 frames per second), when the imagery is centered over a specific location for that full duration. In addition, the software also has a post-processing step that uses the image scale or other geo-referencing capabilities to produce vehicle trajectories in space and time. The primary activity of TRAVIS involves the video image processing and trajectory processing.

TRAVIS Features

The process of data collection and processing in TRAVIS is depicted in Figure 3. The process begins with data collection over a roadway or traffic facility of interest. A commercial digital video camera, mounted on the aircraft to collect imagery from near nadir, collects the imagery in the form of a digital video. Once collected, the raw digital video is then processed through TRAVIS. As described in more detail below, the software registers the video to a single image, identifies interesting features (vehicles) in the image sequence, and tracks these vehicles through the sequence. The results of this step are pixel coordinates of the vehicle in the image. Once these coordinates are known, these pixels can then be converted into parameters of individual vehicle trajectories. Specifically, the pixel coordinates can be converted into "ground" coordinates through various means: an image scale, geo-referencing to existing geographic coordinates, or some other method.

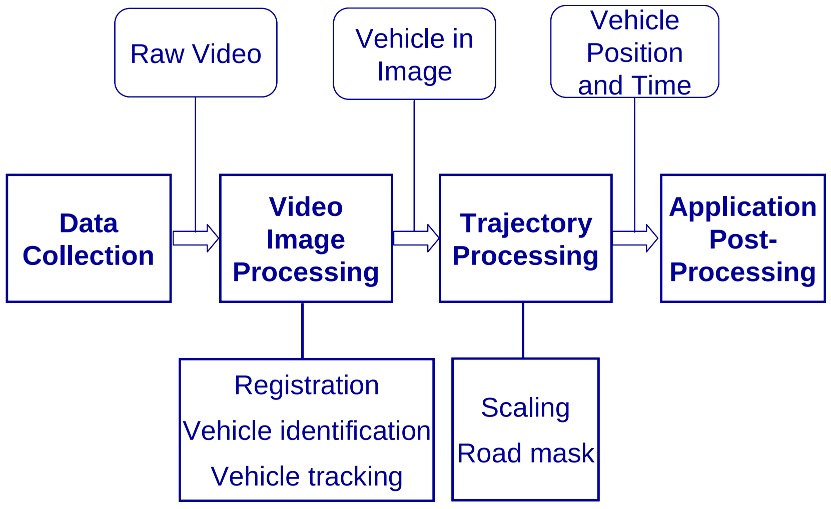


Figure 3: Process of Traffic Data Collection and Analysis in TRAVIS [3]

In addition, the position of the vehicle relative to the roadway can be determined by using a "road mask", describing the geometry of the roadway within the image. From the roadway mask, the direction and the specific lane of a vehicle can be determined. Finally, the vehicle position and time (i.e., the trajectory) can then be used for specific traffic applications. Each of the steps is described in more detail below.

Data Collection

The video data collection conducted to date has used a commercial off-the-shelf digital camcorder with a pixel resolution of 720x480 pixels, collecting images at 30 frames (images) per second. In many of the experiments, we have flown at heights ranging from 500 ft (160 m) to 2000 ft (650 m) above the surface of the roadway. With a typical focal length on the camcorder, this allows us to capture distances on the ground approximately equal to the height; i.e., from 160 m to 650 m on the ground, in the longer direction of the camcorder's field of view. Generally, this provides a scale ranging from 1-4 pixels per meter (or 0.3-1.3 pixels per foot), or commonly around 2 pixels per meter with 360 m in the field of view. Such a scale allows sufficient pixels for each vehicle in the image so that the vehicle can be easily identified as a "good" feature in the image processing software. However, it also provides a large enough field of view to simultaneously capture numerous vehicles over a roadway segment of interest. More details on the data collection platform can be found in [1].

Video Image Processing

The input to the video image processing is the sequence of images from the video, which can be large. In the video image processing in TRAVIS, the sequence of images in the video (up to 30 frames per second) are first registered. This step of registration identifies common features from one image that reappear in the second image. These are determined as "ground control points" in the imagery, using the registration technique in the Kanadi-Lucas-Tomasi algorithm. Once the matching is performed, the algorithm performs an affine transformation of the coordinates of the second image to the first. This removes any movement of the airborne platform (both systematic movement and jitter) from the image sequence, allowing the images to appear as if they were taken from a fixed point of view.

In the second step, two adjacent images are then subtracted. This subtraction operation leaves only moving features in the image. Once these features are identified, the algorithm then identifies the "blobs" in each individual image that generate these moving features. Within each image, the highest difference in the red, green, and blue bands in the image is found, and used to threshold for "blobs" in the images. Simple dilation is also performed, and the objects are then screened using simple heuristics (i.e., using the approximate size and shape of a vehicle). The "blobs" are then labeled in each image, using connected component analysis. It is these features that are then tracked throughout the sequence. An example of adjacent images with connected components is shown in Figure 4.

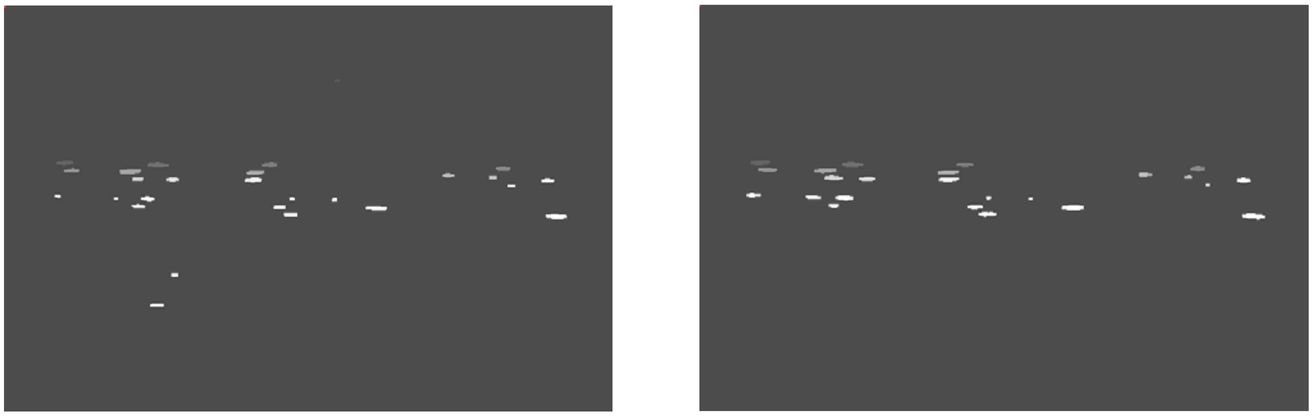

(a) (b)

Figure 4: Connected Components in Adjacent Images (a) and (b)

The third step of vehicle tracking provides both short- and long-term filtering of these “blobs” to eliminate noise in the image, but to retain the “blobs” that represent moving vehicles. A short-term tracker is used to detect new moving vehicles in the sequence, and to screen out “blobs” that are not actually vehicles. Some small blobs can be seen in Figure 4 that are removed by the short-term tracker. At the same time, new vehicles entering the image can also be tracked: an example of a new vehicle is that on the lower left side Figure 4. The long-term tracker, on the other hand, maintains the coordinates of the vehicle over time. It also allows the tracking of vehicles even when they are not detected for short periods of time (e.g., the vehicles stops moving for a short time, or it is not detected in a short sequence of images).

The net result from the tracker is a sequence of pixel coordinates of vehicles as they are tracked through the image sequence. An example of the video output, in the form of vehicle coordinates, is shown in Figure 3. This illustrates the text output, in the form of pixel coordinates for each object (the rows) and for each frame in the video sequence (the columns).

Trajectory Processing

The subsequent processing of the vehicle coordinates is shown as “Trajectory Processing” in Figure 3. The objective is to convert the vehicle “location” in pixels to ground coordinates. The most direct method is to scale the pixels to ground coordinates, using a simple transform (e.g., a linear transform of X meters per pixel). More sophisticated non-linear models can also be used, depending on the camera height, focal length and field of view.

Scope of TRAVIS Enhancements

Several major shortcomings of TRAVIS could be noted above, and fit in the scope of the enhancements enabled by this project:

- We enhanced this software with a new graphical user interface (GUI) which may be more flexible for input and operator control of TRAVIS. In addition to modifying the interface in Unix (the native platform for TRAVIS), we also made a Windows™ version of the code.
- Second, we have developed effective techniques of applying a “road mask” (i.e., defining the actual road surface within an image) to speed up the image processing. If the TRAVIS software can focus on “areas of interest” (i.e., on the roads), the computational burden associated with the image processing can be reduced.
- Third, the current software is effective at identifying 60-80 % of vehicles in an image. While this might be acceptable for some applications, it is clearly not sufficient if detailed vehicle trajectories are desired. So, improvements in vehicle identification are desired.
- Fourth, a manual identification feature was added to the software, allowing a user to specify which vehicles should be tracked. This is useful if new vehicles enter the field of view. However, this can only be done once at the beginning of the video sequence, not later as other vehicles enter the image, or not if the helicopter moves between different areas that are not based on the original location. If the vehicle identification and tracking can be made “dynamic,” following the trajectory of the helicopter and allowing other vehicles to enter the field of view, vehicle trajectory processing can occur for much longer video sequences than the current 90 second limit.

GUI Improvements and Windows

One of the major improvements for TRAVIS was to implement the software on a Windows platform. The code was successfully ported from Unix, its original platform, to Windows. The code retains much of its original structure, using C code for the main program and maintaining a Windows-compatible version of TCL/TK for the graphical user interface.

Modes

The basic GUI allows for two modes of operation: Auto Detect, in which the software itself identifies vehicles in the image (see Figure 5), and Manual Detect, in which the user may click on individual vehicles in an image that he/she wants the software to track (see Figure 6). The primary difference between the two is the ability to specifically pinpoint vehicles and their characteristics in Manual Detect mode, which is not available in Auto Detect mode. In Figure 6, the interface includes buttons for Help, Type (vehicle type), and VOInfo.

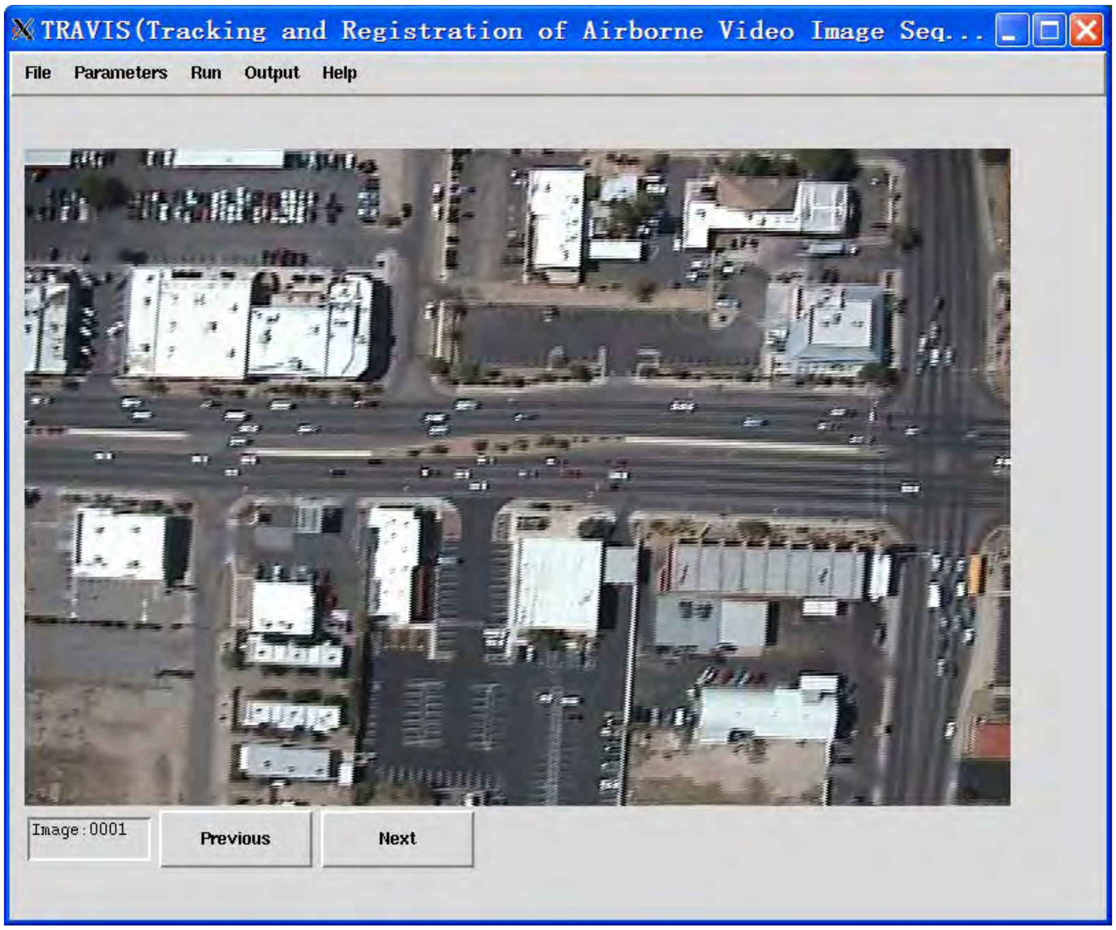

TRAVIS(Tracking and Registration of Airborne Video Image Seq...
File
Parameters
Run
Output
Help
Image:0001
Previous
Next

Figure 5: Basic Interface for Auto Detect Mode

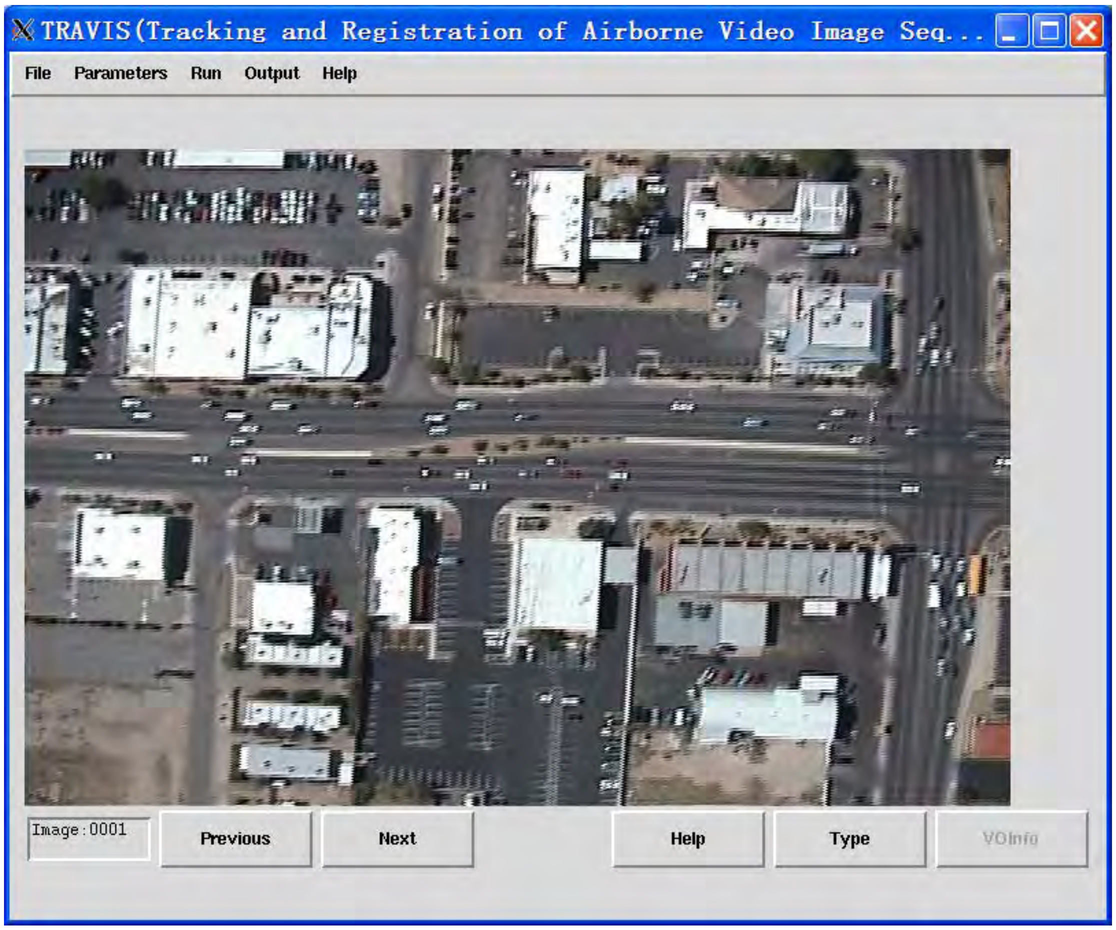


Figure 6: Basic Interface for Manual Detect Mode. Note the use of the "Help", "Type", and "VOInfo" buttons on the lower right, in comparison with the Auto Detect Mode.

Manual Detection Options

When the Manual Detection Mode is chosen, the user also has the options of identifying specific object characteristics, for each vehicle object identified in the image. These include two factors that would appear to have the highest value in determining microscopic vehicle behavior: the vehicle type (or vehicle classification), and the traffic lane in which the vehicle is located. These are specified using the ***Type*** button in the main screen. The information can be queried using the ***VOInfo*** (**v**ehicle **o**bject **info**rmation) button on the main screen.

Menu Systems

File

The options under the File menu (see Figure 7) include specification of the location of the input images ("Open Image"), the location of output images that have been registered and from which the vehicles are tracked ("Output Image"), and the key to exit the program.

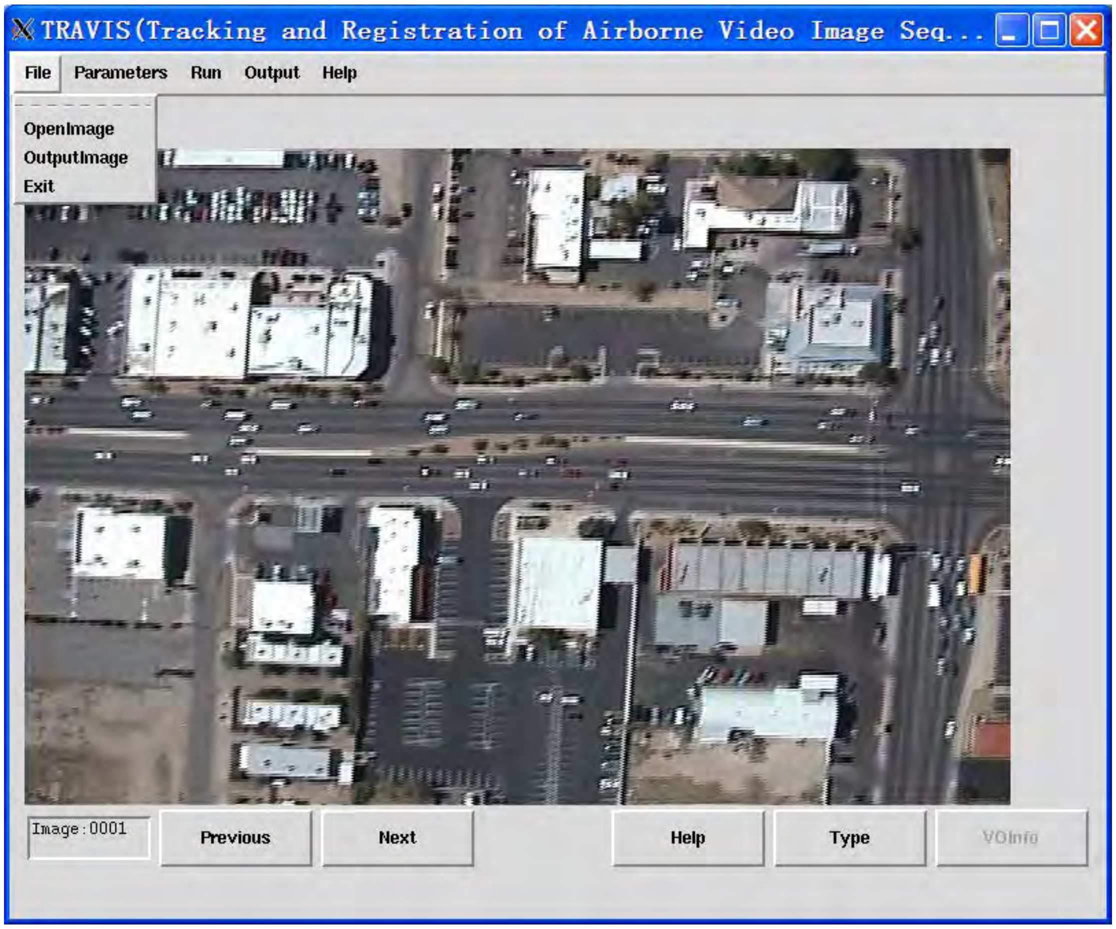


Figure 7: The File Menu

Parameters

The parameters for the model are specified under the "Parameters" menu (see Figure 8). These include:

- *Detect Mode*: Within this mode, we can specify either the Auto Detect or Manual Detect format.
- *Start Frame*: This is the frame number of the first frame in the image sequence.
- *End Frame*: This is the frame number of the last frame in the image sequence.
- *Tracking Box Height*: This is the height (vertical pixel count) of the box used to indicate a vehicle that has been tracked.
- *Tracking Box Width*: This is the width (horizontal pixel count) of the box used to indicate a vehicle that has been tracked.

- *Gap*: This is the number of successive frames used to identify a vehicle (vs. other noise in the image) that should be tracked over the long term. E.g., if a object is tracked for *Gap* frames, then it will be tracked as a long-term object.
- *Scale Dependent Variables*: These parameters allow the user to specify characteristics associated with the image, but reflecting elements of the vehicle identification and tracking process that may depend on the scale of the image. The dialog box for these variables, shown in Figure 9, allows the user to adjust the parameters of:
    - o The blob size (pixels)
    - o Car size (in pixels) for the car length and width;
    - o The minimum car length and width (ThreshLength and ThreshWidth);
    - o The threshold grey value, differentiating the vehicle pixel values from the underlying pavement; and,
    - o The number of images in the sequence to continue tracking a blob, without motion, until it is no longer tracked.

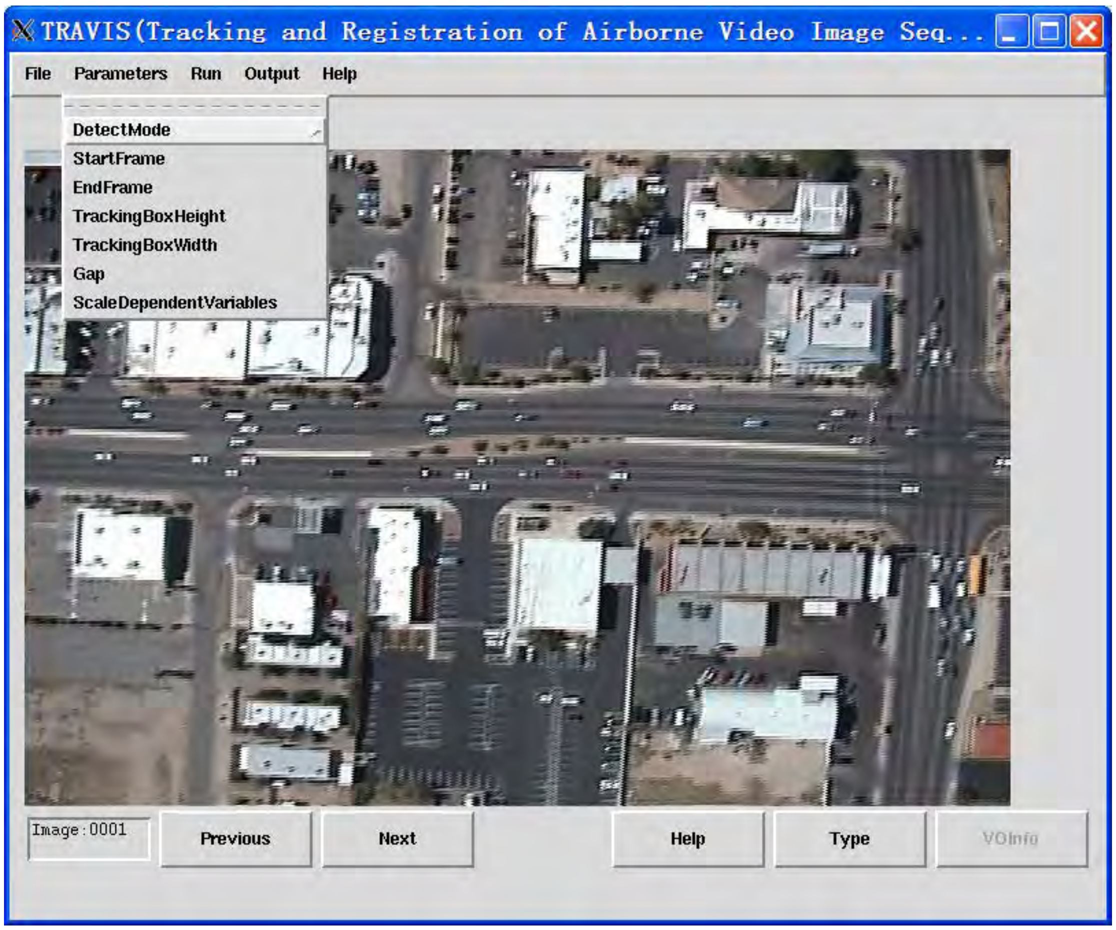
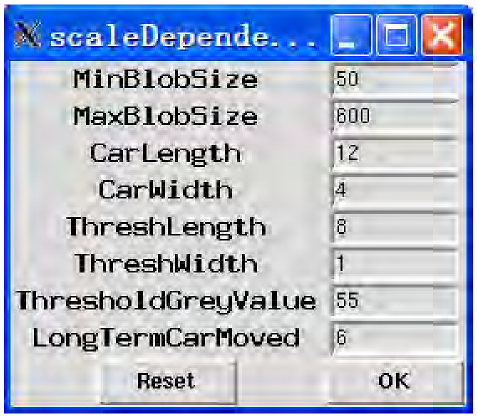


Figure 8: The Parameters Menu Options



scaleDepende...

MinBlobSize	50
MaxBlobSize	600
CarLength	12
CarWidth	4
ThreshLength	8
ThreshWidth	1
ThresholdGreyValue	55
LongTermCarMoved	6

Figure 9: Scale-Dependent Parameters for Identification and Tracking

Run

Running the model is performed through the Run menu. The options within that menu option include the path name containing all the input imagery, and then the command to run the registration and tracking. This is shown in Figure 10.

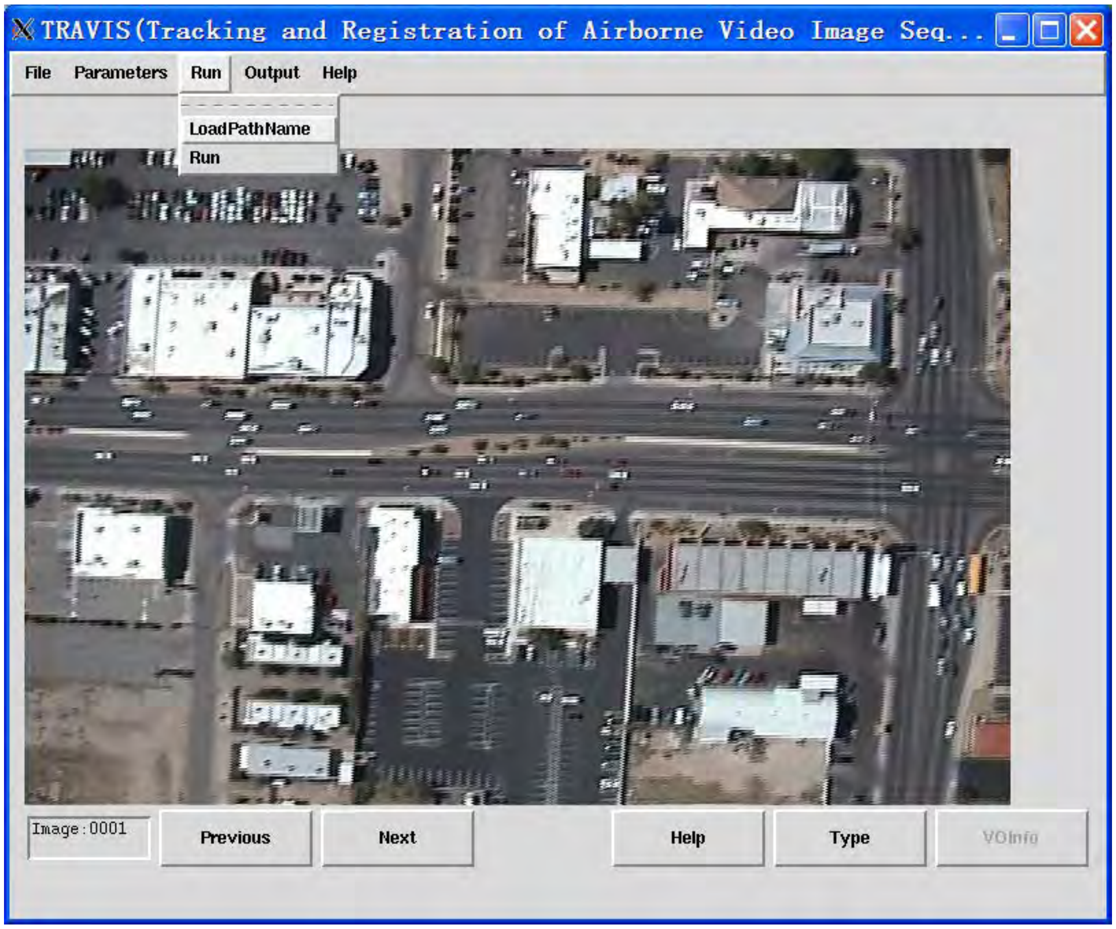


Figure 10: Options in the Run Menu

Output

The Output menu includes options to view the final imagery, including frames around each vehicle that is successfully tracked. These images can be viewed individually. This is illustrated in Figure 11. Separately, outside the TRAVIS software directly, these images may be recombined into a video sequence to illustrate the tracking in normal video.

The other feature is VOInfo, which allows the user to query vehicle object information, with manual vehicle detection and tracking.

