APPLICATION OF AERIAL VIDEO FOR TRAFFIC FLOW MONITORING AND MANAGEMENT

Pitu Mirchandani, Professor, Department of Systems and Industrial Engineering
Mark Hickman, Assistant Professor, Department of Civil Engineering
Alejandro Angel, Graduate Researcher
Dinesh Chandnani, Graduate Researcher
ATLAS Research Center
University of Arizona
P.O. Box 210020
Tucson, AZ 85721-0020
pitu@sie.arizona.edu
mhickman@engr.arizona.edu
aangel@mmla.com
dinesh@cs.arizona.edu

ABSTRACT

Researchers at the University of Arizona are investigating the potential of remotely sensed data, using aerial video, to enhance existing data sources and therefore to improve traffic management. This research has developed new methods for both data collection and image processing of remotely sensed data. While the use of remotely sensed data to monitor traffic flows is not new, this research is examining the integration of digital video, global positioning systems (GPS), and automated image processing to improve the accuracy and cost-effectiveness of data collection and reduction. Several different aerial platforms are under investigation for the data collection. With these platforms, a number of experiments in which aerial video and GPS data were collected from a major arterial and from a freeway are described. A technique has been developed to process the GPS data and the digital imagery, in near real time, to estimate vehicle speeds directly from the video images. The algorithm is capable of detecting vehicles and measuring their speeds, using image registration and edge detection techniques. When vehicles are matched between two frames, their speeds can be estimated directly. In addition, similar techniques are being used to derive other traffic flow parameters, such as densities, travel times, delays, turning counts, queue lengths, and measures of platoon dispersion. The automated image processing significantly reduces the amount of time needed to generate these traffic data.

INTRODUCTION

Traffic flows in many large urban areas are monitored and managed using data from a variety of sensors. The traditional technologies for traffic sensing, including inductive loop detectors and video cameras, are positioned at fixed locations in the transportation network. While these detectors do provide useful information and data on the traffic flows at particular points, they generally do not provide useful data for traffic flows over space. It is not possible to move the detectors, and they are not capable of providing data on vehicle trajectories and paths through the network. It is also difficult to use these sensors to identify traffic flow characteristics (speeds, acceleration/deceleration, and routing information) of individual vehicles.

The use of aerial platforms for traffic data collection has appeal to supplement the traditional traffic sensors. Aerial photography and video has the properties of being both *mobile*, so that it may be deployed where needed to collect traffic data, and of capturing movements of individual vehicles in both *time* and *space*. When used in conjunction with traditional ground-based sensors, more comprehensive data can be collected for traffic monitoring and management.

With this in mind, the US Department of Transportation, with the support of NASA, has initiated the National Consortia on Remote Sensing in Transportation (NCRST). One of the four consortia focuses on traffic flows (NCRST-F). The goal of the initial four-year research effort of the NCRST-F consortium is to develop and demonstrate methods to use satellite and airborne sensing platforms to enhance traffic flow monitoring and management. With advances in digital sensing platforms, image processing, and computational speed, there are significant opportunities to automate traffic data collection. The research described in this paper has been supported by the NCRST-F, and represents a particular investigation of the use of airborne platforms for traffic data collection. Within this scope, we are investigating both *off-line* and *on-line* applications of remotely sensed traffic data.

This paper describes our current research in this area. While the use of remotely sensed data to monitor traffic flows is not new, this research is examining the integration of digital video, global positioning systems (GPS), and automated image processing to improve the accuracy and cost-effectiveness of data collection and reduction. To this end, the second section of the paper gives a description of the data collection methods we are using. In the third section, an algorithm for image processing is described, in which video images are processed to track individual vehicles and to derive their speeds. Some results using data from several helicopter flights in Tucson, Arizona are given in the fourth section. The paper finally offers some conclusions on the applicability of this technique for traffic monitoring and management.