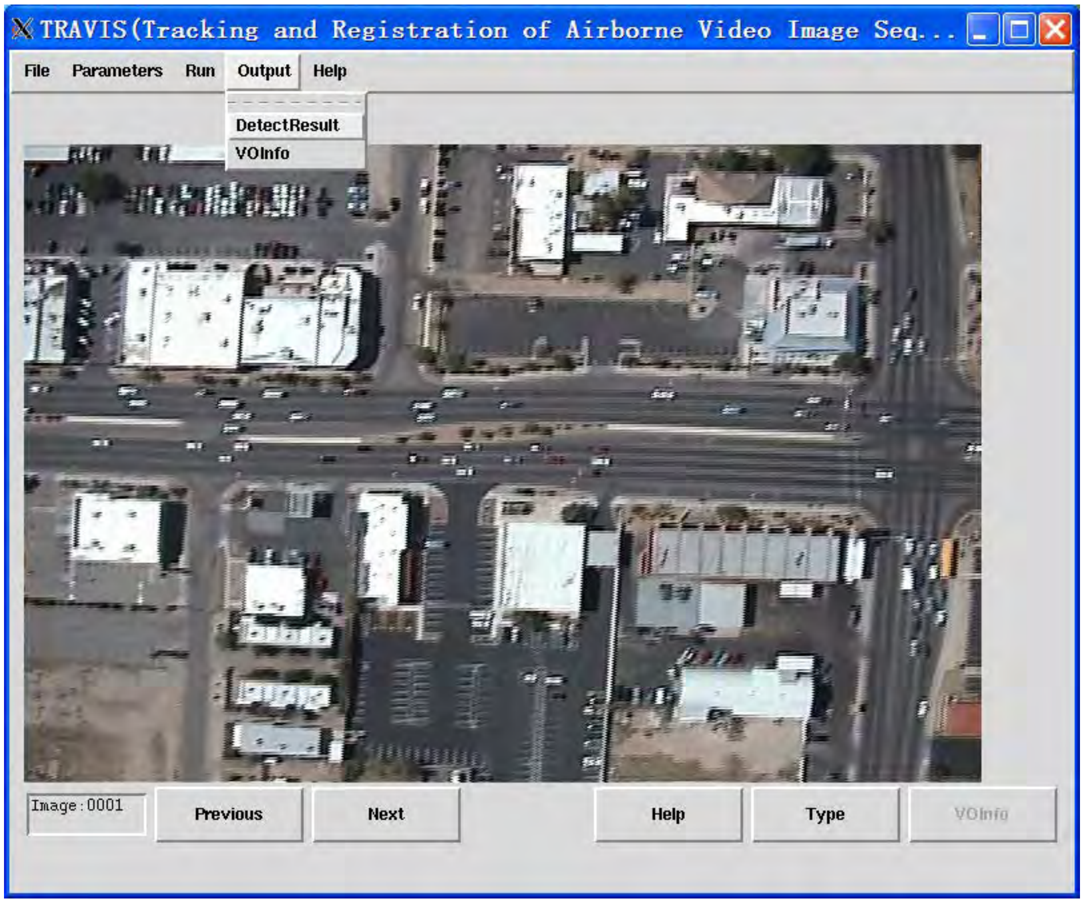
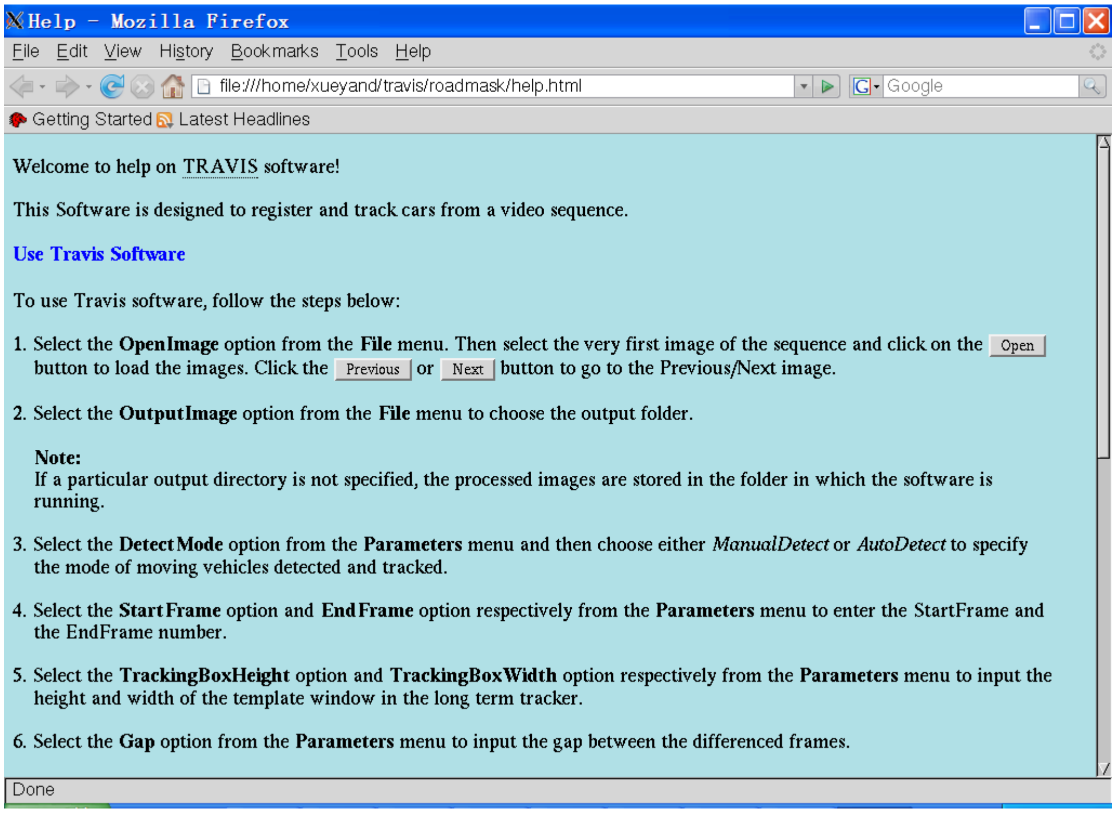


Figure 11: Output Menu Options

Help

Finally, the TRAVIS software includes a simple help feature that guides the user through various menu options and through the steps to run the software. This is illustrated in Figure 12.


Help - Mozilla Firefox
File Edit View History Bookmarks Tools Help
file:///home/xueyand/travis/roadmask/help.html
Google
Getting Started
Latest Headlines
Done

Welcome to help on TRAVIS software!

This Software is designed to register and track cars from a video sequence.

Use Travis Software

To use Travis software, follow the steps below:

1. Select the **OpenImage** option from the **File** menu. Then select the very first image of the sequence and click on the Open button to load the images. Click the Previous or Next button to go to the Previous/Next image.

2. Select the **OutputImage** option from the **File** menu to choose the output folder.

   Note:
   If a particular output directory is not specified, the processed images are stored in the folder in which the software is running.

3. Select the **DetectMode** option from the **Parameters** menu and then choose either *ManualDetect* or *AutoDetect* to specify the mode of moving vehicles detected and tracked.

4. Select the **StartFrame** option and **EndFrame** option respectively from the **Parameters** menu to enter the StartFrame and the EndFrame number.

5. Select the **TrackingBoxHeight** option and **TrackingBoxWidth** option respectively from the **Parameters** menu to input the height and width of the template window in the long term tracker.

6. Select the **Gap** option from the **Parameters** menu to input the gap between the differenced frames.

Figure 12: The Help Screen

Road Mask

Our method for creating a road mask from the existing airborne imagery involves several steps. We assume that there are sufficient vehicles on the lanes closest to the boundaries of the roadway of interest. Vehicle positions are smoothed using Kalman filtering. We then use the dominant slope of lines connecting every two vehicles to identify the general direction of the road boundary, assuming that the lanes and vehicles run parallel to the edge of the roadway. Finally, we create the road mask, according to the positions of vehicles and the general direction of the road boundary. More details on each of these steps are given below..

Smooth the Vehicle Position Using Kalman Filtering

The output of TRAVIS is a sequence of noisy measurements of vehicle pixel coordinates as they are tracked through the image sequence. If desired, Kalman filtering is available to smooth the vehicle trajectories. A Kalman filter has two steps: (1) to determine the system state at time k+1 based on the system state at time k and the state transition matrix; and, (2) to update the system state at time k+1 to reflect the information in the noisy measurement of the system state at time k+1. For position, speed, and acceleration, the Kalman filter resembles the following [31]:

Prediction:

$$S_{(k+1|k)} = \Phi * S_{(k|k)} \tag{1}$$

in which

$$\Phi = \begin{bmatrix} 1 & \tau & \tau^2/2 \\ 0 & 1 & \tau \\ 0 & 0 & 1 \end{bmatrix}$$

$$S_{(k)} = \begin{bmatrix} x_{(k)} & v_{(k)} & a_{(k)} \end{bmatrix}^T$$

In this case, the state of the system is described in terms of a vehicle's position, speed and acceleration. Its update depends on the time step $\boldsymbol{\tau}$, as given in the transition matrix $\boldsymbol{\Phi}$**.**

Update/filtering:

$$S_{(k+1|k+1)} = S_{(k+1|k)} + K_k * (Y_{(k+1)} - C^T * S_{(k+1|k)}) \tag{2}$$

$$C^T = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

$$K_k = \begin{bmatrix} \alpha & \beta/\tau & \gamma/\tau^2 \end{bmatrix}^T$$

with

$$\alpha = 1 - \theta^3 \qquad \beta = 1.5 * (1-\theta^2) * (1-\theta) \qquad \gamma = (1-\theta)^3$$

where θ is the maneuverability index, which determines the noise ratio between the system state equation and the observations ($0 < \theta < 1$), τ is the time difference between consecutive frames, and $Y_{(k+1)}$ is the noisy measurement of vehicle pixel coordinates. The parameters *α*, *β*, and *γ* are functions of the maneuverability index, reflecting the filtering of the error from the previous step for the new system state. More details on this process are given in [31].

Estimate the Slope of Roadway

The slope of the roadway is estimated based on the smoothed positions of vehicles tracked by the short-term and long-term tracker in TRAVIS. The detailed procedure is as follows:

Step 1: Calculate the slope of each line connecting every pair of vehicles, as shown in Figures 13 and 14.

Step 2: Divide the vehicle slopes into several groups by grouping close slopes into the same group.

Step 2.1: Sort all the slopes in an ascending order to get a sorted list of slopes, $S(i)$, i=1,2,…,N ($N = \frac{n(n-1)}{2}$, where *n* is the number of vehicles).

Step 2.2: Group the slopes based on the following criteria until the last slope $S(N)$ is grouped:

$$S(i) \le S_{ref}(j) + err1 \tag{3}$$

where

$$S_{ref}(j) = \begin{cases} S(1) & j = 1 \\ S(\sum_{r=1}^{j-1} GS(r) + 1) & j > 1 \end{cases}$$

$S(i)$: the ith slope in the sorted list of the slopes.
$S_{ref}(j)$: the reference slope of the jth group.
$err1$: a calibrated error.
$GS(r)$: the number of slopes in the rth group (the group size).

Step 3: Find the group *r* with the largest group size, *GS(r)*.

Step 4: Average the slopes in the group *r* to get the slope of the roadway, k_{road}.

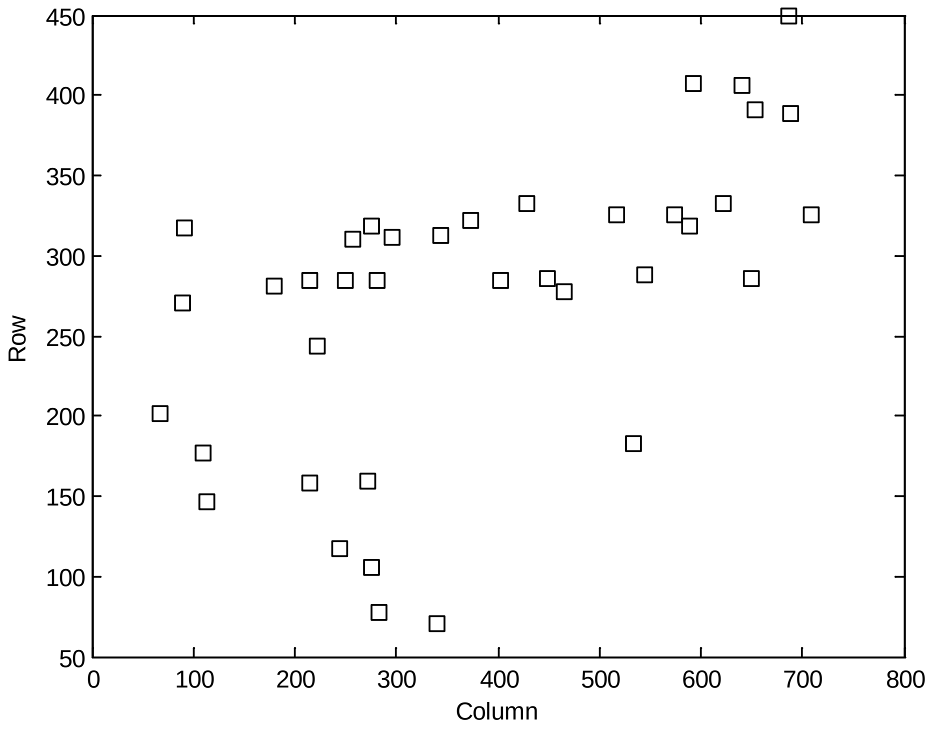


Figure 13: Detected and tracked vehicles in a single image

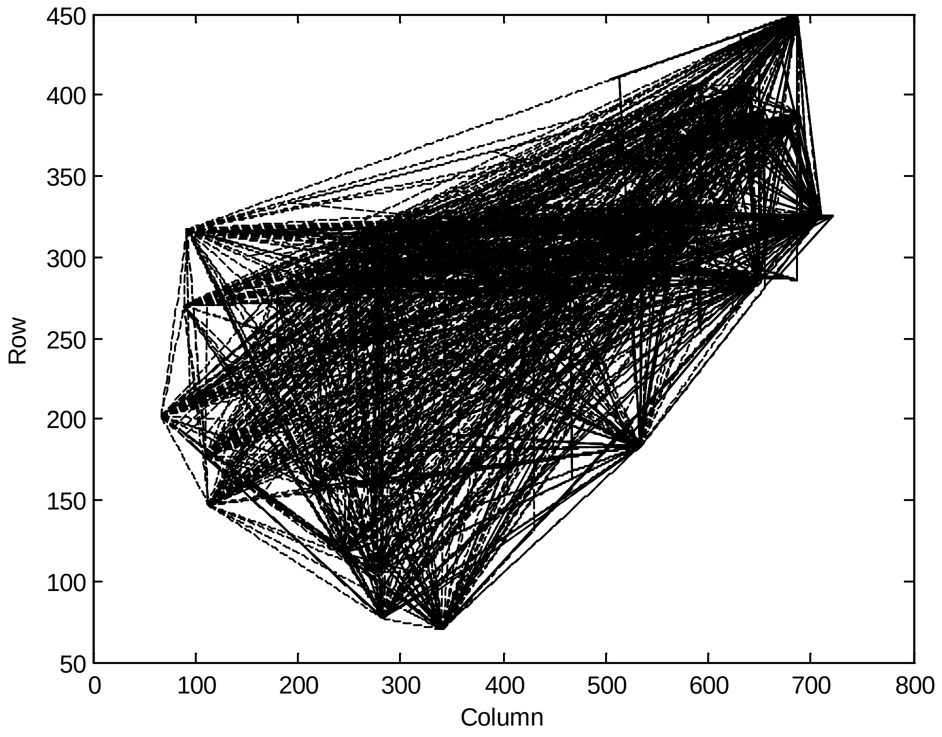


Figure 14: Lines connecting every two vehicles

Calculate the Boundary of the Roadway

The boundary of the roadway is also calculated according to the positions of the tracked vehicles, as follows:

Step 1: Pick a vehicle *i* randomly and calculate the slope of each line connecting this vehicle to all other vehicles *k (i,j)* , as shown in Figure 15.

Step 2: Find all vehicles in the same lane as vehicle *i* based on the following criteria:

$$k(i,j) \le k_{road} + err2 \text{ and } k(i,j) \ge k_{road} - err2 \tag{4}$$

where

k_{road} : the estimated slope of the roadway.

$err2$: a calibrated error.

Step 3: If there are more than two vehicles in the same lane, calculate the average *y* coordinate y_{ave}, and find the vehicle with *y* coordinate closest to y_{ave}, called vehicle *m*.

Step 4: Approximate the center of the lane *l* using k_{road} and the position of vehicle *m*, namely x_m and y_{m}:

$$l : y = k_{road} * (x - x_m) + y_{\text{m}} \tag{5}$$

Step 5: Calculate the intercepts of the center of the lane with the edges of the image, $x_{\min}$ and $x_{\max}$, as shown in Figure 16. In the registered frames that we use to illustrate this technique, $x_{\max}$ is equal to 720 (the width of the images), while $x_{\min}$ increases with the frame number. This occurs because the helicopter is moving to the right in this example: as frames are registered to an initial image, the left edge of the new images moves to the right. This is only an example; a similar procedure to determine $x_{\min}$ and $x_{\max}$ could be applied for any movement of the helicopter, in any direction.

Step 6: If there is no vehicle in the same lane as vehicle *i*, remove vehicle *i*. Otherwise, remove all the vehicles in the same lane as vehicle *i*.

Step 7: Repeat Step 1 ~ Step 6 as long as the number of remaining vehicles is greater than 1.

Step 8: Calculate the boundary of the roadway in the current frame *j*, $y^{\text{j}}_{\min}$ and $y^{\text{j}}_{\max}$, using (6):

$$\begin{aligned} y^{j}_{\min} &= \min(\min(i_{\min}), \min(i_{\max})) - C \\ y^{j}_{\max} &= \max(\max(i_{\min}), \max(i_{\max})) + C \end{aligned} \tag{6}$$

where $i_{\min}$ and $i_{\max}$ are arrays which store the intercepts of the lane slopes with the edges of the image, $x_{\min}$ and $x_{\max}$, respectively, and *C* is a calibrated constant, in which the distance from the roadway boundary to the center of the lane closest to the roadway boundary and the estimation error are bundled. In this way, the estimated roadway boundary is a distance *C* from the center of the nearest lane that has vehicles that are tracked.

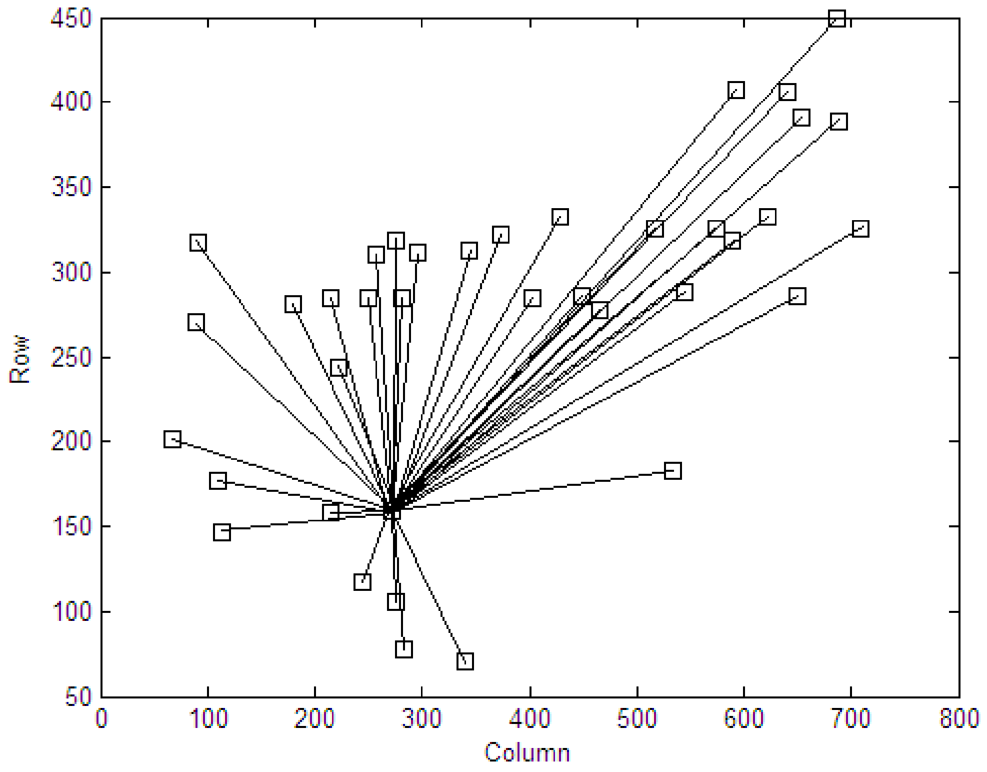


Figure 15: Lines connecting vehicle *i* to all other vehicles

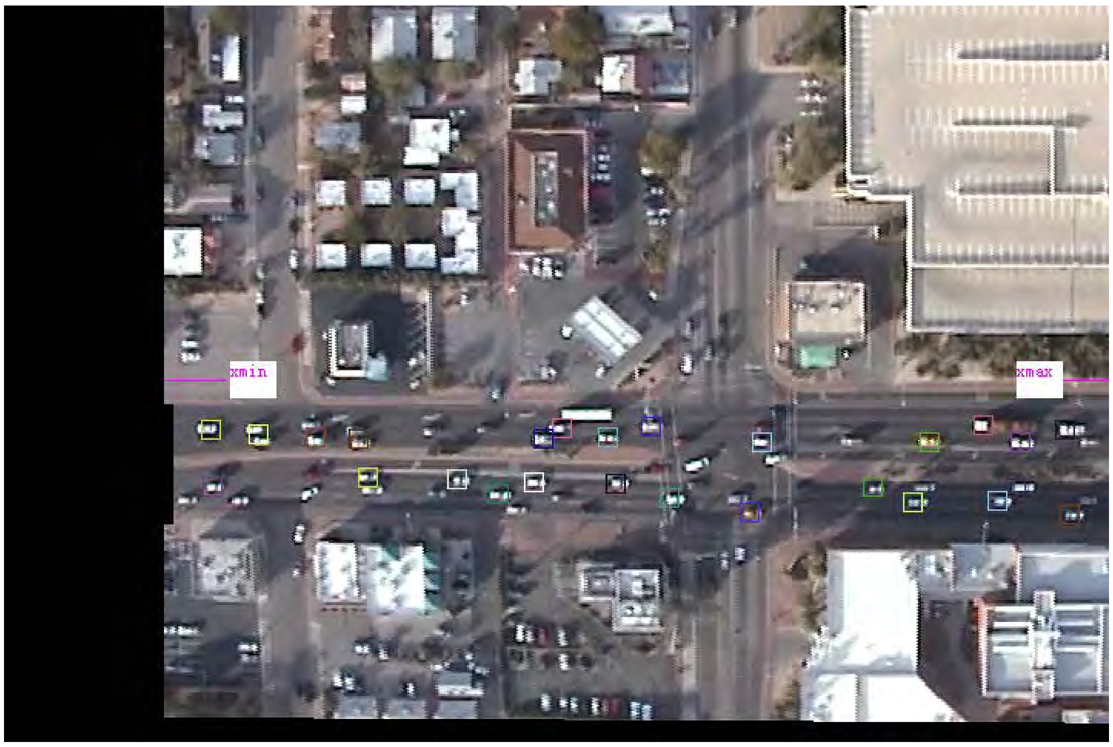


Figure 16: Edges of the image, $x_{\min}$ and $x_{\max}$

Applying the Road Mask in Vehicle Detection and Tracking

The road mask is applied in vehicle detection and tracking to remove false detections and to reduce the computation time. We define a “cycle” of images for which the road mask is unchanged; that is, there is insufficient change in the field of view during this “cycle” to warrant re-estimating the roadway boundary. The road mask is updated every cycle, and the same road mask is used within one cycle. For our purposes, a fixed cycle of 8 frames is used, although this could be varied, depending on the speed of the helicopter, which affects the change in the field of view. The number of 8 frames in the examples here is chosen to be consistent with previous work [26], and matches well with a relatively low helicopter speed of 20-30 mph over an arterial. Figure 17 shows the flow chart applying the road mask in vehicle detection and tracking.

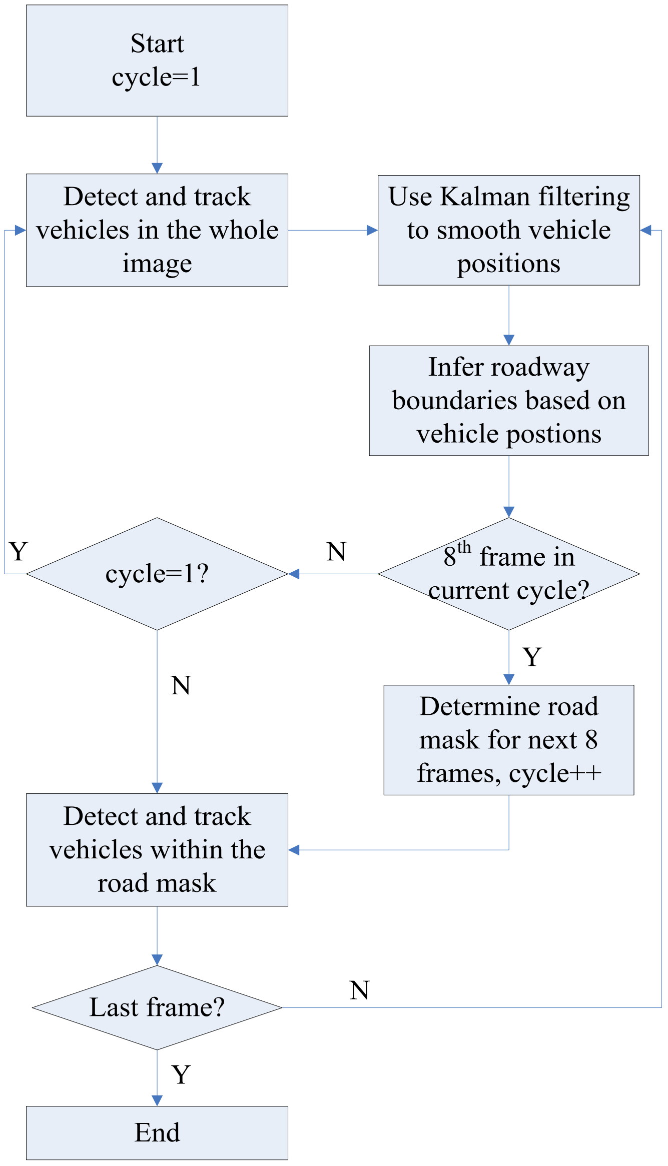

Start
cycle=1
Detect and track vehicles in the whole image
Use Kalman filtering to smooth vehicle positions
Infer roadway boundaries based on vehicle postions
8th frame in current cycle?
N
Y
cycle=1?
Y
N
Determine road mask for next 8 frames, cycle++
Detect and track vehicles within the road mask
Last frame?
N
Y
End

Figure 17: Flow chart applying the road mask in vehicle detection and tracking

The procedure is as follows:

Step 1: If cycle=1, detect and track vehicles using the short term and long term tracker in the whole image (from [26]). If cycle > 1, only detect and track vehicles within the road mask.

Step 2: Use Kalman filtering to smooth the vehicle positions, if necessary.

Step 3: Calculate the roadway boundaries in every frame based on the positions of tracked vehicles.

Step 4: Repeat Step 1 ~ Step 3 for one cycle (e.g., 8 frames).

Step 5: Determine the road mask boundaries based on the calculated roadway boundaries in the last cycle (e.g., the last 8 frames), using (7).

$$\begin{aligned} y_{\min} &= \min(y_{\min}^{j}) \qquad j = 1, 2, 3, \ldots, M \\ y_{\max} &= \max(y_{\max}^{j}) \qquad j = 1, 2, 3, \ldots, M \end{aligned} \tag{7}$$

where M is the number of frames in one cycle, e.g., 8 frames.

Step 6: Repeat Step 1 ~ Step 5 until all images are processed.

Figure 18 shows an example of the estimated boundaries of the road mask, $y_{\min}$ and $y_{\max}$.

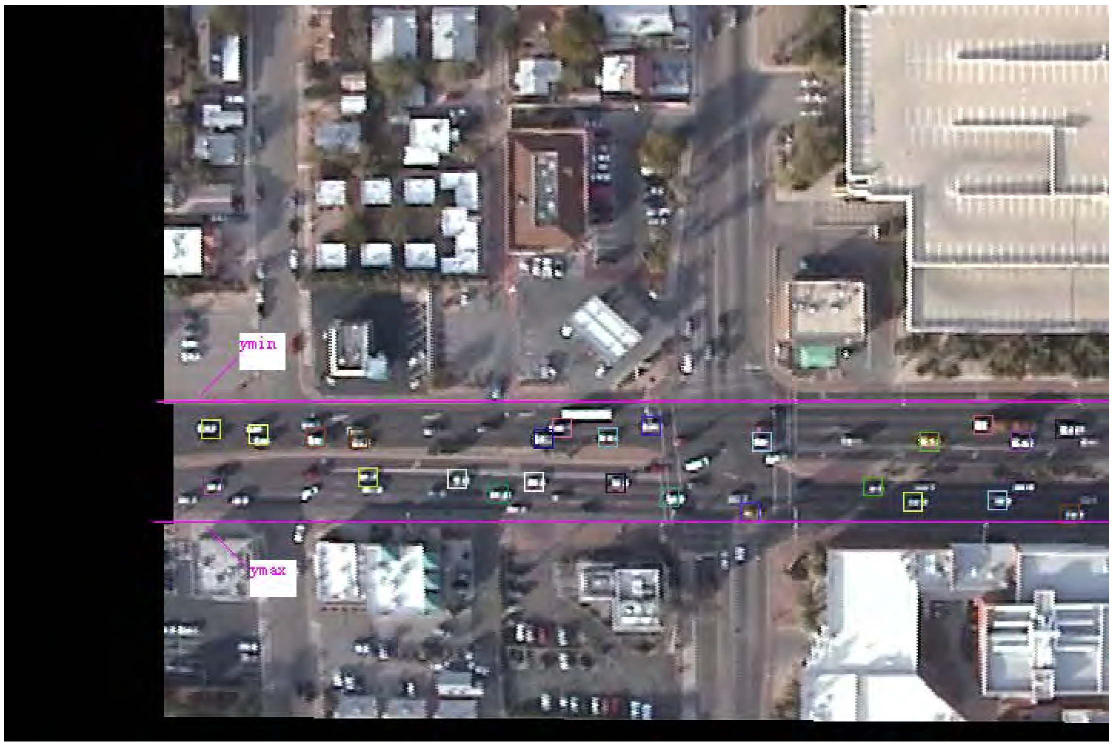


Figure 18: Estimated Boundaries of the Road Mask, $y_{\min}$ and $y_{\max}$

Experimental Results Using the Road Mask

Experimental Design

A set of images were collected in January 2011, from a helicopter traveling over Speedway Boulevard in Tucson, AZ. A sample of 300 frames was selected for testing using the road mask. These images were first converted to *.ppm format and then fed into TRAVIS, which registered the image sequences to an initial common reference frame, detected vehicles in the images, and tracked the vehicles through the image sequence [26]. Two experiments were conducted: one applying the road mask in the vehicle detection and tracking process (RMVDT), and the other without applying the road mask (VDT). The estimated roadway boundaries were compared with the actual roadway boundaries obtained manually from the images. The correct detection and tracking rate (CDTR) and the computation time were compared. There are two types of errors we can reduce: one is to reduce the number of undetected vehicles, which is referred to a Type I error, and the other is to decrease the number of misdetected vehicles, which is referred to a Type II error. CDTR is related to a Type II error: reducing the number (or percentage) of detected vehicles outside the roadway of interest.

Results

Figure 19 shows a single vehicle trajectory provided by TRAVIS, and Figure 20 shows the estimated vehicle trajectory using Kalman filtering. Because the pixel coordinates of the vehicles are integers, the vehicle trajectory fluctuates and the power of the Kalman filter is weakened. However, we can still see that several segments of the vehicle trajectory are smoothed using the Kalman filter. Overall, however, the Kalman filter has only a very minor effect on this particular vehicle trajectory, and on several vehicle trajectories that were processed.

Figure 21 shows the actual roadway boundaries and estimated boundaries of the road mask for the set of test images. 64.7% of $y_{\min}$ values were underestimated and 67.6% of $y_{\max}$ values were overestimated; that is, the estimated boundaries of the road mask exceed the actual roadway boundaries in about two-thirds of the cycles. Table 1 shows the relative error of estimated roadway boundaries. We can see the average and maximum relative error of estimated roadway boundaries are less than 2.5% and 7%, respectively. Hence, the accuracy of the estimated roadway boundaries is reasonable.

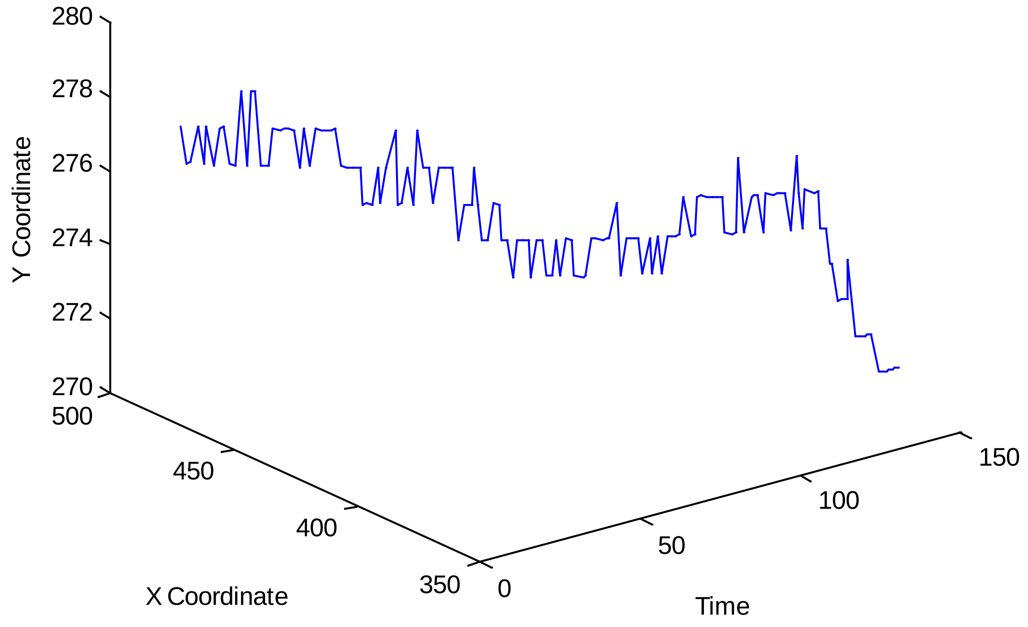


Figure 19: A Single Vehicle Trajectory provided by TRAVIS

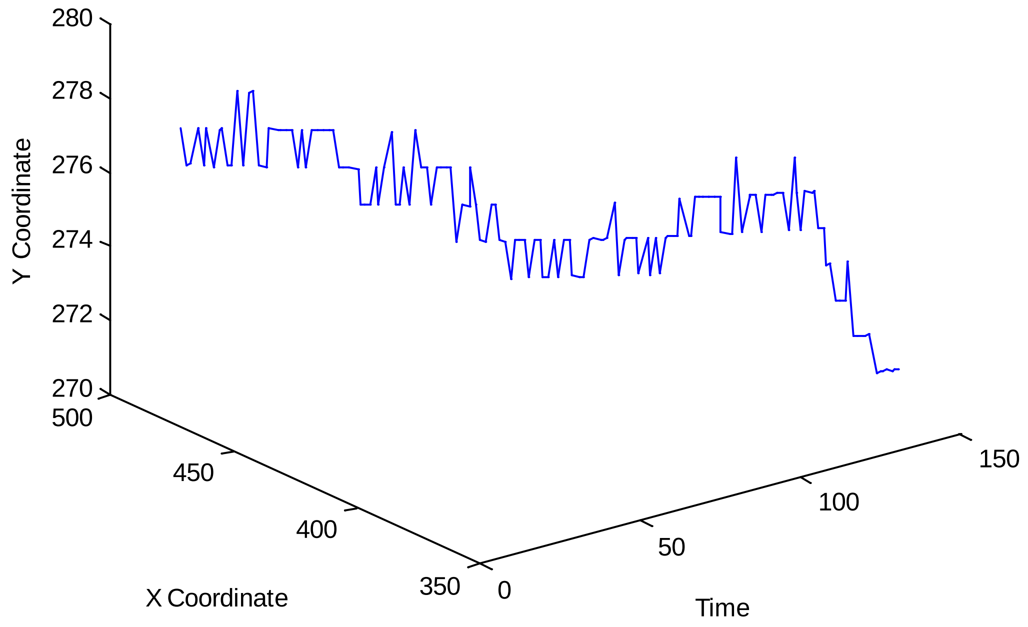


Figure 20: The Estimated Vehicle Trajectory using Kalman Filtering

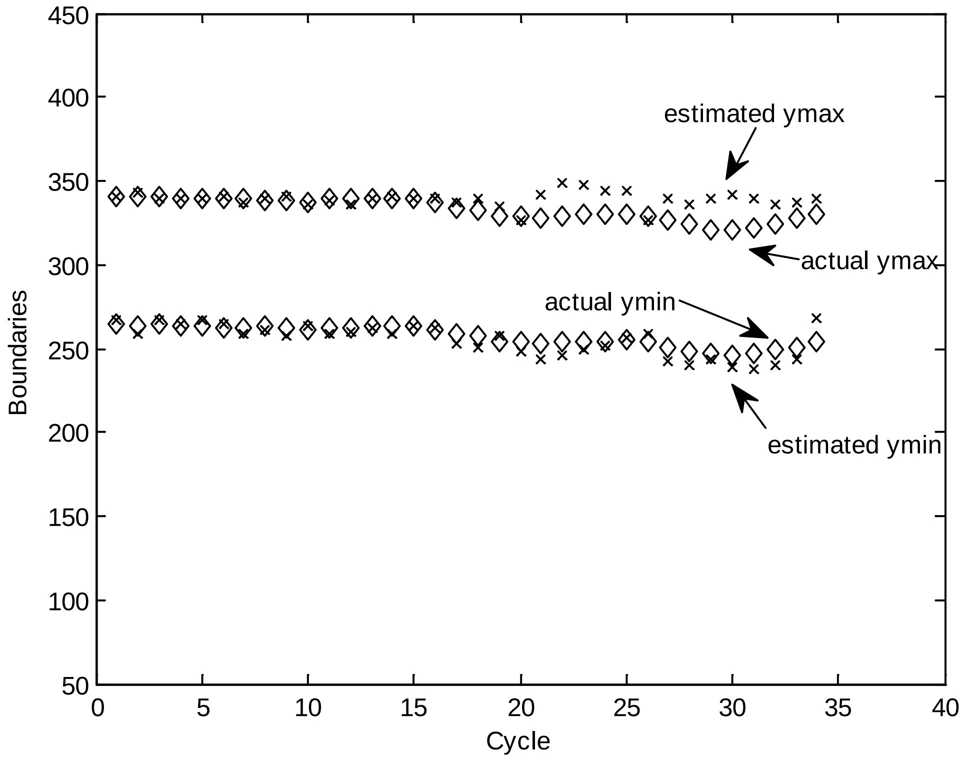


Figure 21: Actual Roadway Boundaries and Estimated Boundaries of the Road Mask

Table 1: Relative Error of Estimated Roadway Boundary

Error	$y_{\min}$	$y_{\max}$
Min	0	0
Max	5.5%	6.5%
Avg	1.8%	2.1%

Figures 22 and 23 show the detected and tracked vehicles before and after applying the road mask, respectively. Using the road mask, the outliers (those objects detected beyond the roadway) are removed, and the Type II error is significantly reduced. The Type I error is roughly 40%. Figures 24 and 25 show the CDTR of RMVDT and VDT in the last frame of each cycle, respectively. From these figures, we can see the percentage of falsely detected and tracked vehicles is 0% in over 60% of the frames of interest when using the road mask while the similar false detection and tracking rate without the road mask is higher than 30% in 55% of the frames of interest. Table 2 shows the comparison for the CDTR of RMVDT and VDT. We can see the improvement of the CDTR is significant. In addition, the computation time for vehicle detection and tracking is reduced by 44%.

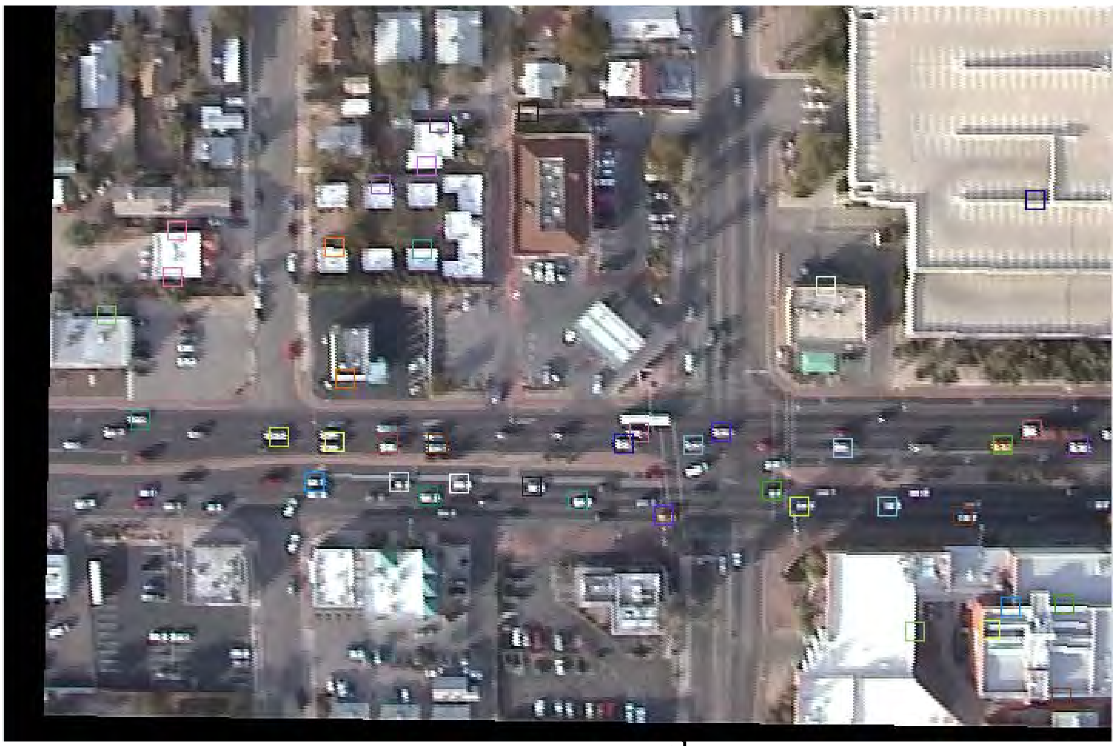


Figure 22: Detected and Tracked Vehicles in 22nd Frame (Without Road Mask)

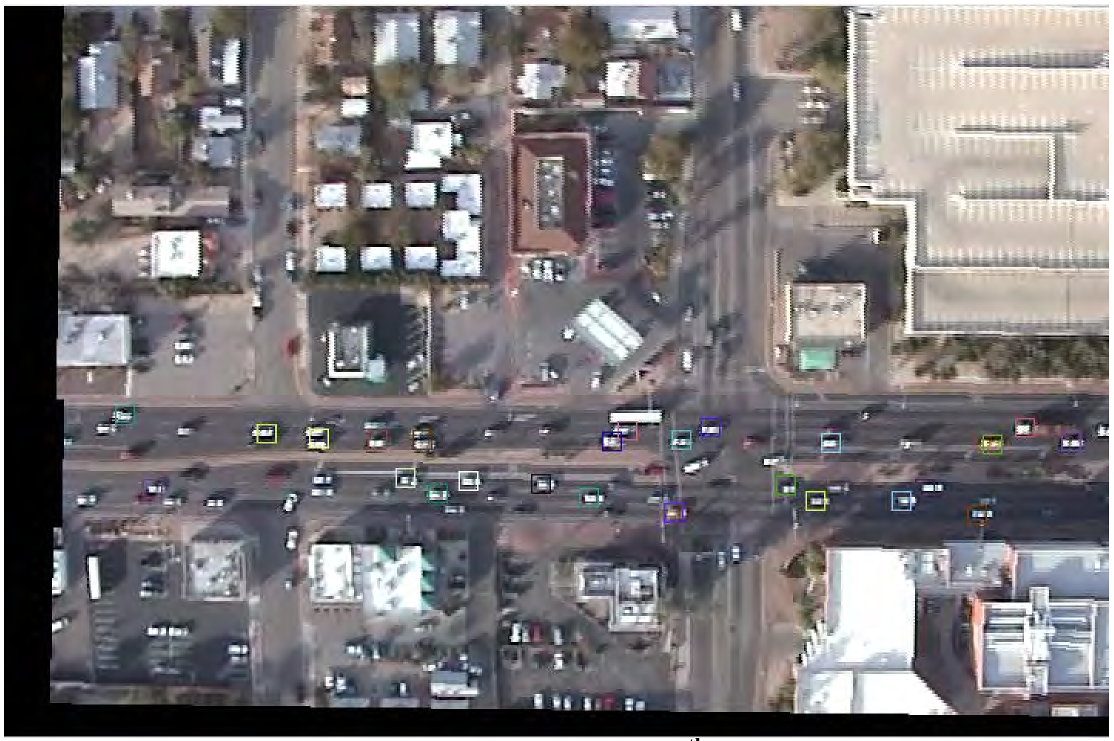


Figure 23: Detected and Tracked Vehicles in 30th Frame (With Road Mask)

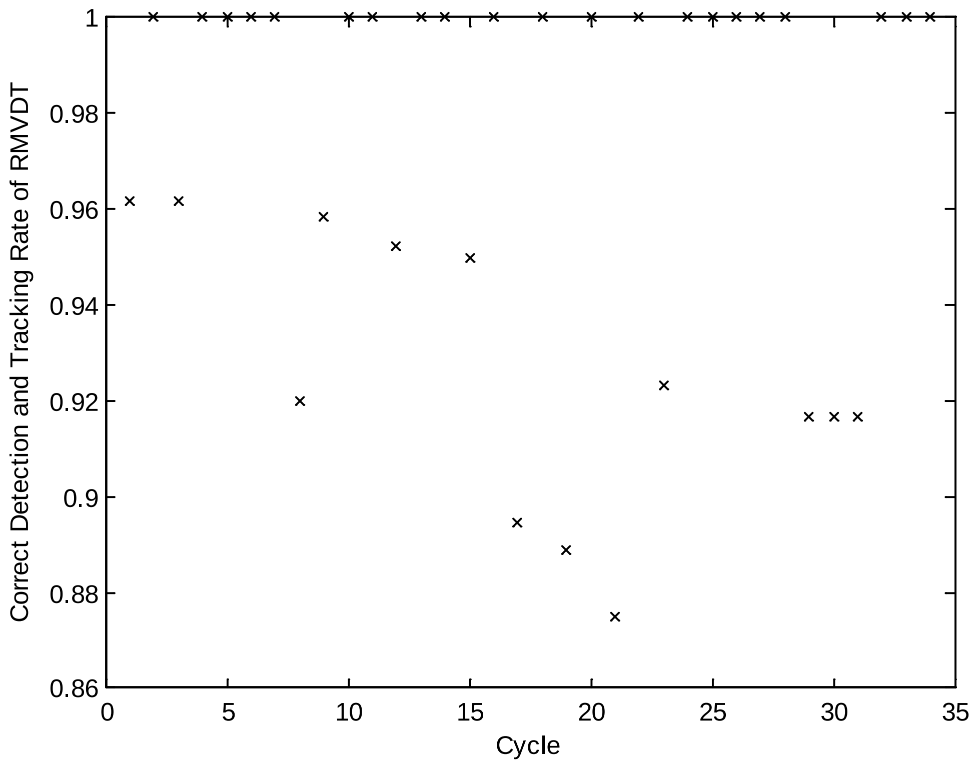


Figure 24: CDTR of RMVDT in the Last Frame of Each Cycle

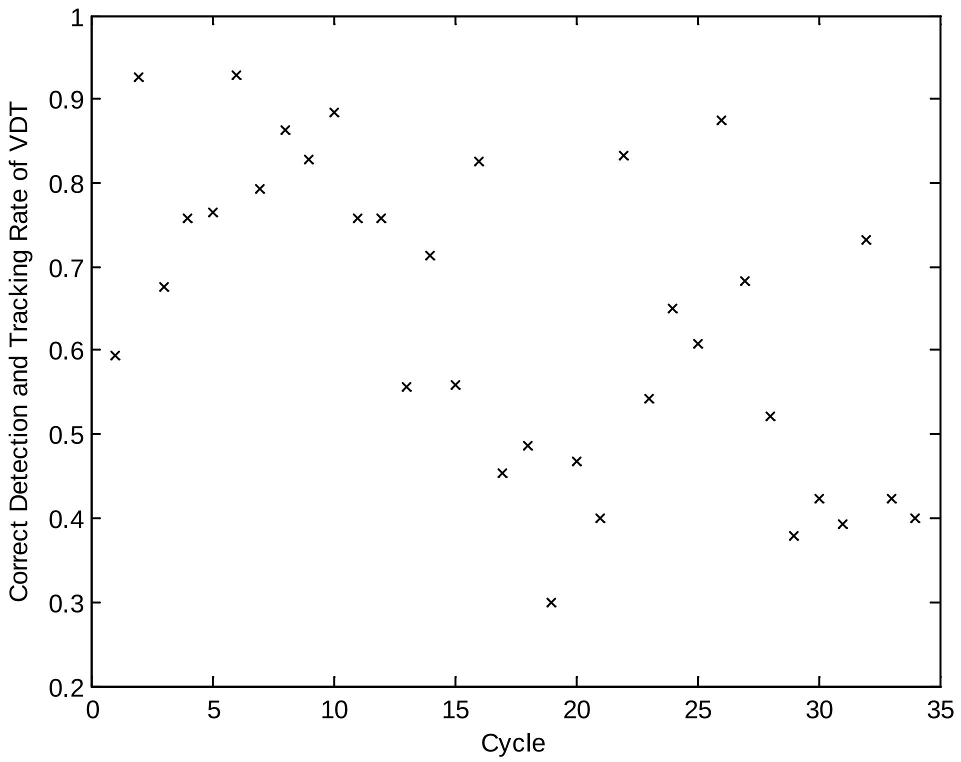


Figure 25: CDTR of VDT in the Last Frame of Each Cycle

Table 2: Comparison for Correct Detection and Tracking Rate

CDTR	RMVDT	VDT
Min	87.5%	30%
Max	100%	92.9%
Avg	97.2%	64%

Vehicle Identification

Our work has focused on airborne traffic sensors using optical platforms (cameras) to record imagery from still images to full-motion video (30 frames per second). The advantage of such platforms is that they are mobile, allowing the platform to move adaptively in time and space to record traffic flow under any desired conditions. However, these platforms can generate a large amount of data in a short amount of time. Manual processing of the imagery is possible, but extracting more precise information (e.g., vehicle counts, vehicle classification, vehicle speeds, etc.) may require some automated image processing.

In this context, we are exploring the use of airborne camera platforms to record individual vehicle trajectories. Data from such trajectories can be used to explore individual driver behavior, so-called microscopic traffic models, and as a means to calibrate traffic simulation models. But, generating such data automatically has significant challenges. Paramount among the image processing challenges are: (1) identifying individual vehicles, and (2) tracking individual vehicles over an extended sequence of images. For the traffic applications we are discussing, these challenges involve a nearly exhaustive enumeration and tracking of vehicles from such imagery. This is because we need *all* vehicle trajectories to examine the behavior of *individual* vehicles in response to *all other* vehicles traveling along a given roadway.

The work described here deals with ways of improving the number and percentage of vehicles that are correctly identified and tracked in airborne imagery. The following section describes other research in this area through a literature review. The third section provides the methodology to improve vehicle detection and tracking, and an illustrative example is shown in the fourth section. The fifth section provides conclusions on this research.

Literature Review

Many algorithms have been developed to detect vehicles automatically in aerial images. Most of them use either explicit [32-35] or implicit models [36,37] of the vehicles. For example, Hinz [32] uses a model-based approach for automatic vehicle detection in high resolution aerial images. The primary geometric and radiometric features of cars including their shadows are described using a 3D-wireframe representation. The model is matched “top-down” to the image, and the support found in the image is evaluated. This approach does not rely on digital maps or site models, and it is not constrained to highway scenes. However, when the road is in the shadow of adjacent buildings, failures will occur.

In contrast, Moon et al. [33] analyze a simple model-based vehicle detection algorithm. An operator for vehicle detection is constructed by combining four elongated edge operators, which collect edge responses from the sides of a vehicle. The proposed method relies on a site model, i.e., the positions and structures of buildings and roads, a context model (i.e., areas where some activity is more possible within the image), and the vehicle model.

In addition, Kim and Malik [34] introduce a model-based 3-D vehicle detection and description algorithm based on a probabilistic line feature grouping, and apply it for vehicle tracking. It is fast and flexible. However, while the vehicle detection is reasonable, the ability of the algorithm to track vehicles is not satisfying, because problems with occlusions, sharp reflections, and changes in image perspective were not addressed.

Zhao and Nevatia [35] describe a car detection system for aerial images by formulating the car detection as a 3D object recognition problem. Important features for human detection of cars are first found by psychological tests. Then the boundary of the car body, the boundary of the front windshield, and the shadow are selected as the primary features. Finally, all features are integrated by a Bayesian network model. This method works well on tested examples. However, dark-colored cars have a higher misdetection rate and a higher false alarm rate than cars of other colors. In addition, other objects with rectangular shape, the foliage of trees, and road markings generate false alarms.

Nguyen et al. [36] propose using the AdaBoost (Adaptive Boosting) algorithm for vehicle detection in large-scale aerial images. An integral image is used for efficient representation and computation of car features. In addition, different types of features, i.e., Haar-like Viola-Jones, an orientation histogram, and the local binary pattern are used to generate hypotheses for training the detector. This framework does not rely on *a priori* knowledge of the image. However, the interactive training takes about four hours on a standard desktop computer for images of 4000 × 4000 pixel size.

Tuermer et al. [37] detect vehicles automatically in aerial image sequences using a pre-processing algorithm to limit the search space for the detector, and using the HoG (Histogram of Oriented Gradients) features, to generate a reliable vehicle detector. In addition, the Real AdaBoost algorithm is used to select features and their combination. The precision rate, i.e., (true positives) divided by (true positives + false positives), is high. However, the recall rate, i.e., (true positives) divided by (true positives + false negatives), is only 80%.

Reinartz et al. [38] present a technique of automatic vehicle detection in serial images collected by airborne cameras. A difference image is generated between the mapped images and the median image. A modified Laplace operator is applied to detect the borders of the objects. Using this method on a set of imagery, the detected car fraction is 80%, and more than 20% false alarms are generated.

Hinz et al. [39] develop a system to detect and track vehicles in low-frame-rate aerial image sequences. In the vehicle detection process, a channel differencing approach is first used to extract vehicles with significant color features, and then dynamic thresholding is applied in a constrained manner only to blob-like features. The extraction process is fast. However, the shadow of a vehicle is falsely detected as a vehicle in the example.

Nejadasl et al. [40] incorporate a scaled pixel-space approach into an optical flow method for vehicle tracking. The individual car pixels and the complete car region are first tracked separately using the gradient-based optical flow method, which is initiated by the scaled pixel-space approach. A statistical decision making method, based on a rigid-object assumption, is then performed for the best result. It is assumed that the displacement vector with the largest number of occurrences represents the real car displacement. The experimental results are promising. However, some dark cars on a dark background are lost.

As a general summary, existing methods for vehicle detection and tracking suffer from a variety of drawbacks. Most notably, they usually require a vehicle model for determining vehicle-shaped objects in the image. Shadows, occlusions, and dark vehicles (or vehicles of a similar color of the background pavement) represent particular problems in vehicle tracking. Finally, correct detection of vehicles is important, but there is a fundamental tradeoff with false alarm rates.

Methodology

Figure 26 shows a flow chart explaining the vehicle detection and tracking process, based on the previous work by other researchers at the University of Arizona [26,41].

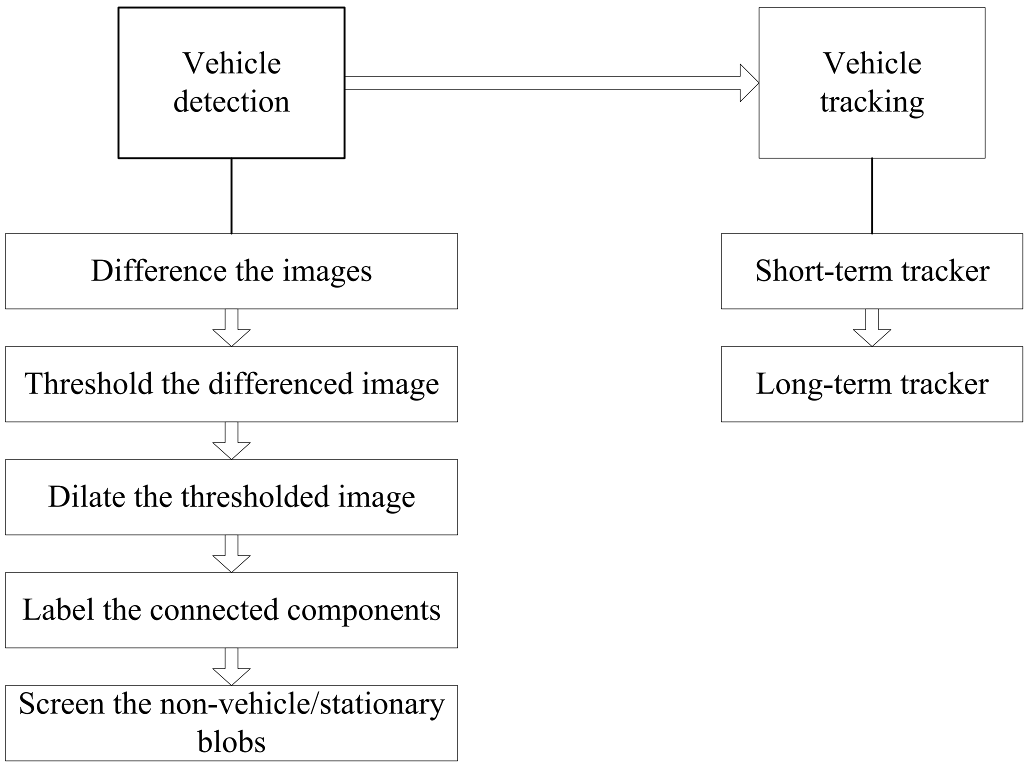

Vehicle detection
Vehicle tracking
Difference the images
Threshold the differenced image
Dilate the thresholded image
Label the connected components
Screen the non-vehicle/stationary blobs
Short-term tracker
Long-term tracker

FIGURE 26 Vehicle detection and tracking process in previous work [26,41].

The procedure is as follows:

Step 1: Detect the blobs representing vehicles in the registered image.

Step 1.1: Take the difference between the current image *i* with the image *i-gap* (*gap* is a calibrated integer).

$$diff(i,j)=max(abs(fy\_r(i,j)-ref\_r(i,j)),abs(fy\_g(i,j)-ref\_g(i,j)),abs(fy\_b(i,j)-ref\_b(i,j)))$$

$$diff\_img(i,j)=\begin{cases}0 & if\ \ diff(i,j)<thresh\_diff \\ diff(i,j) & o.w.\end{cases} \tag{8}$$

where

$fy\_r(i,j)$, $fy\_g(i,j)$ and $fy\_b(i,j)$ represent the red, green and blue intensities in the current image *i*, respectively.

$ref\_r(i,j)$, $ref\_g(i,j)$ and $ref\_b(i,j)$ represent the red, green and blue intensities in the image *i-gap*, respectively.

$diff(i,j)$ represents the difference of the two images in pixel (i,j).

$diff\_img(i,j)$ represents the intensity in pixel (i,j) in the differenced image.

$thresh\_diff$ represents the calibrated threshold for the differenced image.

Step 1.2: Threshold the differenced image using equation (9):

$$thresh(i,j)=\begin{cases}255 & thresh1>=threshold\_value \text{ OR } thresh2>=threshold\_value \\ 0 & o.w.\end{cases} \tag{9}$$

where

thresh1/*thresh2* represents a local average in a horizontal/vertical window centered at pixel (i,j) in the differenced image.

threshold_value represents a calibrated threshold.

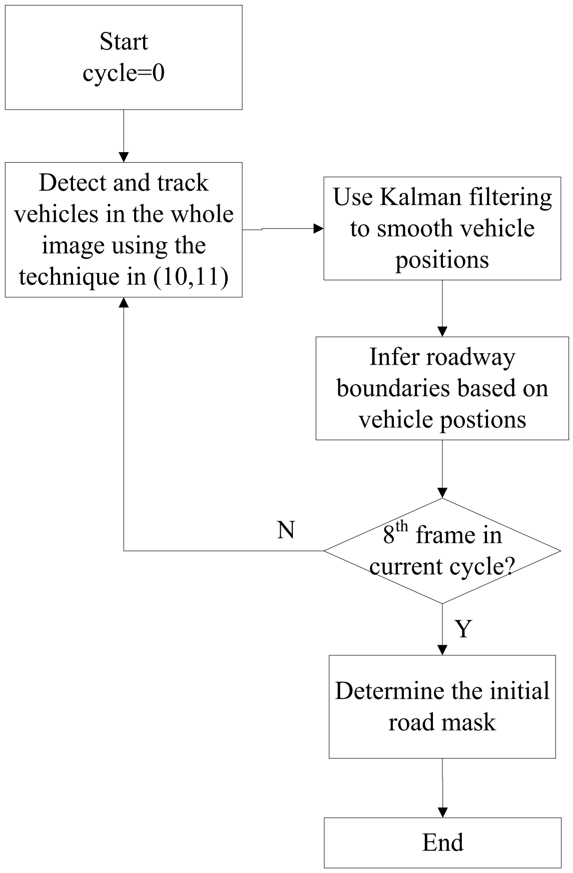
Step 1.3: Dilate the thresholded image.

Step 1.4: Do connected component labeling to give a unique label for each blob.

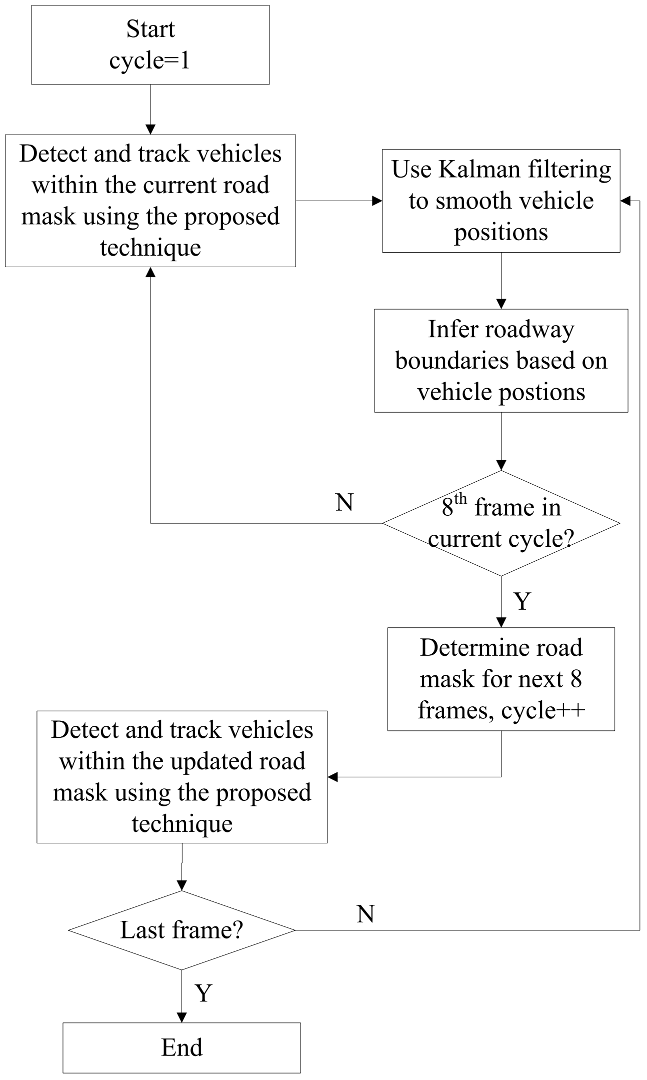
Step 1.5: Screen out the non-vehicle and stationary blobs based on the blob size and the movement of the blobs in x and y directions in 14 frames. The number of 14 frames matches well with a relatively low airborne platform speeds of 20-30 mph over an arterial and the frame rate of 30 frames/second.

Step 2: Track the vehicle blobs using a short-term and long-term trackers. The short-term tracker is used to detect new moving vehicles and screen out non-vehicle blobs. The long-term tracker is used to maintain the vehicle coordinates over time and track vehicles even when they are not detected for short periods of time.

For more details on each of these steps, please refer to [26,41]. The proposed framework involves two steps as shown in Figure 27: first, estimate the initial road mask based on the vehicle detection and tracking technique in [26,41] using the first n frames; and, second, detect and track vehicles within the road mask using the proposed technique starting from $(n+1)^{th}$ frame, and update the road mask every cycle, i.e., 8 frames. More details on estimating the road mask are given in [42]. The reference frame for registration is the initial frame of the image sequence, and the vehicles detected and tracked in step 1 are not inherited in step 2. More details on the proposed technique are given in the following sections.


Start
cycle=0
Detect and track vehicles in the whole image using the technique in (10,11)
Use Kalman filtering to smooth vehicle positions
Infer roadway boundaries based on vehicle postions
8th frame in current cycle?
N
Y
Determine the initial road mask
End

(a)


Start
cycle=1
Detect and track vehicles within the current road mask using the proposed technique
Use Kalman filtering to smooth vehicle positions
Infer roadway boundaries based on vehicle postions
N
8th frame in current cycle?
Y
Determine road mask for next 8 frames, cycle++
Detect and track vehicles within the updated road mask using the proposed technique
Last frame?
N
Y
End

(b)

FIGURE 27 Proposed vehicle detection and tracking process: (a) estimate the initial road mask based on the vehicle detection and tracking technique in [26,41] and (b) detect and track vehicles within the road mask using the proposed technique.

Difference the Images

A lower threshold, *thresh_diff* (=15), is used to determine whether a pixel is a vehicle pixel or not in the differenced image, compared to *thresh_diff* (=25) in the previous work. In this way, the vehicle pixels with low contrast to the roadway may still remain in the differenced image.

Threshold the Differenced Image

We assume that there are only red/blue vehicles shown as red/blue on the roadway in the image. Take the red vehicles for example. Then, we seek to determine whether it is a red pixel or not based on the intensity difference between the red and the green/blue, using equation (10):

$$thresh(i,j)=\begin{cases} 255 & fy\_r(i,j)>fy\_g(i,j)+thresh\_r \text{ OR } fy\_r(i,j)>fy\_b(i,j)+thresh\_r \\ 0 & o.w. \end{cases} \tag{10}$$

where

$thresh(i,j)$ represents the intensity in pixel (i,j) in the initial thresholded image.

$fy\_r(i,j)$, $fy\_g(i,j)$ and $fy\_b(i,j)$ represent the red, green and blue intensities in the current original image, respectively.

$thresh\_r$ represents a calibrated threshold.

If $diff\_img(i,j) \geq thresh\_normal$ ($thresh\_normal$ =25), we detect the potential vehicle pixels based on the pixel intensity in the differenced image and the current original image, using equation (11):

$$thresh(i,j)=\begin{cases} 255 & thresh1>=thresh\_normal \text{ AND } thresh3>threshold\_origavg \\ & \text{AND } (fy\_r(i,j)+fy\_g(i,j)+fy\_b(i,j))/3>threshold\_orig \\ 0 & o.w. \end{cases} \tag{11}$$

where

$thresh(i,j)$ represents the intensity in pixel (i,j) in the initial thresholded image.

$thresh1$ represents a local average in a horizontal window centering at pixel (i,j) in the differenced image.

$thresh3$ represents a local average in a horizontal window centering at pixel (i,j) in the current original image.

$fy\_r(i,j)$, $fy\_g(i,j)$ and $fy\_b(i,j)$ represent the red, green and blue intensities in the current original image, respectively.

$threshold\_origavg$ and $threshold\_orig$ represent the calibrated thresholds.

This previous step in (11) assists by detecting the vehicles in the image, using a typical difference threshold of 25 units in pixel intensity. Noises (for example, shadows) as well as vehicles with lower contrast with the roadway are eliminated in the initial thresholded image. To better detect vehicles with lower contrast with the roadway, we use a lower threshold value. So, if $diff\_img(i,j) \geq thresh\_low$ ($thresh\_low$=15), find the vehicle pixels with low contrast to the roadway using equation (12):

$$thresh\_temp(i,j)=\begin{cases} 255 & thresh1>=thresh\_low \text{ AND } abs(diff\_img(i,j)-thresh3)<threshold\_l \\ 0 & o.w. \end{cases} \tag{12}$$

where

thresh_temp(i,j) represents the intensity in pixel (i,j) in the temporary thresholded image.

thresh1 represents a local average in a horizontal window centering at pixel (i,j) in the differenced image.

diff_img(i,j) represents the intensity in pixel (i,j) in the differenced image.

thresh3 represents a local average in a horizontal window centering at pixel (i,j) in the current original image.

threshold_l represents a calibrated threshold.

Because a lower threshold, *thresh_low*, is used and the intensity of the black vehicle is relatively low, the black vehicle pixels remain.

We assume that vehicles run approximately in the middle of the lane, and the lane width is twice the vehicle width. Hence, the minimal distance in the lateral direction between any two vehicles running in parallel is approximately the width of a vehicle. In addition, we assume that the minimal car-following distance (for moving vehicles) is the length of the vehicle. Using these assumptions, we then screen out the non-vehicle pixels, for example, the shadows of the vehicles, based on the relative position of the vehicles on the roadway using equation (13):

$$thresh\_tempf(i,j)=\begin{cases}255 \quad thresh\_temp(i,j)=255 \text{ AND } thresh(i,j)=0 \text{ AND } thresh(i+a,j+b)\neq 255 \\ (a\in[-car\_width/2-2,car\_width/2+2], b\in[-car\_length/2-3,car\_length/2+3]) \\ 0 \qquad o.w.\end{cases} \tag{13}$$

where

thresh_tempf(i,j) represents the intensity in pixel (i,j) after filtering in the temporary thresholded image.

thresh_temp(i,j) represents the intensity in pixel (i,j) in the temporary thresholded image.

thresh(i,j) represents the intensity in pixel (i,j) in the initial thresholded image.

car_width and *car_length* represent the estimated width and length of the vehicle in the image.

Then, we obtain the final thresholded image by making a union of the initial thresholded image obtained from equations (10) and (11) and the temporary thresholded image generated by equation (12), using equation (14):

$$thresh\_final(i,j)=\begin{cases}255 \qquad thresh(i,j)=255 \text{ OR } thresh\_tempf(i,j)=255 \\ 0 \qquad o.w.\end{cases} \tag{14}$$

where

thresh_final(i,j) represents the intensity in pixel (i,j) in the final thresholded image.

thresh(i,j) represents the intensity in pixel (i,j) in the initial thresholded image obtained from equations (10) and (11).

thresh_tempf(i,j) represents the intensity in pixel (i,j) after filtering in the temporary thresholded image.

Dilate the Thresholded Image

The task of dilation, which was originally intended to create more contiguous blobs, was problematic in the previous work, because some of the vehicles very near each other were merged into a single blob. For this reason, we removed the step of dilating blobs in the image.

Connected Component Labeling

Because of different colors, shading, and textures on different parts of the vehicles, vehicles may be split into two or more blobs when differencing the images. After the connected component labeling, two blobs belonging to one vehicle should be merged. Because of different moving patterns between vehicles running in a single direction and turning vehicles, we perform the following steps:

If $abs(centroid\_x(i)\text{-}centroid\_x(j))<car\_length$ and $abs(centroid\_y(i)\text{-}centroid\_y(j))<thresh\_s$, where $thresh\_s$ is a calibrated threshold, merge the two blobs representing one vehicle running in a single direction using equation (15):

$$
\begin{aligned}
& centroid\_x(i)=\begin{cases}(centroid\_x(i)+centroid\_x(j)\text{-}gap)/2 & centroid\_y(i)<(y_{min}+y_{max})/2\\(centroid\_x(i)+centroid\_x(j)+gap)/2 & centroid\_y(i)>(y_{min}+y_{max})/2\end{cases}\\
& centroid\_y(i)=(centroid\_y(i)+centroid\_y(j))/2\\
& centroid\_x(j)=0\\
& centroid\_y(j)=0
\end{aligned} \tag{15}
$$

where

$centroid\_x(i)$, $centroid\_y(i)$, $centroid\_x(j)$ and $centroid\_y(j)$ represent the *x* coordinate and *y* coordinate of the centroid of the blob *i* and blob *j*, respectively.

gap represents the image number difference between two differenced images. In our test image, normal vehicles travel one pixel between two consecutive frames. Hence, equation (15) gives a reasonable estimate of the x coordinate of the merged blob.

y_{min} and y_{max} represent the estimated roadway boundaries.

A separate method is used for vehicles that are turning along the roadway. Specifically, if $abs(centroid\_x(i)\text{-}centroid\_x(j))<car\_length/2$ and $abs(centroid\_y(i)\text{-}centroid\_y(j)) \in [thresh\_tl, thresh\_th]$, where $thresh\_tl$ and $thresh\_th$ are calibrated thresholds, merge the two blobs representing one turning vehicle using equation (16):

$$\begin{aligned} &centroid\_x(i)=(centroid\_x(i)+centroid\_x(j))/2 \\ &centroid\_y(i)=(centroid\_y(i)+centroid\_y(j))/2 \\ &centroid\_x(j)=0 \\ &centroid\_y(j)=0 \end{aligned} \tag{16}$$

where $centroid\_x(i)$, $centroid\_y(i)$, $centroid\_x(j)$ and $centroid\_y(j)$ represent the x coordinate and y coordinate of the centroid of the blob i and blob j, respectively.

Compared to the minimal blob size (=50) in the previous work [26,41], a much smaller minimal blob size(=5) is used to screen out the non-vehicle blobs, considering there are some vehicles that appear as small white blobs in the image. Conversely, if the blob is even longer than a truck, it is not considered to be a vehicle. So, we can remove this non-vehicle blob if inequality (17) is satisfied:

$$col_{max}(i)\text{-}col_{min}(i)>truck\_length \tag{17}$$

where

$col_{max}(i)$ and $col_{min}(i)$ represent the maximal and the minimal column number of blob i.

$truck\_length$ represents the estimated length of the truck in the image.

If the blob does not move considerably, it is considered as a stationary object. To do this, we have a threshold of movement in the long-term tracker that ensures that the blob moves sufficient pixels in order to be considered. So, we screen out the stationary object if inequality (18) is satisfied:

$$\begin{aligned} &abs(fh\text{->}feature[ii\text{-}1]\text{->}x - fh\text{->}feature[ii\text{-}15]\text{->}x)+ abs(fh\text{->}feature[ii\text{-}1]\text{->}y - \\ &fh\text{->}feature[ii\text{-}15]\text{->}y)<=longterm\_car\_mov \end{aligned} \tag{18}$$

where

$fh\text{->}feature[ii\text{-}1]\text{->}x$, $fh\text{->}feature[ii\text{-}1]\text{->}y$, $fh\text{->}feature[ii\text{-}15]\text{->}x$ and $fh\text{->}feature[ii\text{-}15]\text{->}y$ represent the x coordinate and y coordinate of the current blob in the (ii)th frame and (ii-14)th frame, respectively.

$longterm\_car\_mov$ is a calibrated threshold, representing how many pixels normal vehicles traveled in 14 frames.

Experiments

Experimental Design

A set of images were collected in January 2011 from a camera mounted on a helicopter, traveling over Speedway Boulevard (an arterial roadway) in Tucson, AZ. A sample of 200 frames was selected for testing using the new vehicle detection and tracking techniques. These images were first converted to Portable Pixel Map (PPM) format and then fed into TRAVIS. For this imagery, the scale of the image is set to 1.34 feet per pixel and is expected to be fairly constant across each individual image, and across the full set of images.

In analyzing the imagery, there are two types of errors we can reduce: one is to reduce the number of existing vehicles in the imagery that are not tracked, which we refer to as a Type I error, and the other is to decrease the number of falsely tracked objects that are not vehicles, which is referred to a Type II error. The recall rate (*RR*) is related to a Type I error: reducing the number (or percentage) of vehicles that are not tracked. The precision rate (*PR*) is related to a Type II error: reducing the number (or percentage) of tracked objects that are not vehicles.

$$RR = \frac{\#TP}{\#TP + \#FN} \tag{19}$$

$$PR = \frac{\#TP}{\#TP + \#FP} \tag{20}$$

where *TP* are the true positives, *FP* are the false positives, *FN* are the false negatives. Note that *TP* is the number of tracked objects minus the false positives (*FP*). Also, (*TP*+*FN*) and *FP* are obtained by manually counting the vehicles in the images.

Results

Figures 28 and 29 illustrate an example of the tracked vehicles for the set of Speedway images using the enhanced algorithm and the previous algorithm [26,41], respectively. There is a color box drawn around the tracked objects. Compared to Figure 29, we can see that vehicles running in parallel, vehicles shown as small white blobs, red vehicles, and black vehicles are successfully tracked in the new algorithm, shown in Figure 28.

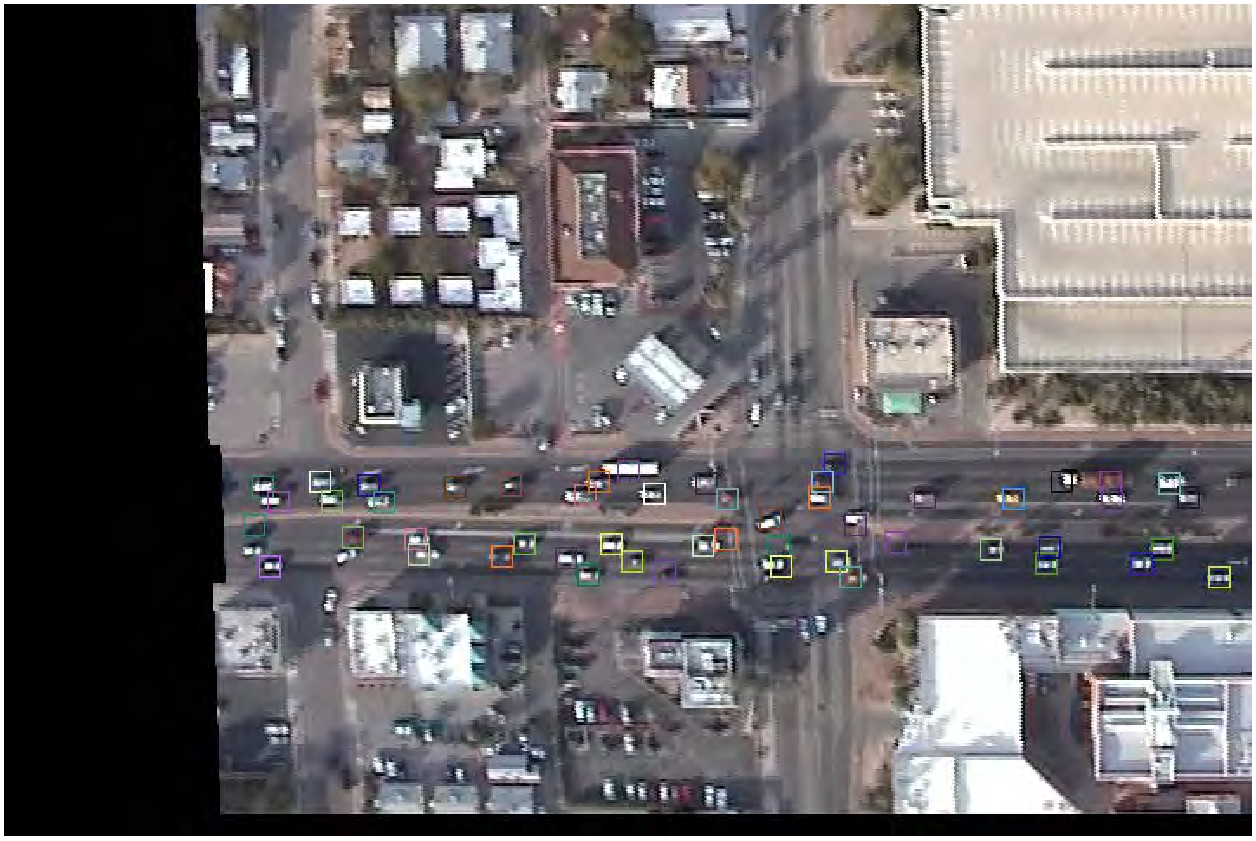


FIGURE 28 Tracked vehicles in 91st frame of Speedway images (Enhanced Algorithm)

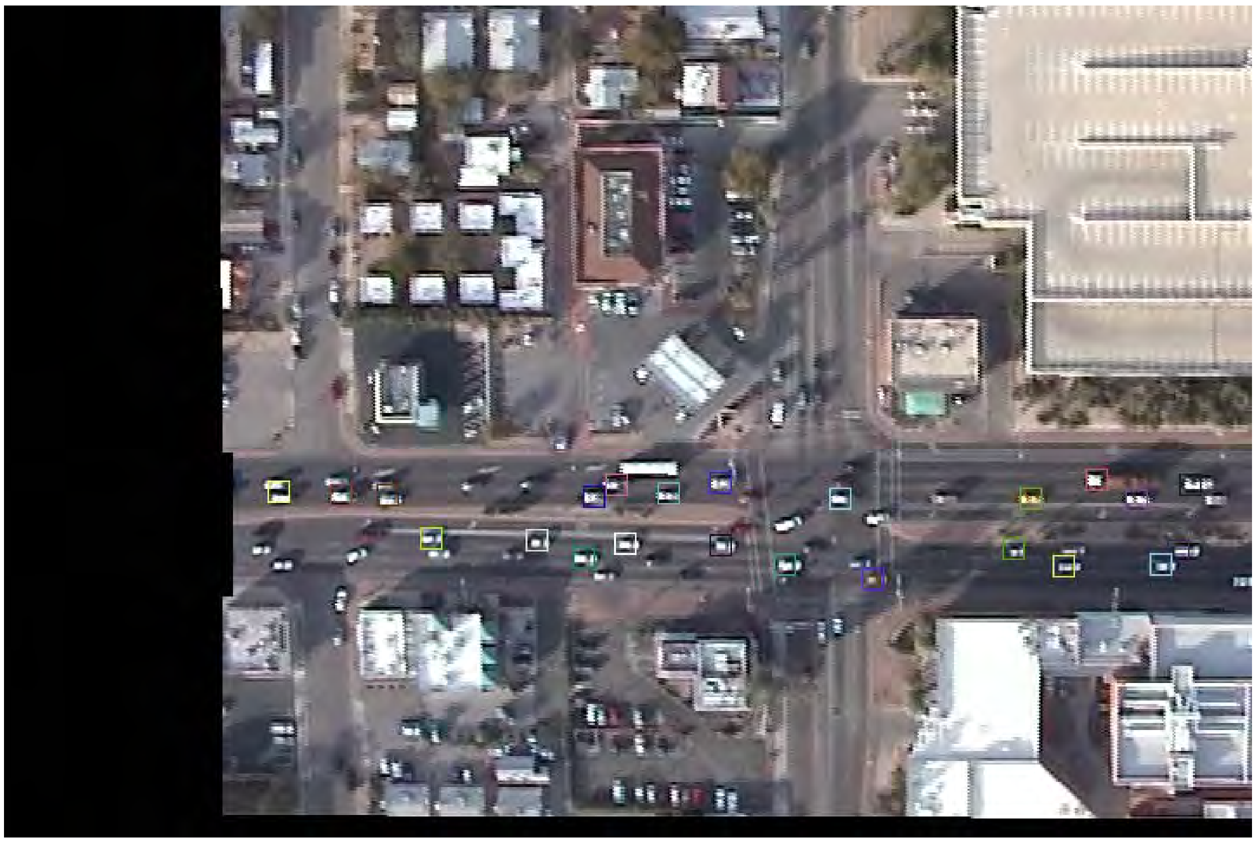


FIGURE 29 Tracked vehicles in 91st frame of Speedway images (previous work [26,41])

Figure 30 shows the *RR* in the last frame of each cycle for the set of images from Speedway Boulevard. We can see that the *RR* fluctuates, as determined by the *longterm_car_mov* checking object tracking every 16 frames. The number of 16 frames in the examples here is chosen to be consistent with previous work [26], and matches well with a relatively low airborne platform speeds of 20-30 mph over an arterial and the frame rate of 30 frames/second. Firstly, there are three vehicles waiting to turn at the intersection. They are tracked firstly and then dropped after the *longterm_car_mov* is checked repeatedly, because these vehicles do not move sufficiently to pass this criterion. Secondly, there are some vehicles turning onto Speedway eastbound from the parking lot nearby. This slows down the vehicles on Speedway and makes those vehicles fail under the same criterion with *longterm_car_mov*.

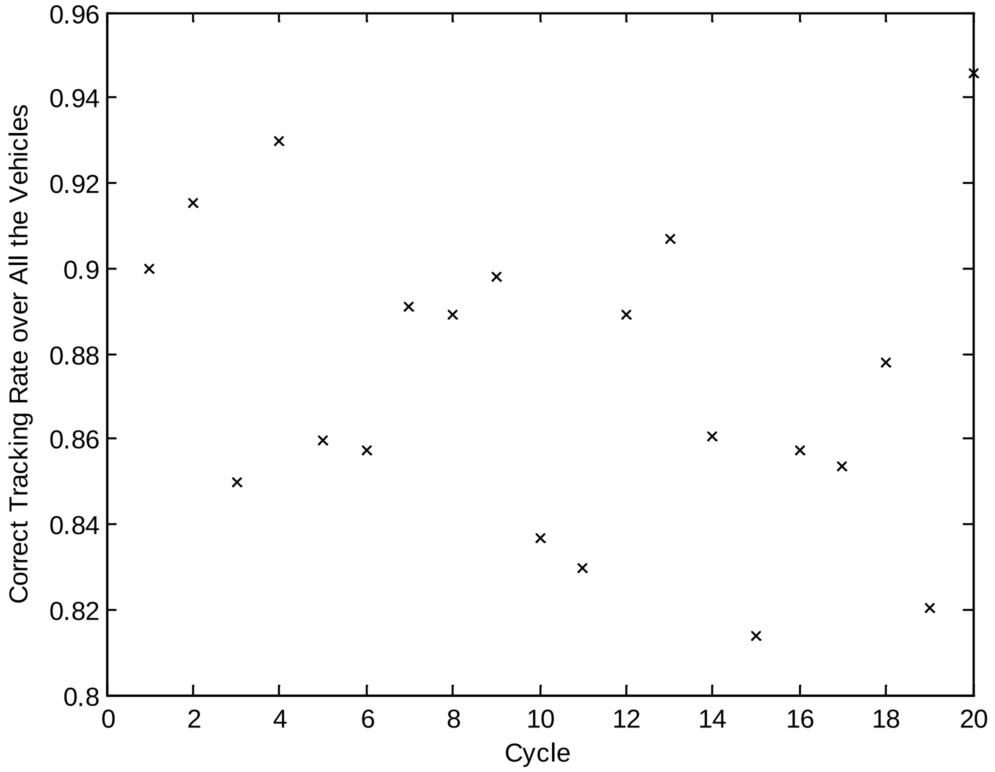


FIGURE 30 *RR* in the last frame of each cycle (Speedway images).

Figure 31 shows the *PR* in the last frame of each cycle for the set of images from Speedway Boulevard. We can see that the *PR* decreases overall. This is because the processed area in each image decreases, while the image number increases in this version of TRAVIS. In the image registration step, a single initial image is used as a reference image. All subsequent images are then registered to this initial image. If the overlap between image 1 and the subsequent image k is small, then only vehicles in this overlapping area are tracked. As the field of view changes (e.g., as the helicopter moves toward other areas along the roadway), the area of overlap with image 1 will be reduced, and fewer and fewer vehicles are tracked.

We can also see that the *PR* fluctuates a little bit. This is mainly because some non-vehicle objects are not persistently tracked because they do not move in a pattern consistent with other vehicles. We can exclude these short trajectories if their column is not close to either the column edge or the width of each image. In addition, there is one truck with different colors in the front and the back which is persistently tracked as two separate vehicles. Because of the car-following pattern of the vehicles on the arterial, it is difficult to differentiate the long truck from two cars following each other.

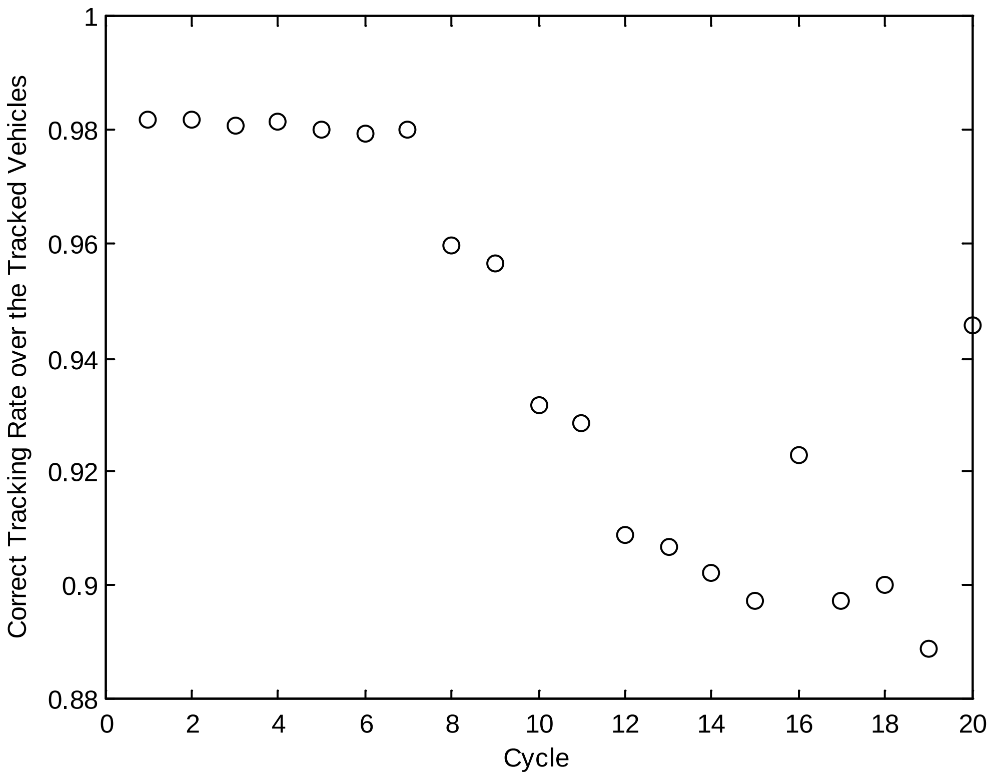


FIGURE 31 *PR* in the last frame of each cycle (Speedway images).

Table 3 shows the Type I and Type II errors of tracked vehicles (Speedway Images). We can see that the average Type I error and Type II error is 12.6% and 5.9%, respectively. As mentioned previously, short trajectories can be excluded if their columns are not close to either the column edge or the width of each image. Hence, the performance of the vehicle detection and tracking algorithm appears to be reasonable.

TABLE 3 Type I and Type II Errors of Tracked Vehicles (Speedway Images)

Error	Type I Error	Type II Error
Min	5.4%	1.8%
Max	18.6%	11.1%
Avg	12.6%	5.9%

Conclusions

Image processing methods provide a useful means of extracting traffic data from video image streams. In applications that require more exhaustive enumeration of vehicles and their positions, there remain challenges to vehicle identification and tracking. We have shown in this research that it is possible to detect and to track a high percentage of vehicles, while managing the number of false detections (false alarms). This method requires a more careful treatment of the imagery, as the intensity of some vehicles in the imagery is very similar to that of the underlying roadway. Care is also needed to recognize sufficiently small vehicle blobs, small vehicle movements, and correlation of connected components to recognize and track vehicles without other false positives. In this paper, specific methods of dealing with these challenges have been presented, and these techniques have been incorporated into our image processing software, TRAVIS.

By improving these image processing techniques, we are now able to generate a sufficient sample of vehicle trajectories to begin estimation and calibration of traffic models with a minimum of manual data processing. At the same time, many others are working in similar research areas, with the hope of designing image processing tools that can respond to similar challenges. We remain optimistic that such tools can significantly improve the quality and quantity of microscopic traffic data that are available to the transportation community.

Dynamic Image Processing

In the image registration step in the old version of TRAVIS, a single initial image is used as a reference image. All subsequent images are then registered to this initial image. If the overlap between image 1 and the subsequent image *k* is small, then only vehicles in this overlapping area are tracked. As the field of view changes (e.g., as the helicopter moves over other areas along the roadway), the area of overlap with image 1 will be reduced, and fewer and fewer vehicles are tracked.

The goal of what we call "dynamic image processing" is to allow the tracking of vehicles in extended image sequences, even when the field of view changes. This allows more continuous tracking of vehicles and processing of their trajectories.

Calculate Dynamic Image Processing Region

The key idea of dynamic image processing is that the region of the image to be processed is moving in the same direction of the helicopter. There is one single reference frame, which is usually the first frame of the image sequences. Assuming the helicopter moves to the right (or left), and the reference frame is the first (or last) frame of the image sequence, the relative positions of the non-reference frames with respect to the reference frame can be categorized in one of six scenarios, as shown in Figure 32. In this figure, the image in the background is the reference frame, and the one in the front is the non-reference frame. In the case of the helicopter moving to the left (or right) and the reference frame being the first (or last) frame of the image sequence, the analysis is similar.

Take scenario 1 as an example, as detailed in Figure 33. Here, the non-reference frame has shifted to the right, as the helicopter has moved along the roadway. In the old version of TRAVIS, the result would be that only the area of overlap of the two frames would be processed. In this case, however, we would like to add the region on the right, to allow vehicle tracking in that direction in the image.

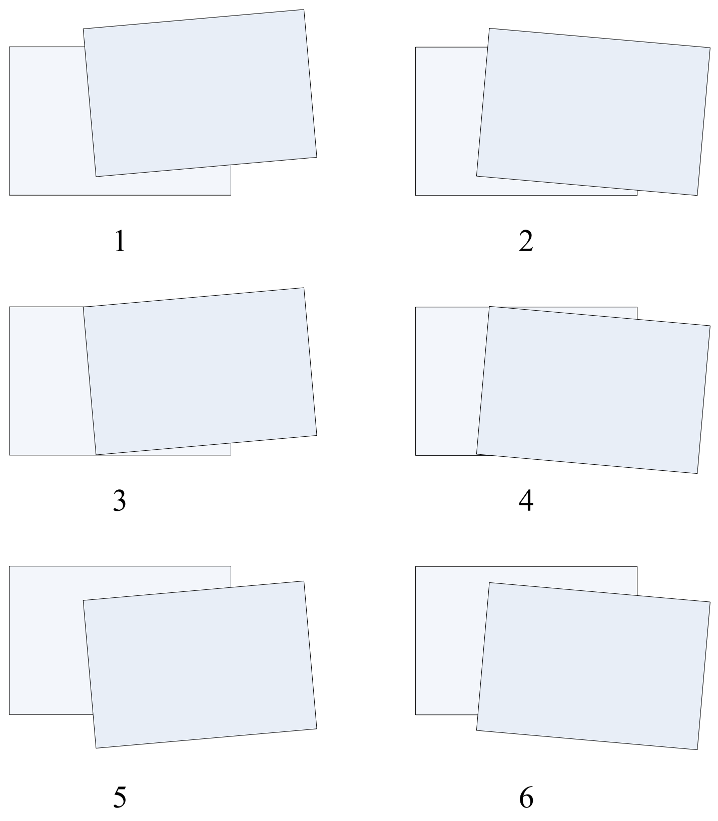


Figure 32: Relative Positions of the Non-reference Frames with respect to the Reference Frame

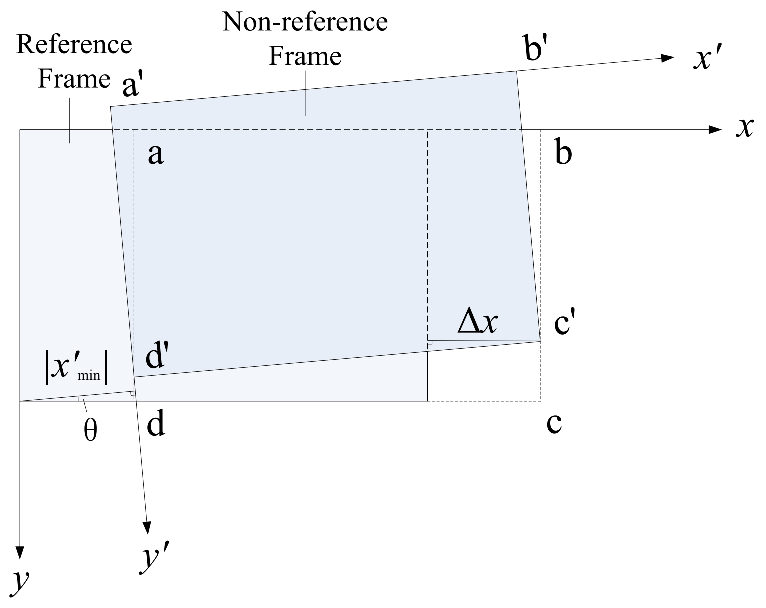


Figure 33: Dynamic Image Processing Region Calculation

In such a case, the dynamic image processing region for the *i*th frame (the non-reference frame) is calculated as follows:

Step 1: Transform each pixel coordinate from the reference frame coordinate system (*x*, *y*) to the non-reference frame coordinate system (*x*', *y*').

Step 2: Calculate the minimal x' coordinate for all pixels in the reference frame, $x'_{\min}$.

Step 3: Approximate Δx using $|x'_{\min}|$, with

$$\Delta x \approx L\cos\theta - L + |x'_{\min}|\cos\theta \qquad (28)$$

L denotes the width of the image. θ is small, hence $\Delta x \approx |x'_{\min}|$.

Step 4: Calculate the processing region for the *i*th frame.

$$x \in [\Delta x, \Delta x\text{+L}] \qquad y \in [y_{\min}, y_{\max}] \qquad (29)$$

$y_{\min}$ and $y_{\max}$ are roadway boundaries, which are calculated in the road mask part.

Obtain Actual Pixel Intensities

The actual pixel intensities for the *i*th frame are obtained as follows:

Step 1: Transform each pixel coordinate in the region of $abcd$, in which $x \in [\Delta x, \Delta x\text{+L}]$ and $y \in [0, nrows]$, from the reference frame coordinate system (*x*, *y*) to the non-reference frame coordinate system (*x*', *y*').

Step 2: Obtain the pixel intensities for the *i*th frame using (30):

$$I(x,y) = \begin{cases} I(x',y') & x' \in [0,L]\; y' \in [0,D] \\ 0 & \text{Otherwise} \end{cases} \qquad (30)$$

L and D denote the width and height of the image, respectively.

For each pixel (x,y) in the region of $abcd$ in the reference coordinate system, if its transformed coordinate (x',y') is in the region of $a'b'c'd'$, the pixel intensity of (x,y) equals to the pixel intensity of (x',y') in the non-reference frame; otherwise the pixel intensity is set to 0.

Differencing Images Dynamically

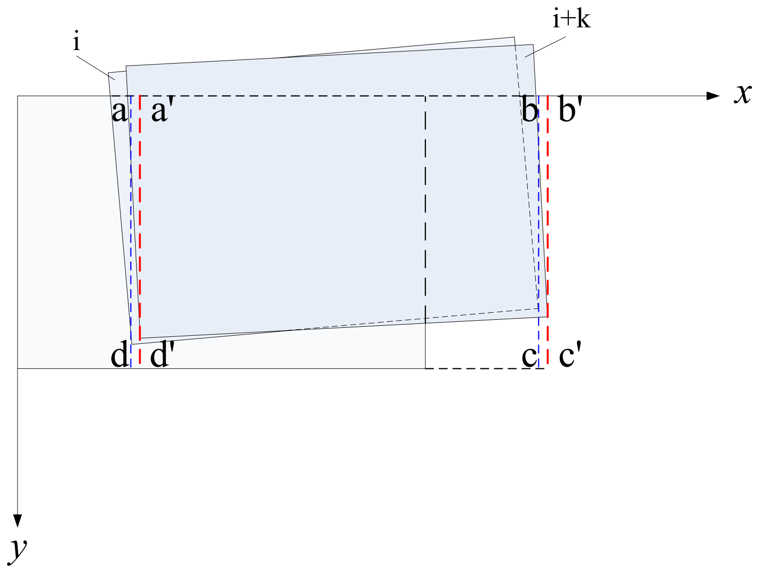


Figure 34: Principles of Differencing Images

The procedure of differencing the (*i*+*k*)th image to the *i*th image, as shown in Figure 34, is as follows:

Step 1: Assign 0 to pixel intensities for the reference frame *i* in the region of $a^{'}b^{'}c^{'}d^{'}$, in which $x \in [Col_{\min}(i+k), Col_{\max}(i+k)]$ and $y \in [0, nrows]$.

$Col_{\min}(i+k)$ and $Col_{\max}(i+k)$ denotes the minimal and maximal *x* coordinate for the non-reference frame, (*i*+*k*), respectively. $nrows$ denotes the maximal *y* coordinate for the non-reference frame, (*i*+*k*).

Step 2: Overwrite the pixel intensities for the reference frame *i* in the region of $a^{'}bcd^{'}$ by obtaining the actual pixel intensities for the reference frame *i* in the region of $abcd$, in which $x \in [Col_{\min}(i), Col_{\max}(i)]$ and $y \in [0, nrows]$.

$Col_{\min}(i)$ and $Col_{\max}(i)$ denotes the minimal and maximal *x* coordinate for the reference frame *i*, respectively.

Step 3: Obtain the actual pixel intensities for the non-reference frame, (*i*+*k*), in the region of $a^{'}b^{'}c^{'}d^{'}$.

Step 4: Difference the non-reference frame, (*i*+*k*), by the reference frame *i* in the region of $a^{'}b^{'}c^{'}d^{'}$.

Results for Dynamic Image Processing

Experimental Design

The same set of 300 frames, used for the road mask, was used for testing using dynamic image processing, ***in addition to*** applying the road mask. Upper and lower boundaries of the columns in each image were estimated. The number of tracked vehicles was compared with respect to RMVDT, and the improvement in the number of tracked vehicles with dynamic image processing with respect to RMVDT was also calculated.

Results

Figure 35 shows the estimated upper and lower boundaries of the columns. We can see the estimated upper and lower boundaries of the columns in the images move steadily at the same velocity, in the same direction. In addition, the differences between the corresponding upper and lower boundaries are equal to the width of the input image (720 pixels).

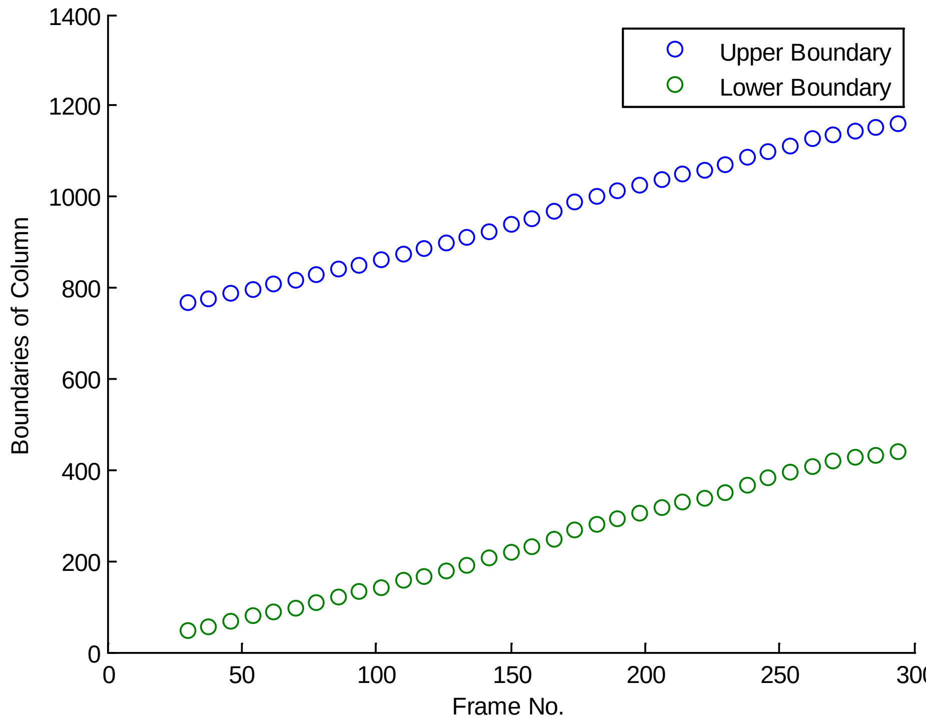


Figure 35: Estimated Upper and Lower Boundaries of Column

Figures 36 and 37 show the output images of the 150th frame without and with dynamic image processing, respectively. We can see only the region which overlaps with the reference frame is processed without dynamic image processing (Figure 36). The whole image of the registered frame is processed with dynamic image processing (Figure 37). This is the fundamental improvement by using dynamic image processing.

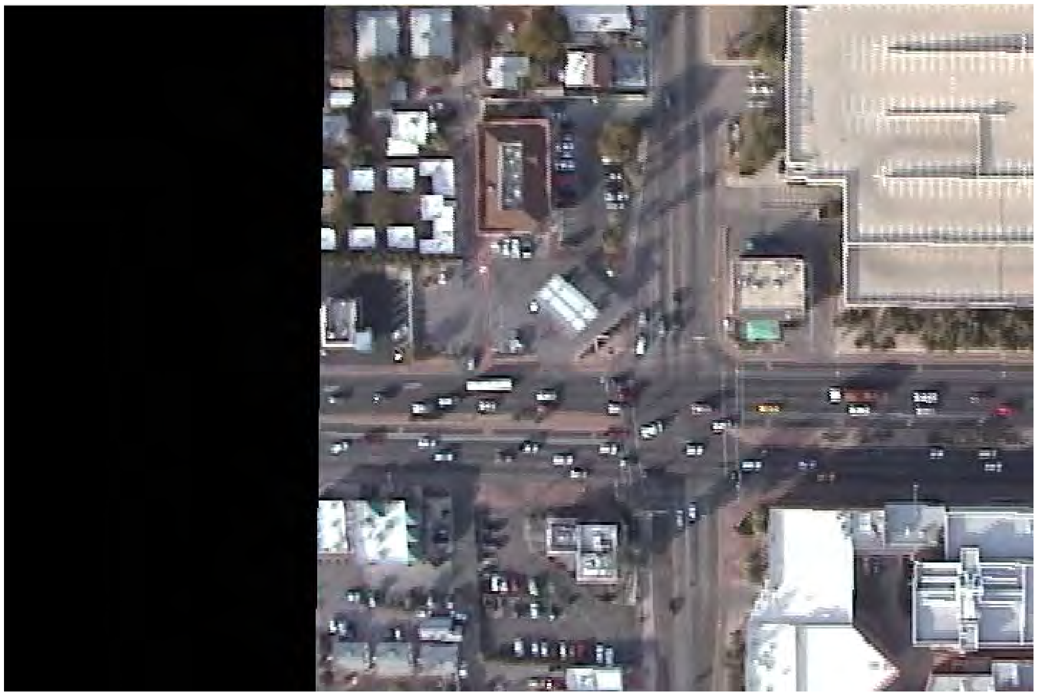


Figure 36: Output Image of 150th Frame (Without Dynamic Image Processing)

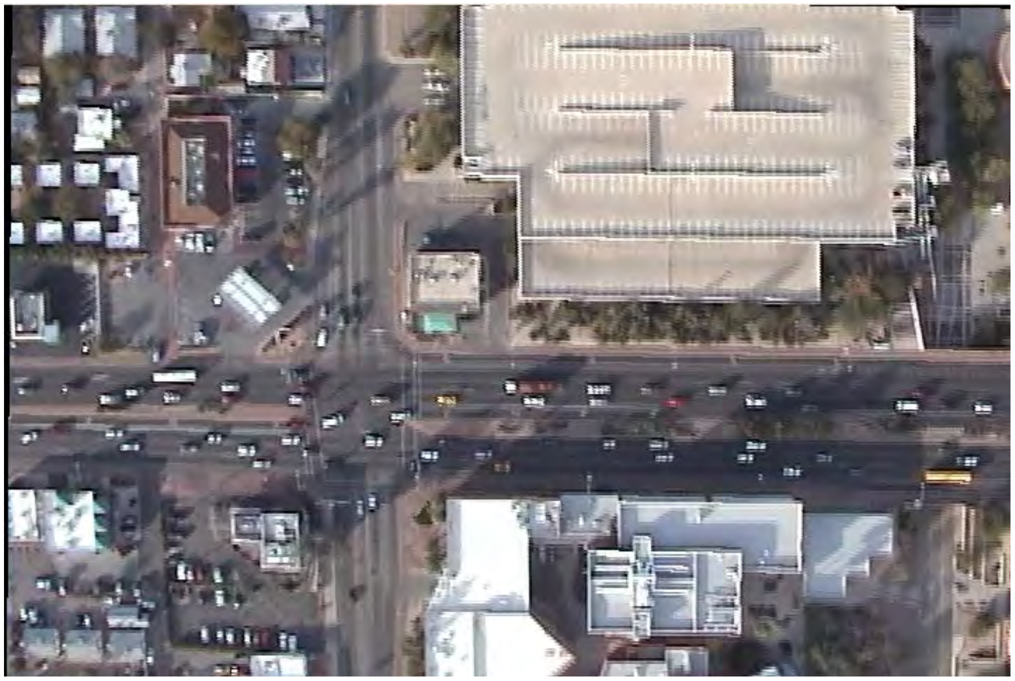


Figure 37: Output Image of 150th Frame (With Dynamic Image Processing)

Figure 38 shows the number of tracked vehicles, with and without dynamic image processing. We can see that the number of tracked vehicles decreases without dynamic image processing, while it remains steady and only fluctuates slightly with dynamic image processing.

This difference is due to the difference in areas captured in the two methods, with and without dynamic image processing.

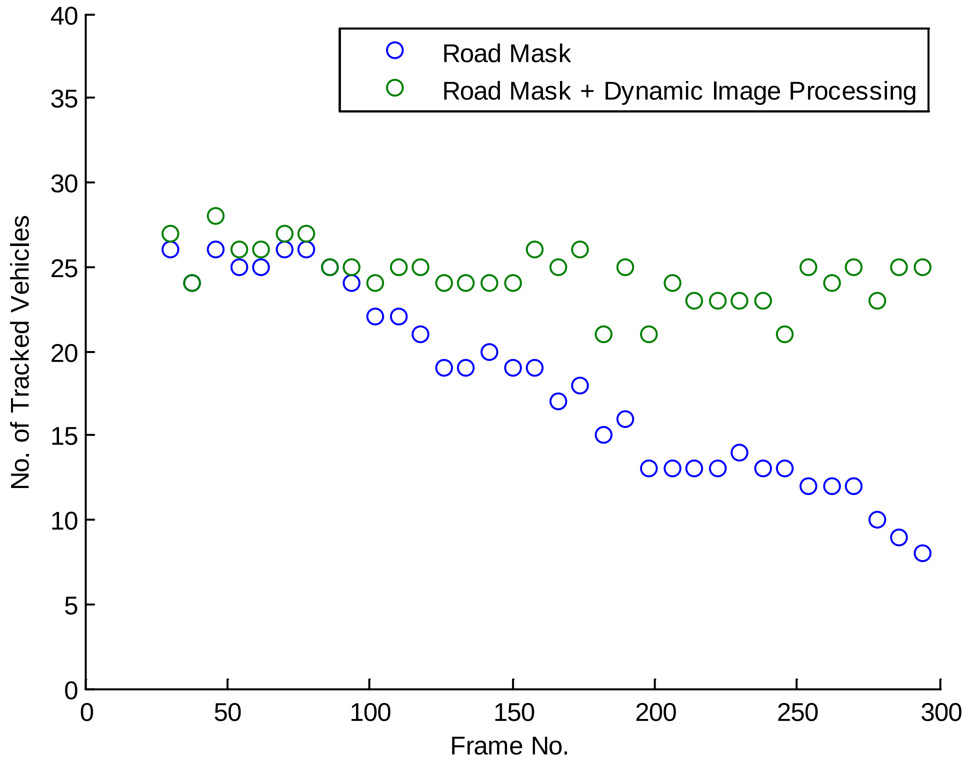


Figure 38: Number of Tracked Vehicles with and without Dynamic Image Processing

Georeferencing

One additional area of improvement with the TRAVIS software is the translation from image coordinates (pixels) to ground coordinates. The software is currently set up to use a single, image-dependent scale (e.g., feet per pixel) to convert pixel coordinates to ground coordinates. This image scaling has many shortcomings. Most prominently, the scale is constant across the entire image, so that any distortions in the image, such as radial distortion, camera not at nadir, etc., lead to incorrect estimates of vehicle positions. Secondly, the constant scale lacks a true reference point: while it is possible to estimate relative positions of objects, there is no absolute reference (or "ground truth") to interpret vehicle positions. That is, we have no way of knowing, without additional information, the precise geographic position of any object in the image.

Our goal in working toward a so-called georeferenced image was to be able to identify the absolute position of an object in the real world, through absolute ground coordinates (latitude/longitude). The conceptual approach was to register the airborne image, taken from a helicopter, with some additional georeferenced image. With access to such georeferenced imagery, the challenge then becomes to process both images to find common features, then to match these common features, and finally to match the images so that the ground coordinates in the georeferenced image could then be applied to the airborne image.

Specifically, the concept follows these steps:

1. Images are extracted from the airborne video.
2. Pre-processing is conducted on these images to improve the efficiency of map matching; e.g., by reducing the resolution of the images.
3. The first image in the video sequence is matched to the map, using some registration technique.
4. The consecutive airborne images are then registered to each other, using the existing registration of the KLT feature tracker in TRAVIS [26,41].
5. Once the images are registered, one may infer the georeferenced (map) coordinates onto each airborne image.
6. The road mask in TRAVIS may be applied to the airborne images, to limit the computational processing required for vehicle tracking.
7. The remaining TRAVIS code is applied to detect and match vehicles across the airborne video image sequence.
8. Vehicle trajectories in ground coordinates can then be derived directly from the coordinates of each image. Trajectories can also be displayed on a georeferenced map, such as Google Earth or other similar imagery.

This system was implemented using the SIFT (Scale-invariant Feature Transform) algorithm [43]. However, the specific integration into TRAVIS was not completed in this project.

1.

Sensor Fusion of Airborne with In-ground Sensor Data

A major task of the project was to develop methods to fuse data from airborne sensors with ground data. The initial focus was on freeway data. The traffic system is a highly dynamic and complex nonlinear system, even for a non-controlled freeway stretch. Traditionally, the data used for freeway traffic state estimation are collected through limited number of ground sensors located at some specific positions along the freeway stretch. This type of data has their intrinsic noise which would deteriorate the estimated results. In this research we focus on the exploration of how to utilize the high accuracy picture data for a freeway stretch collected by a surveillance helicopter to enhance traffic state estimation.

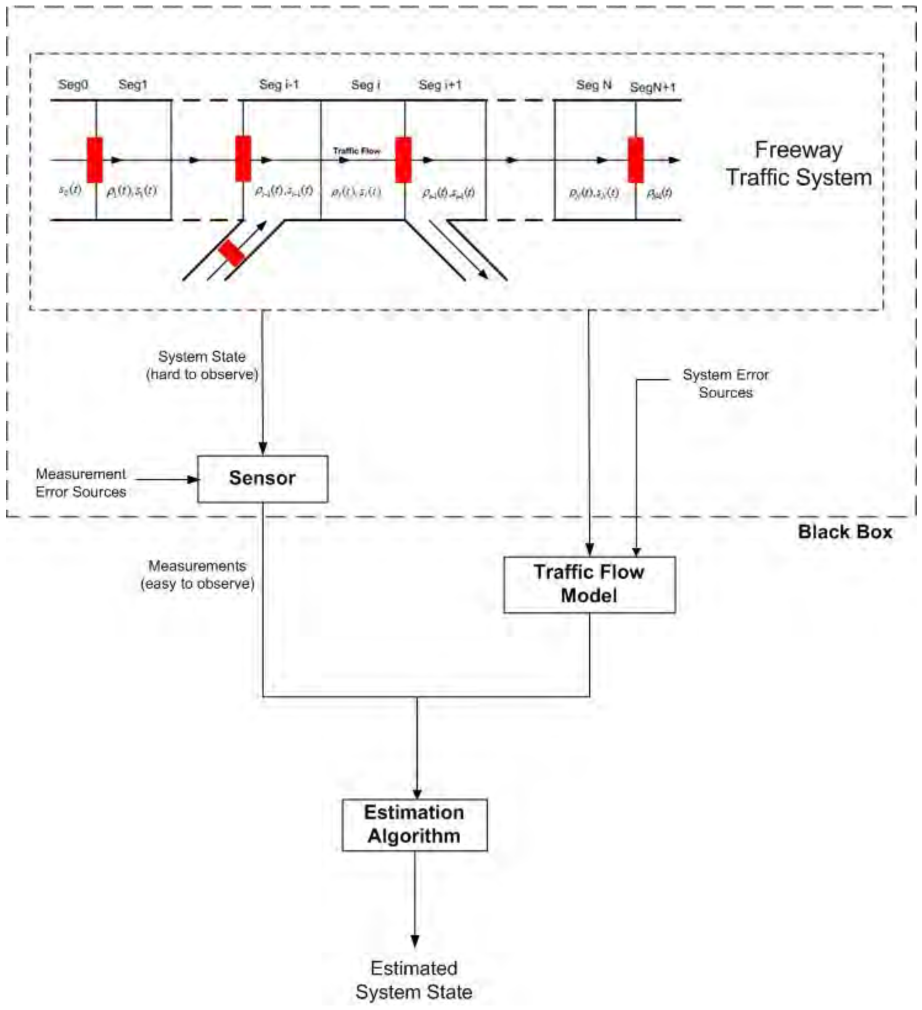


Figure 39: Illustration of State Estimation Procedure

The Generic Traffic State Estimation Problem

System state estimation is a major branch in control area. It refers to approximating a vector of system state that is hard to measure given some noisy measurements that are much less expensive to observe. A system model, constituted of the system dynamics model and the measurement model, and an estimation algorithm are combined to estimate the system state and reduce noises using limited measurement data. Regarding transportation systems, this includes three main components, a) an underlying traffic flow model and a measurement model, b) an applicable estimation algorithm, and c) sufficient measurement data, as is depicted in Figure 39 The freeway traffic system is what we are interested in but cannot be observed, which along with error sources and sensor behavior are all in the black box on the upper side. Traffic flow model and measurement models are imperfect abstracts of the real world system. The estimation algorithm incorporates measurements, the output of sensors, and the system model to produce estimated system state.

Traffic Modeling

Traffic modeling for the state estimation includes modeling the evolution of traffic regarding time and space, and modeling the relationship between measurement variables and state variables. The measurement model is more data-type dependent and varies greatly among different estimators. On the other hand, modeling transportation system is itself a major topic in transportation area. Regarding the levels of fidelity from low to high, traffic models can be categorized as macroscopic, mesoscopic and microscopic models.

Macroscopic traffic models focus on the aggregated traffic state statistics, such as traffic flow and traffic density. Since this requires the least level of details in measurement data, a great number of approaches estimating macro-freeway traffic state based on various macroscopic traffic flow models and diverse estimation algorithms have been developed with satisfactory performances. However, as is known to all, in macroscopic level, computational efficiency is appreciated at the cost of detail information, which is not always enough to meet the need in dynamic assignment.

In the other extreme, the microscopic traffic models emphasize on approximating individual vehicle behavior in the real world as detail as possible. Obviously, given the highly dynamic nature of vehicle behavior as time evolves and the large number of vehicles in count, it is significantly computational expensive. In fact, individual vehicle behavior may be too noisy to be useful, as it is influenced by such a great number of known or unknown factors.

A good tradeoff between computational cost and level of fidelity can be realized by mesoscopic traffic models, which lies between the macroscopic level and the microscopic level. This type of models provides information more detailed than traffic state statistics but filters out microscopic behaviors like lane changing. One can immediately notice that the so called mesoscopic level ranges very wide, so long as it between the macroscopic and microscopic levels. In this research we define the fidelity as the temporal-spatial relationship for each vehicle, but neglecting other behaviors beyond.

Estimation Algorithms

A bunch of estimation algorithms have been developed and applied to all engineering area. Kalman Filter (KF) is proved that can solve the state estimation problem optimally if the system is linear and noises are white and Gaussian. In traffic estimation area, extended Kalman filtering (EKF) and other variation of Kalman filtering are credited the most popularity for the nonlinear transportation system. Particle filter (PF) is found to be helpful when the system is extremely nonlinear and the basic Gaussian assumption of system noises is corrupted. However, a major drawback of these types of estimation algorithms is that constraints cannot be incorporated. A more flexible one attracting increasing attention is the Moving Horizon Estimation (MHE). MHE solves the estimation problem as a minimum squared error optimization problem sequentially using a moving time horizon. In this research, the particle filter and extended Kalman filter algorithms are selected to estimate the macro traffic state with a comparison, whereas the least squares optimization of estimation error method is applied to the mesoscopic vehicle trajectory estimation.

Measurement Data

Traditionally, measurement data for traffic flow estimation is collected from ground sensors. Although it is obvious that aggregated information of traffic flow is not enough for all intentions, research on estimation of more detailed vehicle trajectory is hindered by the type of data that sensors can provide. As the technology develops, aerial data gathered from helicopters are available to provide more information of freeways, either of the same observation variables as the ground detectors but from independent source, or even about different features that ground sensors cannot measure. By combining aerial data, it is hoped that the accuracy and efficiency of the traffic state estimator can be improved with significance. Figure 40 illustrates the measurement data collected by remote sensors. Filled black squares in the image represent captured vehicles.

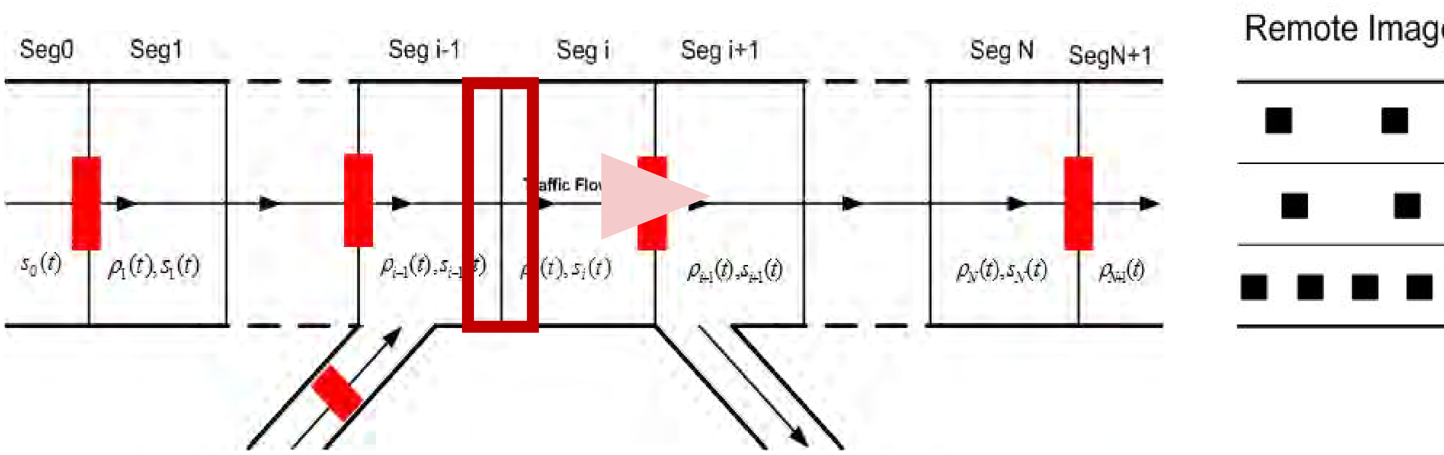


Figure 40: Illustration of Remote Sensing

This research explores the advantages of incorporating extra accurate data collected by helicopters. This source of data is distinguished from traditional ground-sensor-based data in the following aspects. (a) Accuracy. Airborne data are extracted from images taken by helicopters, which would yield very high accuracy and almost zero noise. (b) Mobility and flexibility. Unlike traditional ground sensors that are pre-located underground, one can decide whether to monitor a specific freeway stretch, which area is going to be monitored by the helicopter, and thus can have more flexibility on collecting data. (c) Adjustable helicopter speed. The relative speed of the surveillance helicopter to the ground traffic flow impacts which platoon of vehicles are tracked, and would further have an influence on the estimated traffic states. In fact this results a related problem of optimizing surveillance helicopter speed, which we would discuss shortly later, but for the rest part of this report we simply assume some default tracking time for each vehicle. The most significant disadvantage for using remote sensors is that it is more expensive than the once and for all installation fee of ground sensors. But since high accuracy estimation is needed in many occasions and the extra cost is acceptable, this research still has its merits.

Freeway Traffic State Estimation Problem

In this research we focus on the freeway traffic state estimation problem, regarding both macroscopic and mesoscopic levels, using data collected by both ground sensors and remote sensors. In macroscopic level the efforts aim at enhancing the estimation of real-time traffic state statistics and prediction of the future state by utilizing the high accuracy remote sensing data, while in mesoscopic level smoothing historical and predicting future vehicle trajectories are the objectives.

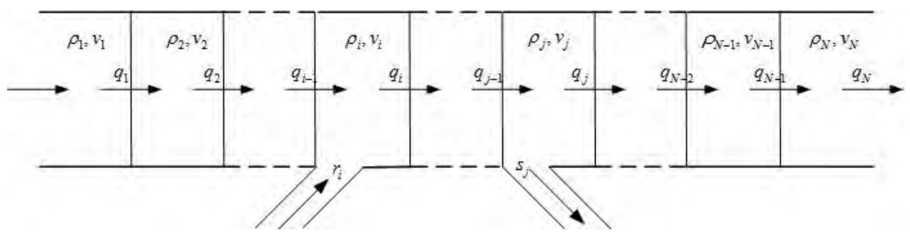


Figure 41: A Freeway System

The evolution of a freeway traffic system is continuous in time and space, but researchers found it very helpful to discretize it into segments and into discrete time. Figure 41 shows a freeway stretch that can be considered as an interactive chain consisting of N segments. Each segment has at most one on-ramp and/or one off-ramp. Limited ground detectors are installed in the stretch in order to measure observable features represented by observation variables $Y_1(t)$, where t is the time index referring to a time step in the discretized time space. Measurement data collected by remote sensors are denoted as $Y_2(t)$. These measurements are essentially determined by the state features of the segments, such as the traffic flow q, traffic density r, the mean-space speed v, onramp flow r and offramp flow s, represented by the stochastic state variables $X_1(t)$. The macroscopic traffic model is to explore the dynamic relationship and evolution of the discrete-time macro-state variables characterizing the segments over time and space, for example, traffic flow, traffic density and mean-space speed. Based on this model and a measurement model, the macroscopic estimator then can apply the particle filter algorithm and is expected to estimate traffic states using measurement data with high accuracy.

Matching the vehicles in a series of images would yield partial trajectories or sparse points along their trajectories. Optimally filling the gaps between these measurements is the objective of our mesoscopic estimator. In this case, only the time dimension is discretized, denoted by time index k. Note that the length of time steps could be different between macroscopic and mesoscopic levels.

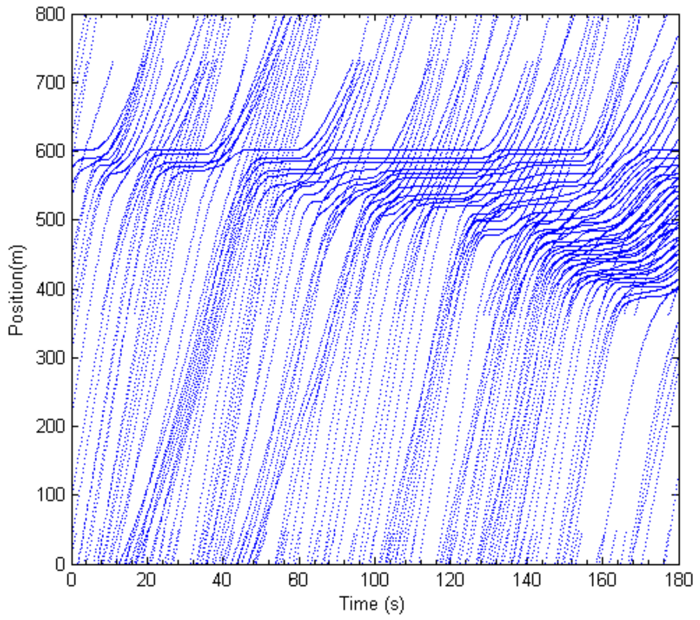


Figure 42: Vehicle Trajectories

Figure 42 shows an illustration of vehicle trajectories travelling through a single-lane multi-entrance multi-exits freeway. Each trajectory starts at the beginning of the stretch or at the position of onramps, and ends at the end of the stretch or offramps. Initially, the congestion is appearing at around 600 meters away from the origin. As time elapse, the congestion area is enlarged to be between 400m and 600m. Obviously, if one can estimate each of these trajectories satisfyingly, the whole picture of the traffic condition would be easily obtained. In this research, each of the trajectories will be estimated separately. A single trajectory is made up of sequential time-space points representing the location of the vehicle at any given time, or when the vehicle gets to the position for any given location. For each single vehicle, its movement is highly influenced by the traffic condition of a specific segment it is travelling through. But its detailed behavior is determined by many uncontrollable factors such as the driver's psychological activity. Thus the mesoscopic model is going to explore its temporal-spatial relationship given macroscopic traffic conditions and ignore the too detailed behaviors by treating them as the system noise. Complete estimated state information about the macroscopic traffic state in the entire domain of time and space, including density, speed and traffic flow at any location at any time within the domain under investigation are assumed to be available but may not be accurate; in fact these are the results of the macroscopic estimation. Based on the mesoscopic traffic state model and a measurement model, the mesoscopic estimator can approximate the vehicle trajectories given macro-state data and image measurements at some sparse points.

The above two are the core problems of this research, but there are also other related sub-problems or possible advanced problems. Helicopter speed optimization, Origin-Destination (OD) flow estimation and ramp metering control rate optimization are the major ones.

The rest of this chapter is organized as follows. First we present a short literature review, followed by discussions on the scope, framework and the methodology of this research. Detailed results on the mesoscopic vehicle trajectory estimator is provided afterwards with experimental results and discussions.

Literature Review

Wang and Papageorgiou [44] proposed and joint estimation approach for macroscopic traffic states, boundary variables and critical model parameters at the same time. Traffic density and mean speed in each segment are defined as states, while traffic flow and space-mean speed in sensor-located segments are defined as measurements. METANET model and EKF approach are adopted to realize the real-time estimation. This is empirically tested to be satisfactory given the consideration that only ground-sensor based data are available. Hegyi et al. [45] show that based on METANET second-order traffic flow model, the Unscented KF (UKF) is slightly or equal in the performance to EKF. The performance of joint filter is better than that of the dual filter. Fewer detectors result in larger state estimation errors but have little effect on the parameter estimation error.

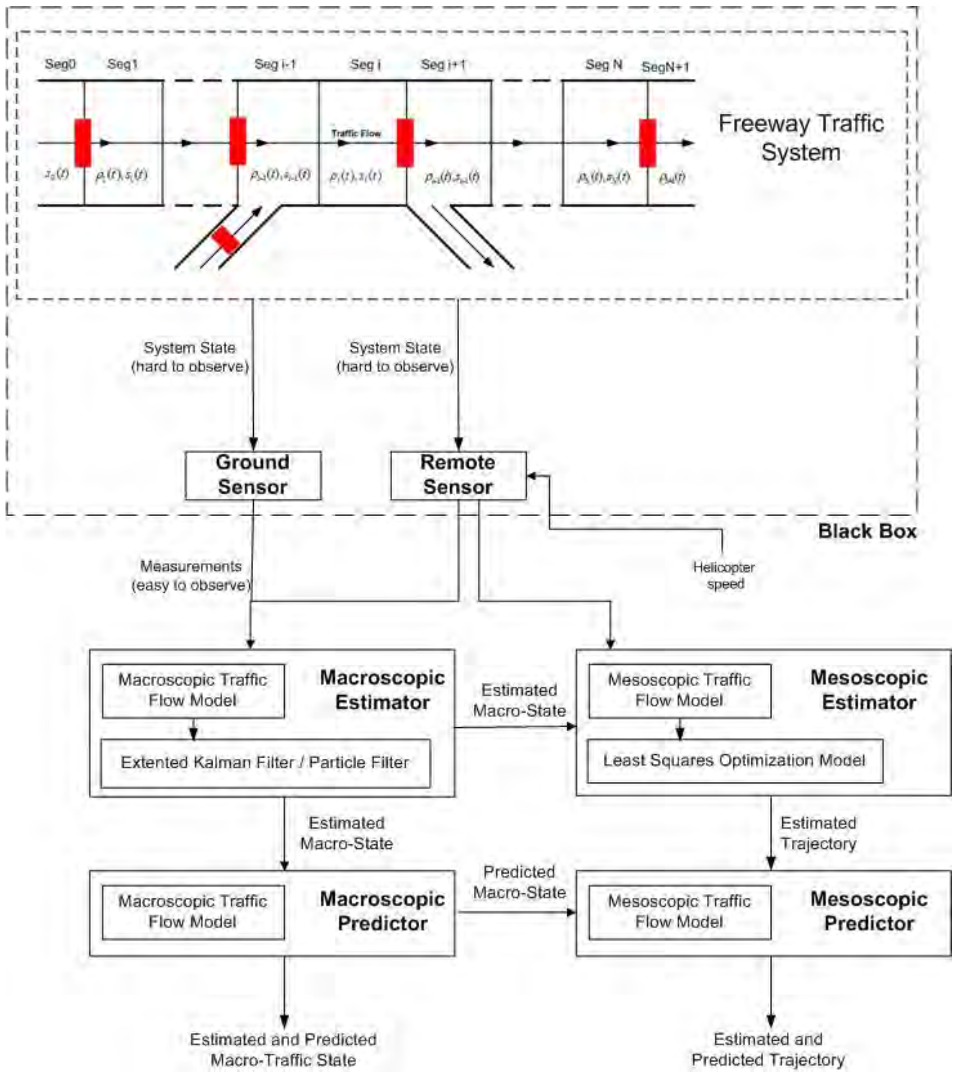
Particle filter (PF) is a relatively new method to replace EKF, but may require much more computation. Mihaylova and Boel [46] develop the particle filter application in traffic state estimation. They investigate the performance of PF and Unscented KF and conclude that PF performs better than UKF. Sun, Munoz and Horowitz [47] develop the mixture Kalman filter for ramp metering control based on first order traffic model and obtain the average mean percentage error of approximately 10%. Ensemble KF (EnKF) may be a good substitute of EKF among the all considering both the performance and efficiency. Work et al. [48] use it in highway traffic estimation using GPS enabled mobile devices.

Obviously, although all of these researches are trying to get better estimate of system states, no attention is paid to exploring incorporating the remote sensing data. For mesoscopic vehicle trajectory estimation and prediction, there is rarely any literature focusing on this topic, let alone the investigation of how to take advantages of aerial data.

Research Methodology

Logic Chart for the Framework

Figure 43 shows the framework and scope of this research. Both macroscopic and mesoscopic levels of traffic state are considered, where within each level both an estimator and a predictor are built. The final output should be estimated and predicted macroscopic traffic state and estimated and predicted mesoscopic vehicle trajectories. The helicopter speed serves as an input for the remote sensors. The output of both levels can be further used as inputs to related problems such OD flow estimation and ramp metering control problems.


Seg0
Seg1
Seg i-1
Seg i
Seg i+1
Seg N
SegN+1
Traffic Flow
Freeway Traffic System
System State (hard to observe)
System State (hard to observe)
Ground Sensor
Remote Sensor
Black Box
Measurements (easy to observe)
Helicopter speed
Macroscopic Traffic Flow Model
Macroscopic Estimator
Extented Kalman Filter / Particle Filter
Estimated Macro-State
Mesoscopic Traffic Flow Model
Mesoscopic Estimator
Least Squares Optimization Model
Estimated Macro-State
Estimated Trajectory
Predicted Macro-State
Macroscopic Traffic Flow Model
Macroscopic Predictor
Mesoscopic Traffic Flow Model
Mesoscopic Predictor
Estimated and Predicted Macro-Traffic State
Estimated and Predicted Trajectory
Figure 43: Logic Chart for the Framework

Typical methods for filtering a multi-sensor system

Assuming the data are synchronously collected, three methods can be applied to the multi-sensor system, namely, parallel filter, sequential filter and data compression [49]. The following provides a general introduction of how to solve the multi-sensor problem, but in our research only the parallel filter method is adopted.

Let x_{k+1} denote the system state with covariance P_{k+1} and $y_{k+1,i}$ denote the measurement taken at time k+1 from sensor i, i=1 if ground data and 2 aerial data, and $v_{k+1,i}$ is the white Gaussian noise with covariance matrix $R_{k+1,i}$ conditionally independent of the other detector's covariance matrix.

The measurement vector then become

$$y_{k+1} = [y_{k+1,1} \quad y_{k+1,2}],$$

and the covariance matrix is

$$R_{k+1} = \begin{bmatrix} R_{k+1,1} & 0 \\ 0 & R_{k+1,2} \end{bmatrix}$$

a) Parallel Filter: Parallel filter is to process measurements in multi-filters at the same time.

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k+1} + \sum_{i=1}^{2} K_{k+1,i}(y_{k+1,i} - h_i(\hat{x}_{k+1|k+1}))$$

The Kalman Gain matrices $K_{k+1,i}$ are calculated according to specific KF algorithm.

b) Sequential Filter: The sequential filter is to update filter sequentially with data blocks from difference sources

$$\hat{x}_{k+1|k+1,i} = \hat{x}_{k+1|k+1,i-1} + K_{k+1,i}(y_{k+1,i} - h_i(\hat{x}_{k+1|k+1,i-1}))$$

c) Data Compression: The data compression method is to compress similar data first and then use them to update the filter for once at each time interval.

On the Macroscopic Level

Models

The cell transmission model [50] represents the system evolution of freeway traffic over time and space using easy-to-solve difference equations. The model has four degrees of freedom, specifically, the free flow speed, the maximum flow, the jam density, and the wave speed. METANET [51] is another widely acknowledged macroscopic traffic flow model relating the current traffic density and speed state to the state at the previous time step with a closed form of formulation. Both are empirically tested to be satisfactory.

Filtering Algorithms

Presently most of the related researches are conducted based on some variations of the Kalman Filter Algorithm. In this part a general idea of Kalman Filter is introduced. This is the basis for any variation of KF, such as the widely applied EKF. Consider the system

$$\sum(x,y): \begin{array}{r} x(k+1) = f[x(k), u(k), \xi(k)] \\ y(k) = h[x(k), \eta(k)] \end{array}$$

where $x(k)$ is the unknown state vector with covariance matrix $\mathrm{P}(k)$; $u(k)$ is the known input vector; $y(k)$ is the measured output vector; $f[x(k),u(k),\xi(k)]$ and $h[x(k),\eta(k)]$ are system dynamic function and observation function respectively; $\xi(k),\eta(k)$ are the independent white Gaussian noise vectors with known covariance matrix $Q(k),R(k)$.

Prediction (Time Update)

(1) Project the state ahead

$$\hat{x}(k+1|k) = F(k)\hat{x}(k|k) + G(k)u(k)$$

(2) Project the error covariance ahead

$$P(k+1|k) = FP(k|k)F^T + Q$$

Correction (Measureme

$$K = P(k+1|k)H^T(HP($$

(3) Update Error Covariance

(1) Compute the Kalman

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + K\big(y($$

(2) Update estimate with me

$$P(k+1|k+1) = (I - KH)$$

Figure 44: Illustration of Kalman Filter Procedure [52]

The original KF is designed for linear system. If $f[x(k), u(k), \xi(k)]$ and $h[x(k), \eta(k)]$ are linear functions, the system become

$$\sum(x, y): \begin{array}{c} x(k+1) = F(k)x(k) + G(k)u(k) + \xi(k) \\ y(k) = H(k)x(k) + \eta(k) \end{array},$$

In this case, KF is proved to be able to get the best estimate of $x(k)$ using the recursive procedure shown in Figure 41. On the other hand, if $f[x(k), u(k), \xi(k)]$ and $g[x(k), \eta(k)]$ are nonlinear, EKF, UKF or other alternative methods should be adopted.

On the Mesoscopic Level

In this part of the research we show that with the help of high accuracy data collected from helicopters we are able to estimate the trajectory of a vehicle traveling on a freeway superior to an estimator without airborne data. We optimize locally along the vehicle trajectory using a least-squares model. By locally we mean the optimization of trajectory estimation problem is decomposed into several same problems but with shorter time horizon. The optimization time horizon is defined as the time length between any two points where the vehicle is captured by the remote sensor. The LS model is applied every time when the vehicle is captured by the helicopter, and finally we obtain the estimated full trajectory by linking up all these segments of trajectories.

Other Related Problems

Optimizing helicopter speed

Suppose the helicopter flies in the same direction as the ground vehicle flow does. Which platoons of vehicles are tracked is influenced by the relative velocity of the monitoring helicopter regarding to the speeds of ground traffic. Naturally, we would ask what the optimal helicopter speed is regarding to obtaining the most amount of “helpful” information of ground traffic flow. By “helpful” we mean that the new data would provide some unknown information rather than just repeat the information can be obtained from ground sensors or our traffic flow model. Another important objective in investigating the optimal speed is to ensure the observability of system state. This is a big issue influences state estimation performance.

Real time Origin-Destination (OD) flow estimation

Real time OD flow is the critical input for real time traffic control systems. Unfortunately, this problem is usually a highly underestimated problem. Consider a general case of n pair of entrances and exits. The number of OD flow state variables would be n2 at each time step, while the observations obtained from ground sensors cannot be enough to ensure observability of system state. New data source from remote sensors may help solving this problem.

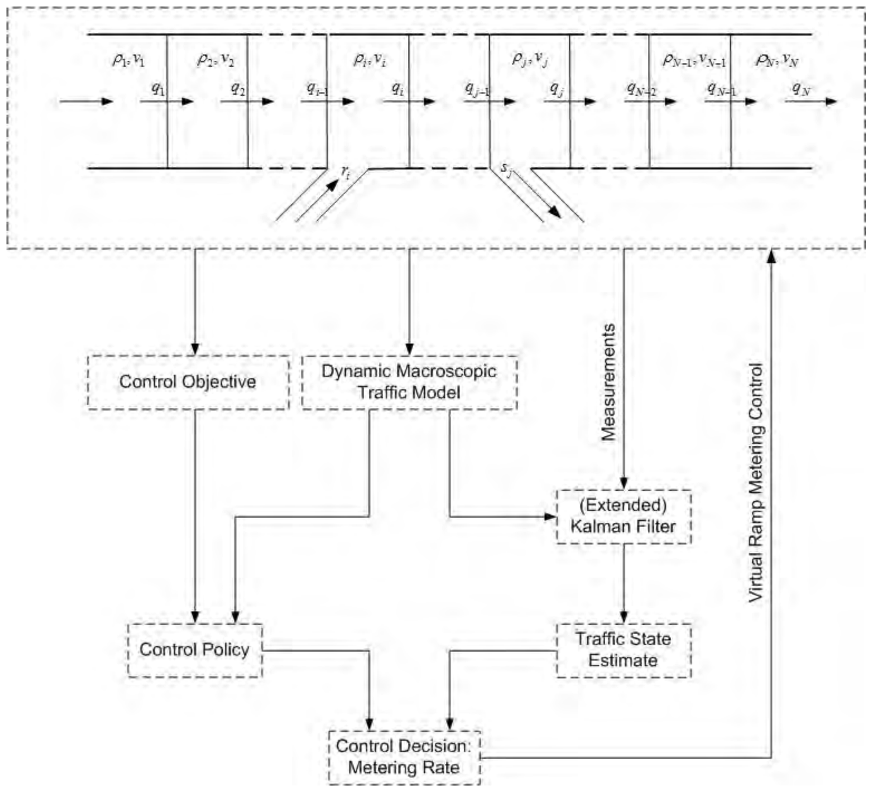


Figure 45: Logic Chart of Ramp Metering

Ramp metering control rate

The freeway was designed to provide unlimited transportation service to drivers. However, as the number of vehicles increases, recurrent and non-recurrent congestions are appearing on freeways more and more frequently. By introducing the philosophy of traffic lights to freeways, appropriate ramp metering control can lead to substantial amelioration of congestions through a trade-off between traffic demand satisfaction and freeway utilization efficiency under flow capacity constraint. It can also have impact on drivers' route choice behavior in the long term as drivers expecting a long queue at the on-ramp may choose available alternative routes. Ramp metering control allows only one vehicle to enter the freeway mainstream each time when the traffic light is green. The time intervals for the light phases stand for certain metering decision limiting the on-ramp stream to the mainstream. It can be shown that ramp metering control can improve the exiting flow in congestions, leading to reductions in OD total travel time.

As is shown in Figure 45, the problem of metering rate decision is divided into two problems, i.e. control policy and traffic state estimation; indeed, the OD flow estimation is the critical input for the controller.

Freeway Traffic State Estimation Model

Mesoscopic Vehicle Trajectory Modeling

In mesoscopic level defined in our research we focus on the temporal-spatial relationship for each individual vehicle while all microscopic behaviors like lane changing are neglected. From this view one only cares about at each time step where the vehicle is located in the freeway stretch. Consider a single vehicle traveling on a freeway, its movement follows the theory of classical mechanics. Theoretically, once one defines a time and space origin, the specific spatial position of the vehicle can be determined by an integration of its instantaneous velocity over traveled time.

$$x(k) = \int_0^k v(h)dh$$

where k refers to a continuous time point, x is the position measured from the start point and v is the instantaneous speed. However, measuring instantaneous velocity alone the whole trajectory is unrealistic, especially when we are interested in the movements of a large number of vehicles; a more sensible way is to collect data only on a limited number of points in the time-space domain and estimate the rest part of the trajectory. First we discretize the time space into time steps, each of which has a length T. Let k denote the time step index. The discrete vehicle model can be described as

$$\begin{aligned} x(k+1) &= x(k) + v(k) * T \\ v(k+1) &= v(k) \end{aligned} \tag{31}$$

This model literally states that at time step k+1, the vehicle is located $v(k)*T$ meters ahead of the position where it was at time step k, while the speed at time k+1 is the same as the previous time step. Let ξ_x and ξ_v be the normally distributed noise vector related to the position model and the speed model, and $\xi_x(k)$ and $\xi_v(k)$ be their elements. Adding the stochastic noise term to the model, we obtain a model that is referred as system dynamics (Equation 32). This model states that model in Equation 31 is not a perfect model so we need to use the noise term to describe the difference between the system state described in Equation 31 and the real world system state.

$$\begin{aligned} x(k+1) &= x(k) + v(k) * T + \xi^x(k) \\ v(k+1) &= v(k) + \xi^v(k) \end{aligned} \tag{32}$$

It is assumed that the vehicle is captured by the camera in helicopter at some fixed time points. These time points naturally separate the whole trajectory into pieces. Each piece can be considered independent since we know exactly where the vehicle is at the start point of each piece. It is the same as estimating a new trajectory from these captured points. Each of the optimization horizon ends when the vehicle gets tracked again. Note that the instantaneous speed may be available when the vehicle gets tracked. Equation 33 describes that at the start time and the end time the vehicle is tracked by the helicopter for twice.

Note these set of measurement model does not include noise terms since these measurements are obtained from remote sensor. We highly trust the accuracy of remote sensors, and the noises would be small enough to be neglected compared to the significance of noises in ground sensors.

$$\begin{aligned} y^x(k_0) &= x(k_0) \\ y^x(k_N) &= x(k_N) \\ y^v(k_0) &= v(k_0) \\ y^v(k_N) &= v(k_N) \end{aligned} \tag{33}$$

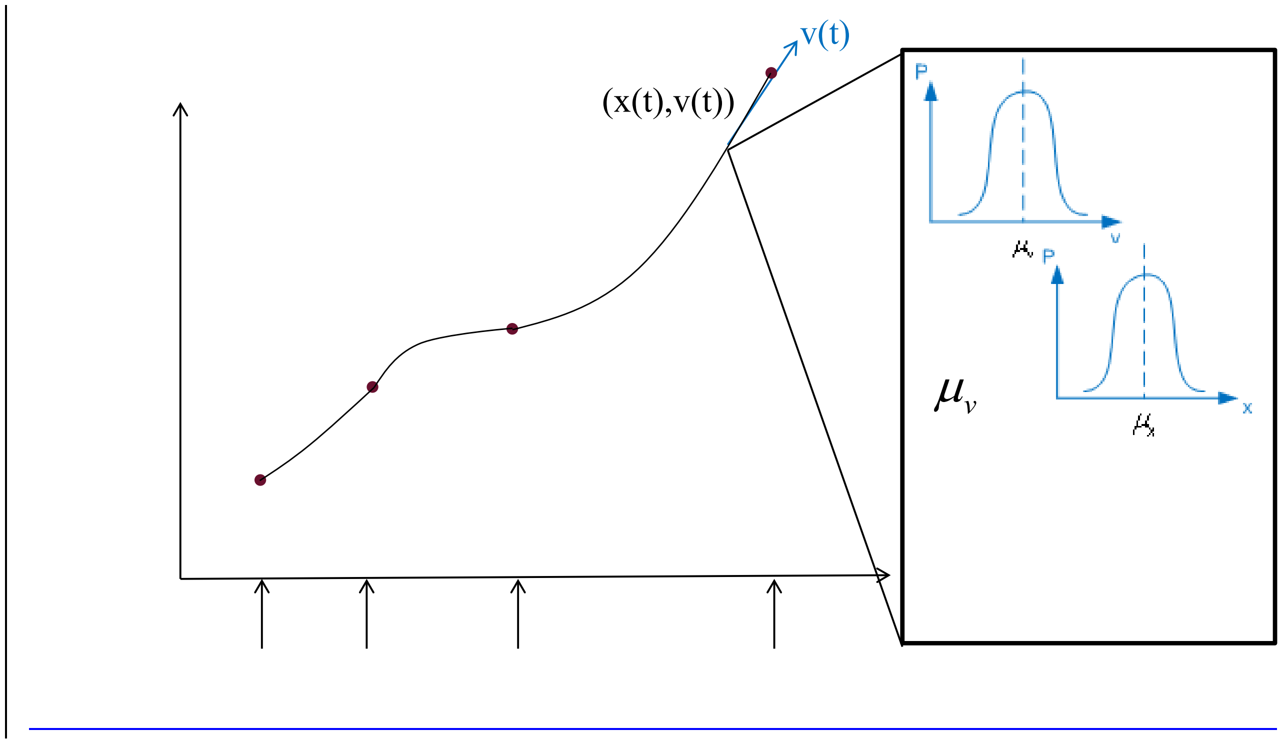


Figure 46: The Single Vehicle Trajectory Estimation Problem

Assumption1: The instantaneous speed at time k of a vehicle follows a normal distribution with a mean of the average speed of the segment it is travelling through at the time k and certain variance. (Figure 46)

The freeway is segmented into cells, each of which is approximately as long as the average travel length of vehicles in one time step T at the average velocity (assumed as the speed limit) V. Within each segment cell the freeway is assumed to be homogeneous, i.e., density and velocity do not change much. The speed of the vehicle highly depends on the average speed on the segment it is travelling through, especially when congestion occurs. Therefore, this assumption is reasonable.

Let η_v be the noise of speed measurements, which is in fact the difference between real instantaneous speed of the vehicle and the estimated mean speed of the segment where the vehicle is located, and $\eta_v(k)$ be its element at time step k. The measurement model for relating macroscopic data to vehicle state variables is as follows.

$$y^v(k) \quad = v\ (k) + \eta_v(k) \tag{34}$$

It is assumed that all macroscopic data are available but with a much longer time step than that is used in estimating trajectories. For example we may use T= 0.2 second for trajectory estimation but use TS=15 seconds in macroscopic traffic flow estimation. Since the mean speed of segments are used as the measurement of instantaneous speed for individual vehicles, a much larger TS implies that the measurement for speed is piecewise linear and the change points in measurement data are sparse. In experimental analysis we would see that even given this limited amount of information one can still get a good estimation of the vehicle trajectory. Equations 33 and 34 together constitute the measurement model.

Assumption2: The vehicle trajectory estimation problem can be separated into several independent same-type sub-problems.

This assumption is adopted to facilitate the estimation, but may lead to loss of historical information. Still in Figure 46, dark dots on the trajectory represent where the vehicle is tracked. In this figure we have four dots, forming three sub-problems. Those sub-problems are solved consecutively and by linking up the estimated trajectory fragments, one obtains the whole estimated trajectory.

Least Squares Approach

A least squares model is built up to optimize the estimation of trajectory given available measurements. Let Q_x^{-1} , Q_v^{-1} and R_x^{-1} be the inverse of the covariance matrices for ξ_x , ξ_v and η_v . Denote the start time of one time horizon as k_0 and the end time of that horizon as k_N .

$$\begin{aligned}
&\min \Phi = \xi_x^T Q_x^{-1} \xi_x + \xi_v^T Q_v^{-1} \xi_v + \eta_v^T R_v^{-1} \eta_v \\
&\quad s.t. \\
&\qquad x(k+1) = x(k) + v(k) \cdot T + \xi_x(k) \quad \text{for all } \mathrm{k} \in \mathrm{K_i} \setminus \mathrm{k_N} \\
&\qquad v(k+1) = v(k) \qquad\qquad\quad + \xi_v(k) \quad \text{for all } \mathrm{k} \in \mathrm{K_i} \setminus \mathrm{k_N} \\
&\qquad y^{\mathrm{x}}(k_0) = x(k_0) \\
&\qquad y^{\mathrm{x}}(k_N) = x(k_N) \\
&\qquad y^{\mathrm{v}}(k_0) = v(k_0) \\
&\qquad y^{\mathrm{v}}(k_N) = v(k_N) \\
&\qquad y^{\mathrm{v}}(k) = v(k) \qquad\qquad\quad + \eta_v(k) \quad \text{for all } \mathrm{k} \in \mathrm{K_i} \setminus \{\mathrm{k_0}, \mathrm{k_N}\} \\
&\qquad x(k+1) - x(k) \geq 0 \\
&\qquad v(k) \geq 0
\end{aligned} \tag{35}$$

Equation 35 minimizes the estimation error, which is the summation of system error and the measurement error. First two constraints are system dynamics from eqn.2. The next five constraints are measurement equations from equations 33 and 34. The last two constraints are constructed to preclude absurd results, which cannot be realized in traditional filtering algorithms such as Kalman filter. Note that in equation 35 the start instantaneous speed is assumed known, but in the experimental analysis, results from equation 35 is compared with results from a modified model which do not have the information of exact start speed.

Experimental Analysis

In this section we present the experimental results. TS is set to be 15s and T is set to be 0.2s. The captured points along the trajectory separate the whole trajectory into eight pieces. The true vehicle trajectory data is simulated in VISSIM. The estimation algorithm is realized in MATLAB and the least-squares model in IBM-ILOG OPL software.

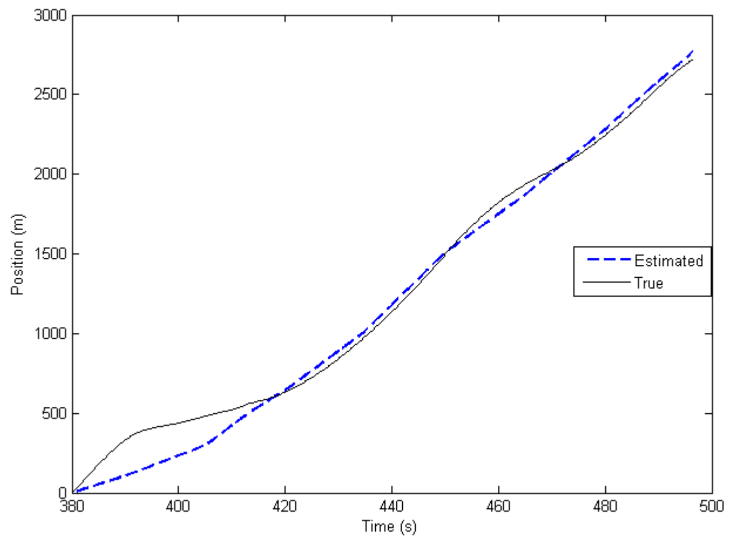


Figure 47. Vehicle Trajectory Estimation without Airborne Data

The vehicle enters the freeway main stretch at time 380s from the initial point of that stretch, and gets off at 495s around 2750m from the initial point. Figure 47 shows the estimated trajectory without any information from the remote sensor, which means all measurements come from the mean speed of macroscopic traffic statistics data. It is obvious that from time t=380s to time t=420s there is a large gap between estimated trajectory and the true trajectory. This is because of the large deviation of vehicle speed from the mean speed of freeway segments it travelled through. The rest part of the estimated trajectory is not bad. This may imply that the vehicle speed is much closer to the mean speed of freeway cells. But it may also results from the cancellation effect of a negative deviation from true position at the previous time step and a positive deviation from true instantaneous speed, or vice versa.

Figures 48 and 49 show how much improvement we can get through the least-squares optimization model. In Figure 48 we assume that the initial instantaneous speed in each local optimization horizon is unknown. It is clear that before time 420s the estimated trajectory gets much closer around the true trajectory. However, at time t = 410s an unexpected gap rises. By checking data we believe that this is due a large difference between vehicle's instantaneous speed and the mean speed of the corresponding segment, and the fact that the limited amount of measurement data is not enough to ensure observability of system state. It may also result from some aberrant microscopic behaviors. Similar case happens at time 460s, where the gap between estimated trajectory and the true trajectory is even enlarged. These problems imply that we may need additional measurements to enhance the estimation.

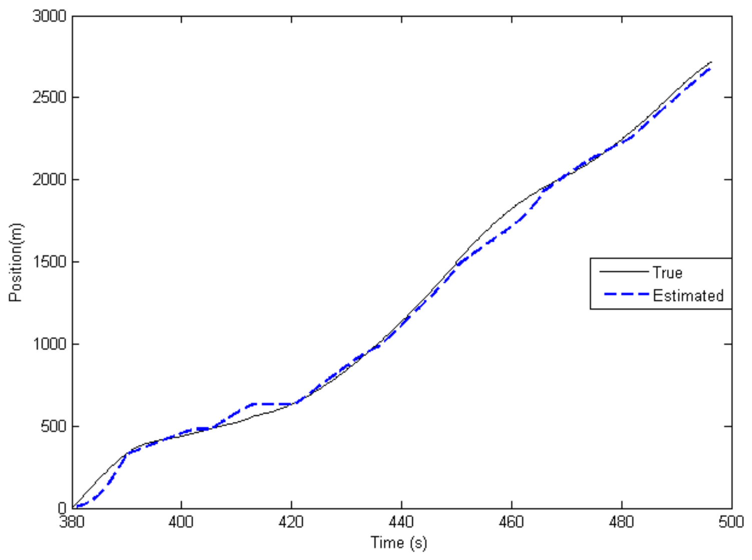


Figure 48: Vehicle Trajectory Estimation Using Airborne Data with Unknown Initial Speed

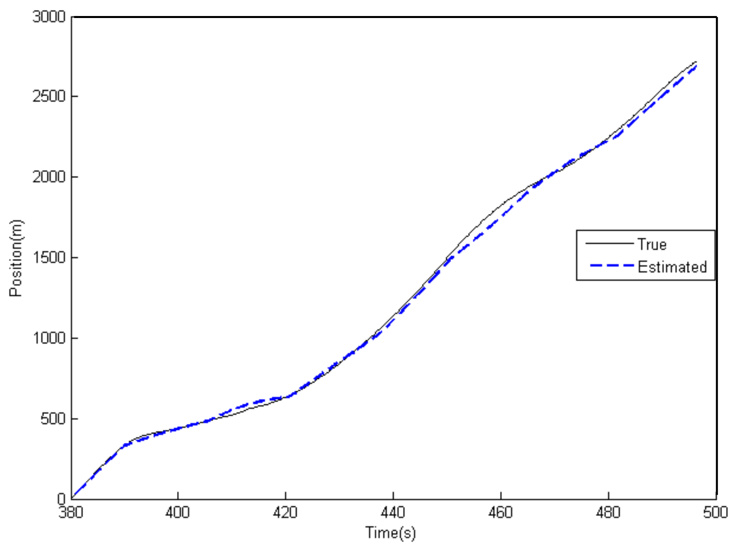


Figure 49: Vehicle Trajectory Estimation Using Airborne Data with known Initial Speed

In Figure 49 it is further assumed that the initial and end instantaneous speed is known for each local optimization problem. It depicts that the estimated trajectory almost overlaps the true trajectory. There are still areas where the estimated line is deviated from the true trajectory. Unfortunately, these parts are hard to eliminate because of the limited measurements and unknown influence of microscopic behaviors. This is another example of the observability problem. Generally, this model can perform well in most cases.

Discussion

This mesoscopic vehicle trajectory estimator shows satisfactory performance in the experimental analysis. It takes advantages of airborne data collected by monitoring helicopters and adopts a least-squares optimization approach. There are several things about this estimator that deserve further discussions. First, this experimental analysis was conducted only on a single case which does not have the statistically persuasive evidence. Further research is going to be carried out on a set of vehicles. Second, future research may emphasize on incorporating a cell transmission model even in the mesoscopic level. Third, the observability problem leads us to think how many measurements are enough. This may be a part of the helicopter speed optimization problem.

Observation Logistics for Airborne Vehicles

A separate research initiative within the scope of this program dealt with logistics of the airborne vehicles (AV), such as helicopters, airplanes or unmanned aerial vehicles (UAVs), operating cameras to remotely monitor traffic. In particular, logistics problems were defined for two cases: (1) when the fleet of AVs has the capacity to monitor all required locations to be observed and the objective is to minimize the cost of monitoring the locations; and, (2) when the fleet has more points to monitor than it is capable of monitoring, in which case the objective considered was to maximize the number of observation locations.

These problems were formulated as optimization models. Their solution complexity was evaluated and heuristic solution approaches were studied. A simple case study was addressed to demonstrate the framework for solving the logistics problems. A prototypal version of the software, named SIM-AIR, was implemented to simulate and visualize the routing/scheduling of a fleet of AVs for the case study network. Each of these activities is described in this section.

The general problem we focused on can be stated as follows:

We are given a fleet of AVs to monitor a set of observation points during a preferred (or pre-specified) time window. Each AV can operate for a given time horizon, say [0, T_{max}]. Once an AV gets near an observation point it monitors it for p time units and then it flies to observe another observation point. The problem is to define a schedule for each AV in the fleet such that all the observation points are observed respecting certain observation time's constraints.

We considered three different scenarios:

- Static (Off-line) scenario: in this scenario we assume all the observation points with the associated time windows are given a priori, that is, they *are known* in advance to be critical congestion-prone locations with time durations for monitoring them.
- Dynamic (On-line) scenario: in this scenario we assume such information is *not known* in advance but during the operating time of the AVs, congested locations that require monitoring become known dynamically;
- Mixed scenario: in this scenario we assume there is a predefined set of observation points with the associated time windows that are known a priori, but also during AVs' operations, some additional observation locations become critical and require monitoring.

So far we have addressed only the first scenario described above. Two optimization problems are defined (Problem 1 and Problem 2) below for the first scenario (off-line).

Optimization Problems, Complexities and Mathematical Models

To better address the problem, we need to introduce some definitions and notation. We assume that each observation point is represented by a location in the space and an associated time window during which the observation needs to be conducted. Hence we can represent an observation location as the triple (O_i, [a_i, b_i], p_i) where:

- O_i represents a physical link segment, say location *i*, in the network;
- [a_i, b_i] denotes the given earliest and latest times during which the location O_i can be monitored;
- p_i represents the required duration for monitoring location O_i.

We define an *observation schedule* of length *k* for each AV to be the sequence {(t_{i1}, O_{i1}), (t_{i2},O_{i2}), (t_{i3}, O_{i3}),…, (t_{ik}, O_{ik})} with $t_{i1} \leq t_{i2} \leq t_{i3} \leq \ldots \leq t_{ik.}$, where t_i *is* the time epoch (decision variable) for start of monitoring location O_i. For example, Figure 1 depicts a simple scenario where there are 6 observation locations to be observed (the red segments in the map). An observation schedule defines for each AV a *plan* on how to monitor the observation locations.

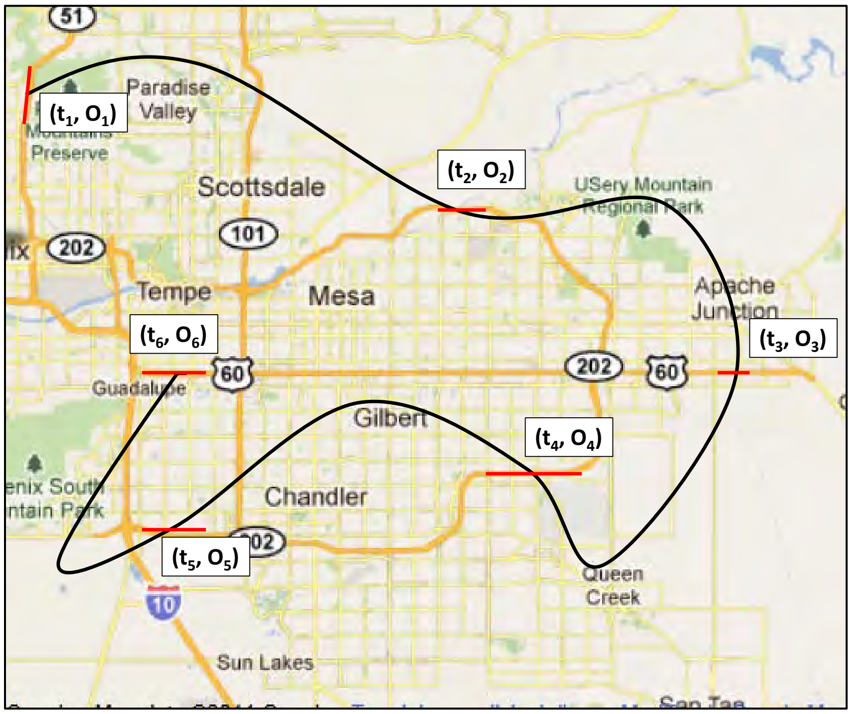


Figure 50: An Observation Schedule for Monitoring 6 Locations Corresponding to 6 Link Segments in the Network

Figure 50 represents the plan corresponding to the observation schedule {(t_1, O_1), (t_2,O_2), (t_3, O_3), (t_4, O_4), (t_5, O_5), (t_6, O_6)}. Obviously, such an observation schedule to be ‘effective’ should satisfy some intuitive characteristics. First, given an observation schedule {(t_{i1}, O_{i1}),

(t_{i2},O_{i2}), (t_{i3}, O_{i3}),…, (t_{ik}, O_{ik})}, if an AV monitors two different observation locations, say O_i and O_j, then the associated monitoring activities carried out by the AV, (that is, observe location O_i and then go from O_i to O_j and then observe O_j) should be feasible as far as the given time windows are concerned. This is ensured by relation (1) described below. Let t_i and t_j be two subsequent time observations ($t_j \geq t_i$) carried out by the same AV on locations O_i and O_j, respectively. We define these observations to be *admissible* if:

$$t_i + p_i + d_{ij}(t_i + p_i) \leq t_j \tag{36}$$

where $d_{ij}(t_i + p_i)$ is the time needed by the AV to go from location O_i to location O_j, departing from O_i at time $t_i + p_i$. In the sequel, for clarity of exposition, we consider such a distance to be independent from the departure time at point O_i, that is $d_{ij}(t)= d_{ij}$ for each t.

We define a *feasible* observation schedule of length k for an AV i in the fleet to be an observation schedule {(t_{i1}, O_{i1}), (t_{i2},O_{i2}), (t_{i3}, O_{i3}),…, (t_{ik}, O_{ik})} such that the observations are admissible and total operating time of each AV is not greater than the given operating time T_{max}, that is:

$$t_{ik}+ p_{ik} \leq T_{max} \text{ for each } i \tag{37}$$

Finally, if time window [a_i, b_i] for the location O_i has to be respected, the observation schedule has to satisfy the following time window constraint:

$$a_i \leq t_i \leq b_i \tag{38}$$

In this context, our aim now is to consider admissible and feasible observation schedules for each AV such that a given objective criterion is optimized and possible additional constraints are considered. In particular the following problems (and related variants) are being considered:

Problem 1: *Given a fleet of m AVs and a set of observation locations with associated time windows, define a set of admissible and feasible observation schedules, one for each AV to monitor all the given observation locations such that the total service time of all AVs is minimized.*

Problem 2: *Given a fleet of m AVs and a set of observation locations with associated time windows, define a set of admissible and feasible observation schedules one for each AV such that the total number of locations monitored is maximized.*

For Problem 2, there is an implicit assumption that the fleet of m AVs cannot observe all the given locations.

In this report, we address both these problems. For each of them we first analyze the simpler case when the fleet is composed of a single AV and we then generalize to the case with *m* AVs. We also analyze some variants of the problems by considering different objective functions and additional constraints. For each problem considered we study its computational complexity and provide a corresponding mathematical formulation.

Modeling Problem 1 which Minimizes Total Observation Cost

Problem 1 can be addressed by defining a complete oriented graph *G=(V,E)*, where the set of vertices corresponds to the observation locations and the length associated with each arc is the travel distance d_{ij} needed by an AV of the fleet to go from location O_i to the location O_j. We added a dummy vertex $v_0 = 0$ representing a depot at an airfield, that is a vertex where the AVs start and end their tour. The dummy vertex is connected with each vertex of the graph. We will refer to this graph in the sequel as the *Observation Graph*.

In case of a single vehicle (AV) then finding a solution for Problem 1 means to look for a Hamiltonian tour on *G* that satisfies the time windows at each vertex; that is a simple tour starting and ending at the dummy vertex, and visiting each vertex during its time window exactly once.

Consider for example the graph in Figure 51 corresponding to the Observation Graph with six observation locations. With each vertex *i* the time window $[a_i, b_i]$ is associated denoting the earliest and the latest time during which the vertex can be visited.

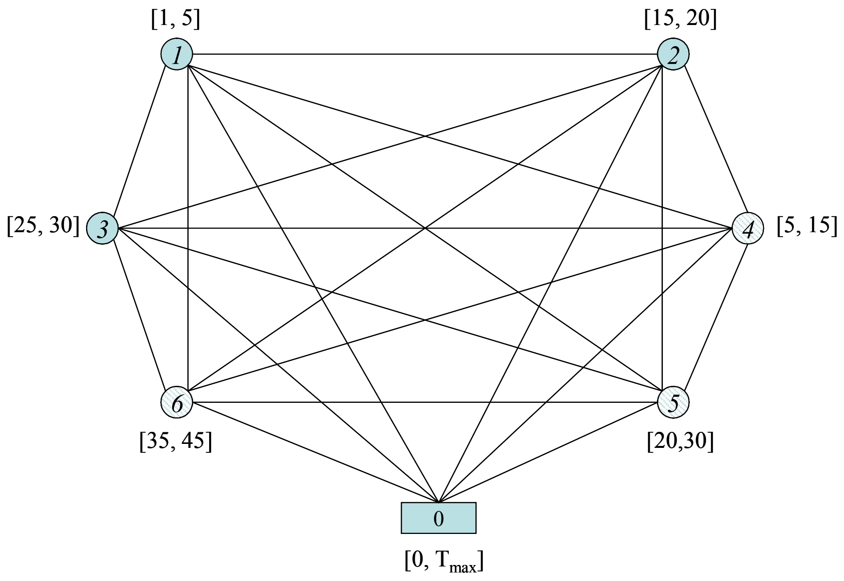


Figure 51: A Complete Graph with Six Vertices plus a Dummy Vertex Representing the Observation Graph for Six Given Observation Locations and an AV Depot

The dummy vertex has associated a time window $[0, T_{max}]$ where T_{max} is the operating time of the vehicles (AVs). Cost d_{ij} is associated with each arc (v_i, v_j); zero cost is associated with the arcs incident to the dummy vertex. A Hamiltonian tour $T=\{0,1,4,2,5,3,6,0\}$ is shown in Figure 52.

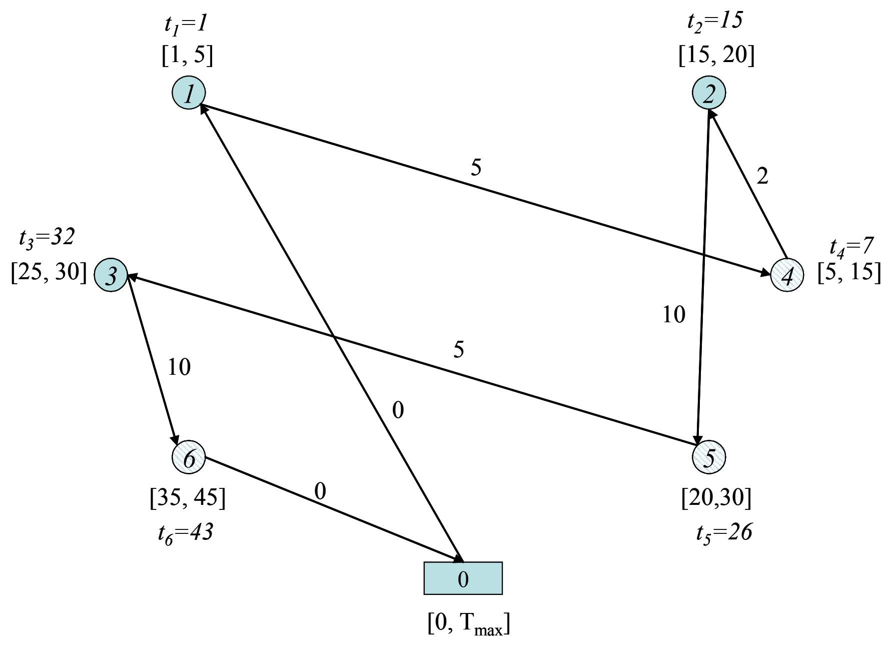


Figure 52: A Feasible Hamiltonian Tour T={0,1,4,2,5,3,6,0}

Let us assume the service (monitoring) time p_i at each vertex i is equal to 1; then by denoting with t_i the time vertex i is visited, the sequence $t_0=t_1=1, t_4=7, t_2=15, t_5=26, t_3=32, t_6=43$ is associated with T. Note that, this defines the observation sequence (t_1, O_1) - (t_4, O_4) - (t_2, O_2) - (t_5, O_5) - (t_3, O_3) -(t_6, O_6) as a solution of the corresponding Problem 1.

This sequence is such that:

- if j is visited after i then such a visit cannot be carried out before the visit at vertex i is over and j is reached, that is:

$$t_i + p_i + d_{ij} \leq t_j$$

- the visit time t_i at each vertex must respect the time window of the vertex:

$$a_i \leq t_i \leq b_i$$

If the vehicle arrives at v_i before the time window is open then is has to wait. For example, there is a waiting time at vertex 2 in the Hamiltonian tour T of Figure 3 equal to $w_2 = 7$.Note that, since the time window constraint is imposed also for the dummy vertex, the resulting tour satisfies the feasibility constraint (37).

Each Hamiltonian tour, whose associated sequence respects the time constraints (36), (37) and (38), is a *feasible* Hamiltonian tour. Each feasible Hamiltonian tour on the complete graph $G=(V,E)$ is an admissible and feasible observation schedule for Problem 1 that satisfies the time windows constraints.

Associated with each Hamiltonian tour is a routing cost corresponding to the sum of the cost of each arc in the tour. The cost of the tour *T* in Figure 52 is equal to *c(T)* = 5+2+10+5+10 = 32.

We can also associate a total service time to each Hamiltonian tour that is the difference between the ending and the starting time of the tour. Service time of tour *T* of our example is *s(T)* = $t_6 - t_1$ = 43 – 1 = 42. Obviously, the service time of a tour is the sum of the routing cost *c(T)*, the service times at vertices (i.e., p_i for each $i \in V$) and the waiting times w_i at each vertex.

Finding a feasible Hamiltonian tour on a graph *G* whose associated routing cost is minimum is the well-known *Traveling Salesman Problem with Time Windows* (TSPTW). The problem is NP-Complete and there is an extended literature that addresses and proposes different solution techniques.

When we have more than one AV in the available fleet to monitor a set of locations, then the problem becomes the Vehicle Routing Problem with Time Windows (VRPTW), where we look for a feasible set of tours (one for each AV), starting and ending at a depot, that together cover all the vertices of the graph. The objective is to minimize the total distance traveled (routing cost) by all the AVs in the fleet.

Assume there are *m* = 2 AVs to visit the vertices. Then the two admissible and feasible observations schedules on the observation graph of Figure 52, are shown in Figure 53.

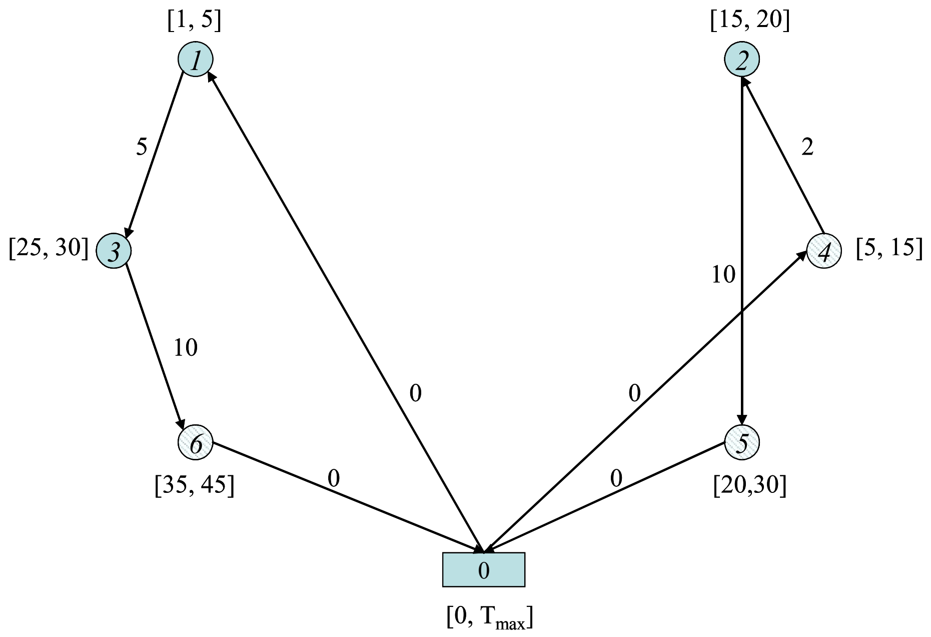


Figure 53: A feasible solution to the 2-AV Vehicle Routing Problem with Time Windows

There are two tours one for each AV. The first tour is $T_1 = \{0,1,3,6\}$, with the associated sequence $t_1 = 1$, $t_3 = 25$, $t_6 = 35$, routing cost $c(T_1) = 15$, and total service time $s(T_1) = 34$. The second tour T_2 has the associated sequence $t_4 = 5$, $t_2 = 15$, $t_5 = 25$, routing cost $c(T_2) = 12$, and total service time $s(T_2) = 20$. The two tours define the two observation sequences: (t_1, O_1) - (t_3, O_3) - (t_6, O_6) for the first AV and (t_2, O_2) - (t_4, O_4) -(t_5, O_5) for the second AV. Note that we can obtain for each set of tours the following costs:

- the *total routing cost* that is equal to $c(T_1) + c(T_2)=27$;
- the *total service time* that is equal to $s(T_1) + s(T_2)=64$;
- the *maximum service time* that is equal to $max\{s(T_1) + s(T_2)\}=34$.

The VRPTW is NP-complete. We can think of other variants of the problem, some of which are given below:

Problem 1b (Lateness): Suppose to have soft time windows, that is, if an AV arrives at the time window $[a_i, b_i]$ at a time $t_i \geq b_i$ (that is, it arrives late) a penalty term can be included in the objective function. Then one could be interested in finding a Hamiltonian tour such that total lateness and service time are minimized. Another objective could be to minimize the maximum lateness.

Problem 1c (Earliness): Suppose the AV could arrive early at the observation point, that is $t_i \leq a_i$ and therefore it waits before resuming monitoring operations. One may then be interested in finding a Hamiltonian tour such that the total idle time and service is minimized.

Problem 1d (Fleet Size): When the problem does not have a single Hamiltonian tour as a feasible solution, one may be interested in looking for a fleet of AVs of minimum size such that all the locations are observed exactly once, satisfying the time windows constraints (38) and the T_{max} constraints (37).

For each of the above problems, we develop now a corresponding optimization formulation.

Formulations for AV Routing to Minimize Total Observation Cost

Let binary variables x_{ij} associated with each arc of the observation graph G assume value equal to 1 if the corresponding arc (i,j) is selected and 0 otherwise. Define a set of variables $t_i \geq 0$ associated with each vertex of the graph, to denote the time epoch vertex i is visited. Finally, let variable s denote the total service time of a tour.

We first provide the mathematical formulations of Problem 1 by considering the case of a single AV (with the corresponding variants Problem 1b and Problem 1c), the provided formulations are easily generalized to the case of a multiple fleet of m AVs. We then provide the formulation for the last variant of Problem 1, that is, Problem 1d.

Formulations for Problems 1, 1b, and 1c for the single AV case

The mathematical formulation for Problem 1 in case of a single AV is the following.

Problem 1 – single AV

$$\min s \qquad (2.1)$$

$$\sum_{j \epsilon V} x_{ij} = 1 \qquad \forall\, i \in V \qquad (2.2)$$

$$\sum_{j \epsilon V} x_{ji} = 1 \qquad \forall\, i \in V \qquad (2.3)$$

$$t_i + p_i + d_{ij} \le t_j + B\left(1 - x_{ij}\right) \qquad \forall\, (i,j) \in E, j \neq 0 \qquad (2.4)$$

$$t_i \ge a_i \qquad \forall\, i \in V \qquad (2.5)$$

$$t_i \le b_i \qquad \forall\, i \in V \qquad (2.6)$$

$$t_0 = 0$$

$$t_i + p_i \le s \qquad \forall\, i \in V \qquad (2.7)$$

$$x_{ij} \in \{0,1\} \qquad \forall\, (i,j) \in E \qquad (2.8)$$

$$t_i \ge 0 \qquad \forall\, i \in V \qquad (2.9)$$

where the objective function (2.1), together with constraints (2.7), minimizes the service time of the tour. Constraints (2.2) and (2.3) require the tour to visit each vertex exactly once. Constraints (2.4)-(2.5)-(2.6) are the time-window constraints. In particular, constraints (2.4) requires that if the arc *(i,j)* is used (i.e., x_{ij}=1), then the time of visit at vertex *j* must satisfy rule (36), and *B* is a large constant value. Constraints (2.5) and (2.6) model the time windows constraints. Binary constraints on variables x_{ij} are imposed by (2.8) and non negativity of variables t_i by (2.9).

To formulate the variant of the problem corresponding to soft time windows that allow lateness, a new set of non negative variables $l_i \ge 0$ defining the lateness at vertex *i* is needed. Let *C* be a penalty cost for a unit time of lateness, then following is the formulation of Problem 1b:

Problem 1b (Lateness) – single AV

$$\min s + C\sum_{i \epsilon V} l_i \tag{2.1b}$$

(2.2)-(2.3)-(2.4)-(2.5)-(2.7)-(2.8)-(2.9)

$$t_i \leq b_i + l_i \qquad \forall\, i \in V \tag{2.6b}$$

$$l_i \geq 0 \qquad \forall\, i \in V \tag{2.10}$$

where the new objective function (2.1b) takes into account the lateness penalty and constraints (2.6) are replaced by the new constraints (2.6b) that allow lateness.

To formulate the variant of the problem with soft time windows allowing for earliness, a set of non negative variables $w_i \geq 0$ defining the waiting time at vertex i is needed. Let C be a penalty cost for unit time of waiting, then following is the formulation of Problem 1c:

Problem 1c (Earliness) – single AV

$$\min s + C\sum_{i \epsilon V} w_i \tag{2.1c}$$

(2.2)-(2.3)-(2.4)-(2.6)-(2.7)-(2.8)-(2.9)

$$t_i \geq a_i - w_i \qquad \forall\, i \in V \tag{2.5b}$$

$$w_i \geq 0 \qquad \forall\, i \in V \tag{2.11}$$

where the new objective function (2.1c) takes into account the penalty for the waiting time and constraints (2.5c) allow for early arrival at an observation point.

Mathematical Formulation of Problem 1d

Let us now consider Problem 1d, that is the case where one is interested in minimizing the number of AVs in the fleet. Let us define the set $M=\{1,2,\ldots,n\}$, where $n=|V|$is the maximum possible number of AVs that may be needed. Then the problem formulates as:

Problem 1d – Minimizing the Total Number of AVs

$$\min \sum_{k \in M} \sum_{j \in V} x_{oj}^k \tag{2.12}$$

$$\sum_{k \in M} \sum_{j \in V} x_{ij}^k = 1 \qquad \forall\, i \in V \backslash \{0\} \tag{2.13}$$

$$\sum_{k \in M} \sum_{j \in V} x_{ji}^k = 1 \qquad \forall\, i \in V \backslash \{0\} \tag{2.14}$$

$$\sum_{k \in M} \sum_{j \in V} x_{j0}^k - \sum_{k \in M} \sum_{j \in V} x_{0j}^k = 0 \tag{2.15}$$

$$t_i^k + p_i + d_{ij} \le t_j^k + B\left(1 - x_{ij}^k\right) \qquad \forall\, (i,j) \in E, j \neq 0, \forall\, k \in M \tag{2.16}$$

$$t_i^k \ge a_i - B(1 - \sum_{j \in V} x_{ij}^k) \qquad \forall\, i \in V, \forall\, k \in M \tag{2.17}$$

$$t_i^k \le b_i - B(1 - \sum_{j \in V} x_{ij}^k) \qquad \forall\, i \in V, \forall\, k \in M \tag{2.18}$$

$$t_0^k = 0 \qquad \forall\, k \in M \tag{2.19}$$

$$t_i^k + p_i \le T_{max} \qquad \forall\, i \in V, \forall\, k \in M \tag{2.20}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall\, (i,j) \in E, \forall\, k \in M \tag{2.21}$$

$$t_i^k \ge 0 \qquad \forall\, i \in V, \forall\, k \in M \tag{2.22}$$

Objective function (2.12) requires the minimization of the number of AVs that start their tours at the dummy vertex, that is, the total number of AVs that would be needed out of the maximum *n* AVs. Moreover, constraints (2.7) has been replaced by constraints (2.20) that force the service time of each AV to be less than or equal to its maximum operating time T_{max}.

Obviously, also for this variant we could consider the case of soft time windows using the same modifications as previously described.

Modeling Problem 2 which Maximizes Number of Locations Observed

Now we address the following problem:

Given a fleet of m AVs and a set of observation locations with associated time windows, define a set of feasible and admissible observation schedules one for each AV such that the total number of observations locations monitored is maximized

Consider the observation graph described in the previous section. The number of vertices visited by the tour corresponds to the total number of observation locations that is monitored. Hence, the longer the tour (in terms of number of vertices visited) the more the observations made by the AV. Therefore the objective function for Problem 2 is the maximization of the length of the route (in terms of vertices visited). However, not all the tours define an observation schedule for Problem 2. Indeed, they need to satisfy the *admissibility, feasibility* and time windows constraints (36), (37) and (38) for an observation sequence.

Hence, Problem 2, in case of a single AV, can be solved by solving the *Constrained Longest Tour* (CLT) on the observation graph *G* where the added constraints ensure (36), (37) and (38) to be satisfied.

If there are *m*>1 AVs in the fleet then Problem 2 becomes the *Constrained Longest Multi-Tour* (CLMT) on the graph, where the problem consists of finding *m* disjoint simple tours covering together all the vertices of the graph and satisfying additional time-window constraints. Both the problems are NP-complete, since they are special cases of the Hamiltonian tour problem and the Vehicle Routing Problem, respectively. Next section provides the corresponding optimization formulations.

Formulations for AV Routing to Maximize Number of Observations

Formulation for Problem 2 for the single AV case

Let *G=(V,E)* be a complete oriented graph, with costs d_{ij} associated with each arc *(i,j)*. Let binary variables x_{ij} associated with each arc of the graph be equal to 1 if the corresponding arc is selected and 0 otherwise, and let variables t_i associated with each vertex of the graph denote the observation epoch for the vertex.

The mathematical formulation of the corresponding Constrained Longest Path Problem on G is the following.

$$\max \sum_{(i,j)\in E} x_{ij} \tag{2.23}$$

$$\sum_{j\epsilon V} x_{0j} = 1 \tag{2.24}$$

$$\sum_{j\epsilon V} x_{j0} = 1 \tag{2.25}$$

$$\sum_{j\epsilon V} x_{ij} \leq 1 \qquad \forall\, i \in V\backslash\{0\} \tag{2.26}$$

$$\sum_{j\epsilon V} x_{ji} \leq 1 \qquad \forall\, i \in V\backslash\{0\} \tag{2.27}$$

$$\sum_{j\epsilon V} x_{ij} - \sum_{j\epsilon V} x_{ji} = 0 \qquad \forall\, i \in V\backslash\{0\} \tag{2.28}$$

$$t_i + p_i + d_{ij} \leq t_j + B(1 - x_{ij}) \qquad \forall\, (i,j) \in E, j \neq 0 \tag{2.29}$$

$$t_i \geq a_i \qquad \forall\, i \in V \tag{2.30}$$

$t_i \leq b_i \qquad \forall\, i \in V \qquad (2.31)$

$t_0 = 0$

$t_i + p_i \leq T_{max} \qquad \forall\, i \in V \qquad (2.32)$

$x_{ij} \in \{0,1\} \qquad \forall\, (i,j) \in E \qquad (2.33)$

$t_i \geq 0 \qquad \forall\, i \in V \qquad (2.34)$

Objective function (2.23) maximizes the number of selected arcs and therefore the length of the tour. Constraints (2.24) and (2.25) impose the tour to start and end at the dummy vertex 0 of the graph. Each node cannot be visited more than once is ensured by constraints (2.26)-(2.27)-(2.28). The admissibility, feasibility and time windows constraints are ensured by constraints (2.29) – (2.32). Binary and non negativity constraints are represented by (2.33) and (2.34).

Formulation of Problem 2 for the multiple AVs case

Let *G=(V,E)* be an oriented graph, with costs d_{ij} associated with each arc *(i,j)*. Let *k∈M={1,2,…,m}* be the index set of AVs to be routed. Let binary variables x_{ij}^k associated with each arc of the graph and each vehicle *k* be equal to 1 if the corresponding arc is traversed by vehicle *k* and 0 otherwise, and let the variables t_i^k associated with each vertex *i* of the graph and each vehicle *k* denote the observation epoch of the vertex by vehicle *k*.

The mathematical formulation of the Multi–Vehicle Constrained Longest Tour on G is the following.

$$\max \sum_{(i,j)\in E} x_{ij}^k \qquad (2.35)$$

$$\sum_{k\in M} x_{ij}^k = 1 \qquad \forall\, (i,j) \in E \qquad (2.36)$$

$$\sum_{k\in M}\sum_{j\epsilon V} x_{0j}^k = 1 \qquad (2.37)$$

$$\sum_{k\in M}\sum_{j\epsilon V} x_{j0}^k = 1 \qquad (2.38)$$

$$\sum_{k\in M}\sum_{j\epsilon V} x_{ij}^k \leq 1 \qquad \forall\, i \in V\backslash\{0\} \qquad (2.38)$$

$$\sum_{k\in M}\sum_{j\epsilon V} x_{ji}^k \leq 1 \qquad \forall\, i \in V\backslash\{0\} \qquad (2.40)$$

$$\sum_{k\in M}\sum_{j\epsilon V} x_{ij}^k - \sum_{h\in M}\sum_{j\epsilon V} x_{ji}^k = 0 \qquad \forall\, i \in V\backslash\{0\} \qquad (2.41)$$

$$t_i^k + p_i + d_{ij} \leq t_j^k + B\left(1 - x_{ij}^k\right) \qquad \forall\, (i,j) \in E, j \neq 0, \forall\, k \in M \qquad (2.42)$$

$$t_i^k \geq a_i \qquad \forall\, i \in V, \forall\, k \in M \qquad (2.43)$$

$$t_i^k \leq b_i \qquad \forall\, i \in V, \forall\, k \in M \qquad (2.44)$$

$$t_0^k = 0 \qquad \forall\, k \in M \qquad (2.45)$$

$$t_i^k + p_i \leq T_{max} \qquad \forall\, i \in V, \forall\, k \in M \qquad (2.46)$$

$$x_{ij}^k \in \{0,1\} \qquad \forall\, (i,j) \in E, \forall\, k \in M \qquad (2.47)$$

$$t_i^k \geq 0 \qquad \forall\, i \in V, \forall\, k \in M \qquad (2.48)$$

Simulation and Visualization of Routing/Scheduling AVs

Test Scenario

In order to test a method we needed to develop test problems. One test problem was developed from a simulation of traffic loaded on a real network. The research team was fortunate to have a small network preliminarily developed by team member Yi-Chang Chiu and simulated using DynusT [30], a traffic simulation software developed at the University of Arizona. The network was for traffic flow for Beaverton, Oregon. (The network data needed to be cleaned a little since the visualization showed some redundant links and also unconnected components. The initial network contained 748 nodes and 1682 links; the cleaned network had 654 nodes and 1421 links). DynusT includes a software code NEXTA (Network EXplorer for Traffic Analysis) which is graphical user interface to facilitate the preparation, post-processing, and analysis of traffic simulation results. Using NEXTA it is possible to change the settings of a simulation run (traffic lights phases, characteristics of the network, OD matrices etc.). Figure 54 shows the Beaverton network visualized on Google Earth.

After the traffic simulation was run (recall that we are currently addressing the off-line scheduling and routing), traffic congestion times and locations were obtained, where congestion was defined for each given link segment *a,* the traffic density exceeded a given density of ρ_a vehicles/mile and speed (mph) was less than some given function of ρ_a that described link segment capacity. These defined the required observation segments, both in space and time. By decreasing ρ_a we had more observation segments; increasing ρ_a decreased this number. In summary we performed these steps for our off-line scheduling/routing.

1. Simulate the traffic behavior on the given traffic network.
2. Analyze the output data at the end of the simulated period.

3. Select the “congested” links according to the speed and density thresholds.
4. Schedule the AVs to monitor the congested links.
5. Visualize the network on Google Earth.
6. Simulate and visualize the monitoring activity of each AV on Google Earth.

We refer to the system for scheduling, routing, and visualization of AVs (steps 4-6) as SIM-AIR.

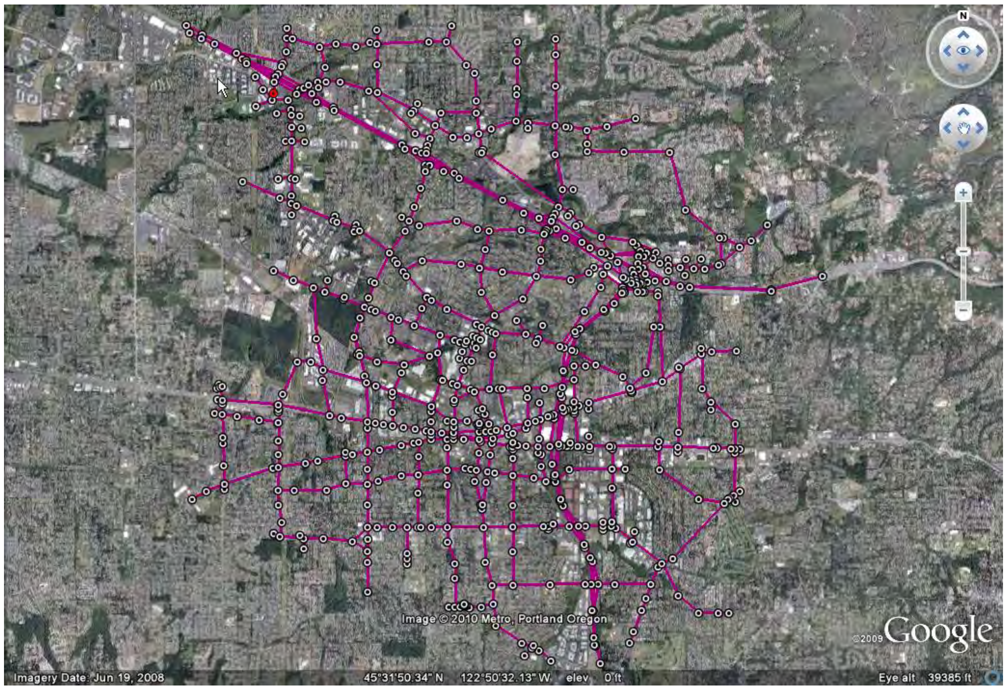


Image © 2010 Metro, Portland Oregon
©2009 Google
Imagery Date: Jun 19, 2008
45°31'50.34" N 122°50'32.13" W elev 0 ft
Eye alt 39385 ft

Figure 54: The Beaverton network visualized on Google Earth.

We ran DynusT on the network considering a time horizon of 24 periods and 107 OD pairs. The input files *network.dat* and *xy.dat* describe the characteristics of the network. The output files *OUTLinkDent.dat*, *OUTLinkSpeedALL.dat* and *OUTLinkVeh.dat* were the output files of DynusT used by SIM-AIR. The description and contents of the two input files are given in Appendices 1 and 3.

Description of SIM-AIR

SIM-AIR reads the problem scenario which includes the network data, the output of a traffic simulator and the corresponding observation segments. The scheduling code then develops the routes and schedules of the AVs in the fleet. SIM-AIR then visualizes the routes and simulates the monitoring activity of each AV on Google Earth.

For simulating real-time or on-line scheduling and routing (a future task), a vehicle simulator and SIM-AIR would need to be integrated to perform the following steps. In this case, traffic control decisions need to be made based on the data gathered by the AVs:

Simulation of on-line monitoring and controlling using AVs:

1. Simulate the traffic behavior of given network for every Δ minutes.
2. Analyze the output data every Δ minutes.
3. Select current "congested" links according to speed and density thresholds.
4. Schedule the AVs to monitor the congested links.
5. Monitor the link and , if necessary, control traffic (through traffic lights phasing, VMS, etc.)
6. Repeat steps 1-5 until end of simulation period for the test scenario.
7. Simulate and visualize the monitoring activity of each AV on Google Earth.

Routing and Scheduling Algorithms

As discussed, the optimization problems for routing and scheduling are computationally hard, but there are heuristics available and new heuristics can be developed. For the purpose of demonstrating the "proof of concept" we simply designed a heuristic that selects the link segments on a "first-identified-first observed" manner, that is, as each observation segment is identified in time it is put on list of points that are observed on a first-in first out basis. In this demonstration a fleet of only one AV was assumed. In the future, heuristic codes will be developed for other cases (multiple AVs, and for variants of Problem 1 and Problem 2).

Simulating and Visualizing AV Movement on Google Earth

The route was visualized on Google earth. When the AV (e.g., helicopter) moves from one link segment to another, the trajectory is red when the helicopter is performing a monitoring activity, while the trajectory is blue when the AV is deadheading to the next link segment to be observed.

Figures 55 and 56 show images from the SIM-AIR simulation. Figure 55 shows the helicopter while it is performing a monitoring activity on a congested link. Figure 56 shows the helicopter flying to the beginning of the next link that needs to be monitored.

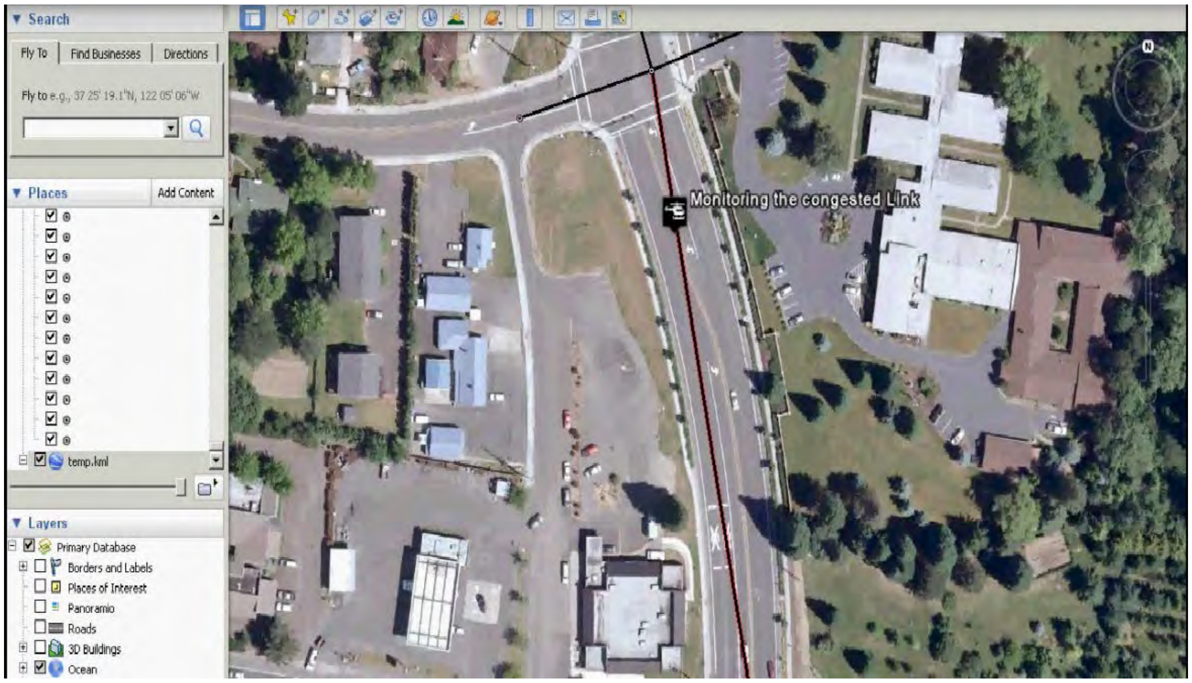


Figure 55: Simulation of the AV (helicopter) while it is performing a monitoring activity on a congested link.

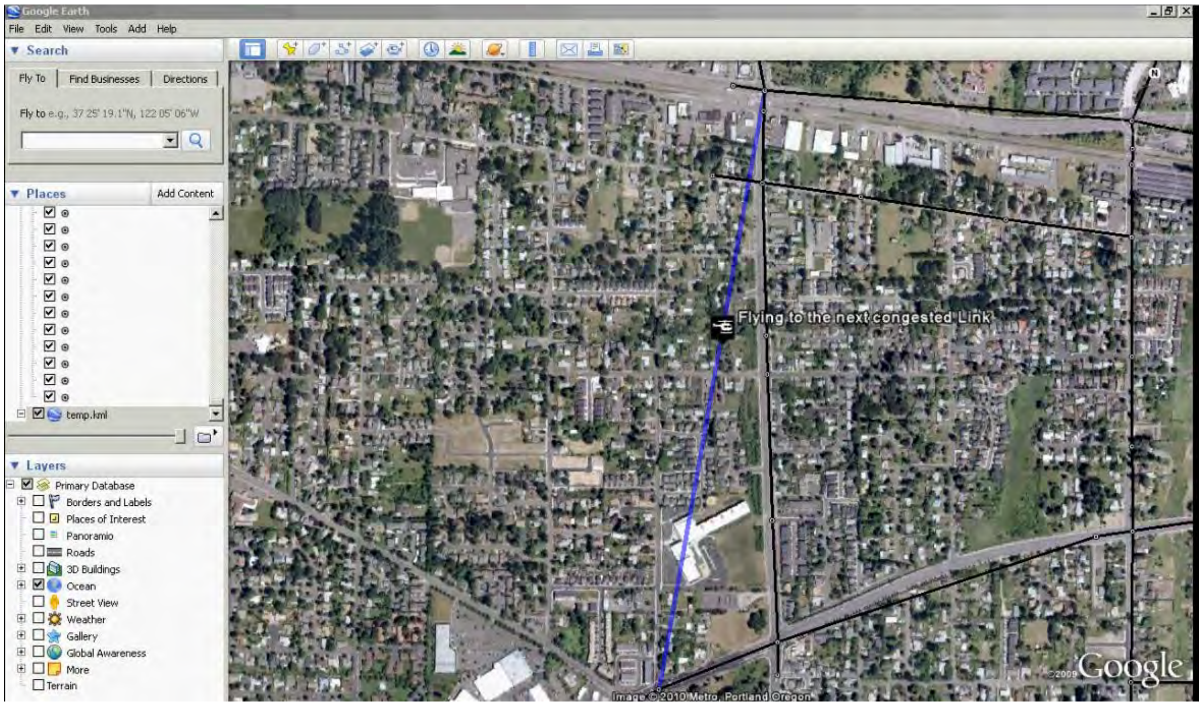


Figure 56: Simulation of the AV (helicopter) while it is flying to the next congested link.

Potential Future Applications

The project has also sought to solve practical problems for the industry, in the sense that these products in turn could feed future applications. This has been achieved through feedback from the Technical Advisory Panel for the project, which was sought both through a kickoff meeting with the panel (held in April 2010) and in soliciting feedback from panel members (electronically) on the value of airborne data products (conducted in summer 2011). The feedback from the technical advisory group was that the airborne traffic data would have value, inasmuch as we are able to capture extended sequences (5-10 min) over reasonable lengths of roadway (at least 0.5-mile) in the imagery field of view. From the view of our technical advisory panel, such imagery allows for more sufficient traffic data to be of use for: (1) microscopic traffic simulation modeling; (2) for understanding patterns of traffic congestion over larger geographic areas where aircraft may be able to travel quickly; and (3) estimating traffic states (levels of congestion) on specific facilities, when integrated with other ground-collected traffic data.

Bibliography

[1] Angel, A., M. Hickman, P.B. Mirchandani, and D. Chandnani, "Methods for Analyzing Traffic Imagery Collected from Aerial Platforms," *IEEE Trans. Intelligent Transportation Systems*, Vol. 4, No. 2, 2003, pp. 99-107.

[2] Agrawal, A., and M. Hickman, "Automated Extraction of Queue Lengths from Airborne Imagery." Proc. IEEE Intelligent Transportation Systems Conference, October 4-7, 2004.

[3] Hickman, M., and P.B. Mirchandani, "Uses of Airborne Imagery for Microscopic Traffic Analysis," *ASCE Conference on Applications of Advanced Technology in Transportation,* August 2006.

[4] Angel, A., and M. Hickman, "Experimental Investigation of Travel Time Estimation Using Geo-referenced Aerial Video," 81st Annual Meeting of the Transportation Research Board, 2002.

[5] Angel, A., and M. Hickman, "A Method for Analyzing the Performance of Signalized Intersections from Airborne Imagery," 82nd Annual Meeting of the Transportation Research Board, 2003.

[6] Jiang, Z., M. McCord, and P. Goel, "Improved AADT Estimation by Combining Information in Image- and Ground-based Traffic Data," *ASCE Journal of Transportation Engineering*, July 2006.

[7] McCord, M., Y. Yang, Z. Jiang, B. Coifman, and P. Goel, "Estimating AADT from Satellite Imagery and Air Photos: Empirical Results," *Transportation Research Record*, No. 1855, pp. 136-142, 2003.

[8] Coifman, B., Yang, Y., "Estimating Spatial Measures of Roadway Network Usage from Remotely Sensed Data" *Transportation Research Record 1870*, 2004, pp. 133-138.

[9] Coifman, B., M. McCord, R.G. Mishalani, M. Iswalt, and Y. Ji, "Roadway Traffic Monitoring from and Unmanned Aerial Vehicle," *IEE Proc. of Intelligent Transport Systems,* Vol. 153, Issue 1, 2006, pp. 11-20.

[10] McCord, M.R., R.G. Mishalani, B. Coifman, Y. Ji, and M. Iswalt, "Determining Origin-Destination Flows across a Two-intersection Network from Non-overlapping UAV and Ground-based Imagery," *ASCE Conference on Applications of Advanced Technology in Transportation,* August 2006.

[11] Grejner-Brzezinska, D., C. Toth, S. Moafipoor, E. Paska, and N. Csanyi, "Vehicle Classification and Traffic Flow Estimation from Airborne LiDAR/CCD Data," *IAG Symposia, Monitoring and Understanding a Dynamic Planet with Geodetic and Oceanographic Tools*, Springer Berlin-Heidelberg, 2005.

[12] Goel, P., M. McCord, S. Ruan, and M. O'Kelly, "Bayesian Estimation of Statewide OD Flows using Link Volumes Estimated from Combined Information in Remotely Sensed

Data and Ground Counts," ASCE Conference, *Applications of Advanced Technology in Transportation,* August 2006.

[13] Angel, A., and M. Hickman, *Considerations for Aerial Traffic Data Collection*, NCRST-F Cookbook #2, 2002.

[14] Angel, A., and M. Hickman, *A Method for Measuring Freeway Level of Service from Airborne Imagery*, NCRST-F Cookbook #3, 2002.

[15] Angel, A., and M. Hickman, *A Methodology for Measuring the Level of Service for Urban Streets from Airborne Imagery*, NCRST-F Cookbook #4, 2002.

[16] Angel, A., and M. Hickman, *A Method for Measuring Level of Service for Signalized Intersections from Airborne Imagery*, NCRST-F Cookbook #5, 2002.

[17] Kühne, R., M. Ruhé, and E. Hipp. "A Model for New Data - Using Airborne Traffic Flow Measurement for Traffic Forecast," Paper presented at TRISTAN VI, Phuket, Thailand, June 2007.

[18] Ernst, S., S. Zuev, K. Thiessenhusen, M. Hetscher, S. Rassmann, and M. Ruhé." LUMOS - Airborne Traffic Monitoring System," *Proceedings of the IEEE 6th International Conference on Intelligent Transportation Systems*, Shanghai, China, 2003.

[19] Hipp, E. *Ansätze zur Modellierung einer Kurzzeitprognose auf Basis Luftgestützter Verkehrsdaten* (in German). MS Thesis, University of Stuttgart, Stuttgart, Germany, 2006.

[20] Hoogendoorn, S.P., H.J. van Zuylen, M. Schreuder, B. Gorte, and G. Vosselman. "Microscopic Traffic Data Collection by Remote Sensing," Paper presented at the 82nd Annual Meeting of the Transportation Research Board, January 2003.

[21] Hoogendoorn, S.P., and M. Schreuder. "Tracing Congestion Dynamics with Remote Sensing," Paper presented at the 84th Annual Meeting of the Transportation Research Board, January 2005.

[22] M. Schreuder, S.P. Hoogendoorn, H.J. Van Zuylen, B. Gorte, and G. Vosselman (2003). "Traffic Data Collection from Aerial Imagery," *Proceedings of the IEEE Intelligent Transportation Systems Conference*, Vol.1, pp. 779-784.

[23] Nejadasl, F.K., B.G.H. Gorte, and S.P. Hoogendoorn. "Robust Vehicle Tracking in Video Images being taken from a Helicopter," Paper presented at the ISPRS Commission VII Mid-term Symposium, *Remote Sensing: From Pixels to Processes*, Enschede, the Netherlands, May 2006.

[24] Coifman, B., M. McCord, M. Mishalani, and K. Redmill. "Surface Transportation Surveillance from Unmanned Aerial Vehicles," Paper presented at the 83rd Annual Meeting of the Transportation Research Board, January 2004.

[25] Coifman, B., M. McCord, M. Mishalani, M. Iswalt, and Y. Ji. “Roadway Traffic Monitoring from an Unmanned Aerial Vehicle,” *IEE Proceedings on Intelligent Transportation Systems*, Vol. 153, No. 1, March 2006.

[26] Kadam, K. *Detection and Tracking of Vehicles in an Airborne Video Sequence*, MS Thesis, Department of Electrical and Computer Engineering, University of Arizona, May 2005.

[27] Gentili, M., and P.B. Mirchandani, “Location Of Active Sensors On A Traffic Network,” *Annals of Operations Research*, January 2005, pp. 229-257.

[28] Mirchandani, P.B., S. Nobe, and W. Wu, “The Use of On-Line Turning Proportion Estimation in Real-Time Traffic-Adaptive Signal Control,” *Transportation Research Record* 1748, 2001, pp. 80-86.

[29] Chiu, Y.-C., and P.B. Mirchandani, “Online Behavior-Robust Feedback Information Routing Strategy for Mass Evacuation,” *IEEE Transactions on Intelligent Transportation Systems*, Vol. 9, No. 3, June 2008, pp. 264-274.

[30] DynusT User’s Manual: http://DynusT.net/wikibin/doku.php.

[31] Melo, J., A. Naftel, A. Bernardino, and J. Santos-Victor, “Detection and Classification of Highway Lanes Using Vehicle Motion Trajectories,” *IEEE Transactions on Intelligent Transportation Systems*, Vol. 7, No. 2, 2006, pp. 188-200.

[32] Hinz, S. (2005). Detection of vehicles and vehicle queues for road monitoring using high resolution aerial images. In: Proc. 9th World Multiconf. Systemics, Cybern. Informatics, Orlando, Florida.

[33] Moon, H., R. Chellappa, and A. Rosenfeld (2002). Performance Analysis of a Simple Vehicle Detection Algorithm. Image and Vision Computing, 20(1), 249-253.

[34] Kim, Z. and J. Malik (2003). Fast vehicle detection with probabilistic feature grouping and its application to vehicle tracking. In: IEEE Int. Conf. Comp. Vision, 524–531.

[35] Zhao, T. and R. Nevatia (2003): Car Detection in Low Resolution Aerial Images. Image and Vision Computing 21, 693-703.

[36] Nguyen, T. T., H. Grabner, H. Bischof, and B. Gruber (2007). On-line boosting for car detection from aerial images. RIVF’07, IEEE, 87–95.

[37] Tuermer, S., J. Leitloff, P. Reinartz, and U. Stilla, (2010). Automatic vehicle detection in aerial image sequences of urban areas using 3D HoG features. PCV 2010 - Photogrammetric Computer Vision and Image Analysis.

[38] Reinartz P., T. Krauss, M. Pötzsch, H. Runge, and S. Zuev (2005). Traffic Monitoring with Serial Images from Airborne Cameras. Proc. of Workshop on High-Resolution Earth Imaging for Geospatial Information, Hannover.

[39] Hinz, S., D. Lenhart, and J. Leitloff (2007). Detection and tracking of vehicles in low framerate aerial image sequences. In Proc. Workshop on High-Resolution Earth Imaging for

Geo-Spatial Information, CD, Hannover, Germany.

[40] Nejadasl, F.K., B. Gorte, and S. Hoogendoorn (2006), Optical flow based vehicle tracking strengthened by statistical decisions, ISPRS Journal of Photogrammetry and Remote Sensing, 61(3-4), 159-169.

[41] Shastry, A., "Airborne Video Registration and Traffic Flow Parameter Estimation in a Multimedia GIS," MS Thesis, Department of Electrical and Computer Engineering, University of Arizona, December 2002.

[42] Du, X. and M. Hickman (2012), Estimating a Road Mask to Improve Vehicle Detection and Tracking in Airborne Imagery, Paper accepted in final form for publication in Transportation Research Record.

[43] Lowe, D. (1999). "Object recognition from local scale-invariant features". *Proceedings of the International Conference on Computer Vision*. **2**. pp. 1150–1157.

[44] Wang,Y., M. Papageorgiou, "Real-time freeway traffic state estimation based on extended Kalman filter: a general approach," *Transportation Research Part B*, 2005, Vol. 39, No. 2, pp. 141–167.

[45] Hegyi, A., D. Girimonte, R. Babuška, B. De Schutter, "A comparison of filter configurations for freeway traffic state estimation," *In Proceedings of the IEEE ITSC Conference on Intelligent Transportation Systems* (pp. 1029–1034), 17–20 September 2006, Toronto, Canada (on CD).

[46] Mihaylova, L., R. Boel, "A particle filter for freeway traffic estimation," *In Proceedings of the 43rd IEEE Conference on Decision and Control*, 2004, pp. 2106–2111, Bahamas.

[47] Sun, X., L. Muñoz, R. Horowitz, "Highway traffic state estimation using improved mixture Kalman filters for effective ramp metering control," *In Proceedings of the 42th IEEE Conference on Decision and Control*, 2003, pp. 6333–6338, Hawaii, USA.

[48] Work, D.B., O.-P. Tossavainen, S. Blandin, A.M. Bayen, T. Iwuchukwu, K. Tracton, "An ensemble Kalman filtering approach to highway traffic estimation using GPS enabled mobile devices," *Decision and Control, 47th IEEE Conference*, 2008, pp.5062-5068.

[49] Willner, D., C. Chang, and K. Dunn, "Kalman filter configurations for multiple radar systems," *Tech Rep. TN 1876-21*, MIT Lincoln Lab, 1976.

[50] Daganzo, C., "The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory," *Transportation Research B*, 28B (4), 1994, pp. 269–287.

[51] Papageorgiou, M., J.-M. Blosseville, "Macroscopic modelling of traffic flow on the boulevard Périphérique in Paris," *Transportation Research B*, 23 (1), 1989, pp. 29–47.

[52] Welch, G and G. Bishop, "An Introduction to the Kalman Filter," 2001, *http://www.cs.unc.edu/~welch/kalman/*