DATA COLLECTION

The experiments to date have used digital video imagery, taken from an airborne platform, to observe traffic behavior. In these experiments, a consumer-grade digital video camera was mounted vertically on a skid of a helicopter, to minimize visual distortion of the imagery. At the same time, the altitude and position of the helicopter were recorded using a Global Positioning System (GPS) receiver. To register the video frames to known geographic locations, ground control points were used in an initial experiment (May 2001). A subsequent experiment (May 2002) used an inertial measurement unit (IMU) to collect data on the roll, pitch and yaw of the helicopter. High-precision GPS and IMU data is then used to automatically register the imagery, providing an accurate means of geo-referencing.

The video camera captures imagery in a 720x480 pixel sensor, with three (RGB) color bands. The video imagery is collected at 30 frames per second, but previous experiments suggest that the imagery can be sampled at rates of 1 frame per second or less with minimal loss of traffic information. Also, with a 35-mm equivalent camera focal length of 42 mm, a field of view of approximately 220 m (720 ft) was obtained at an altitude of approximately 260 m (850 ft) above ground. This field of view allows a considerable number of vehicles to be included in the image, such as a large "platoon" of vehicles or the full queues on all approaches to an intersection. For our vehicle detection technique, the scale of the frames is typically between 3 and 4 pixels/m.

In the course of this research, a series of data collection efforts took place, using aerial video from a helicopter as the primary source of traffic data. The primary objectives were to study the video image properties for different types of transportation facilities, and to characterize the quality of the data for traffic measurements. Video of representative arterials and freeways in the Tucson area (Figure 1), namely Speedway Boulevard and I-10, was obtained using a handheld digital video camera. From these flights, the data were used to develop and enhance the video image processing algorithms.

Figure 1: Data Collection Sites

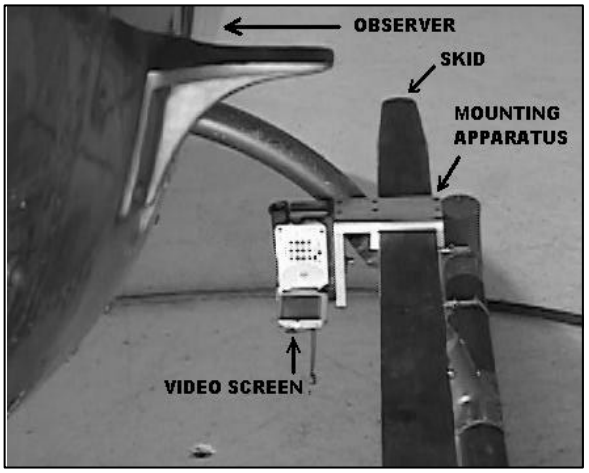


Figure 2: Video Camera Setup

IMAGE PROCESSING

Vehicle Recognition and Image Registration

The analysis of video can be greatly enhanced through automated image processing. Many researchers have studied this problem, primarily from fixed camera locations. In contrast, the use of a mobile camera platform adds additional complications to the image processing task, as the motion of both the vehicles and the camera platform must be considered. In addition, the objective of this research has been to develop an algorithm that can run close to "real time": in this case, the image processing must be completed in seconds (e.g., 2-3 sec).

The algorithm presented below for vehicle detection and tracking is still being developed and tested. It uses many well-established image processing techniques for fixed cameras. However, the main contribution of our technique is to compensate for the motion of the camera platform. To do this, the idea is to detect individual vehicles in a frame, to register consecutive frames, and then to employ image transformation, image subtraction, and object matching to track individual vehicles. This process we are using to do this is illustrated in Figure 3 and described below. The common elements of the image processing technique are included for completeness.

1. Two frames 1 or 2 sec apart are extracted from the video.
2. To accelerate vehicle identification, the frames are pre-processed. Pre-processing operations include converting the frames to 8-bit grayscale and trimming the areas of the frame that do not contain the road (e.g., the top and bottom). A geo-referenced roadway mask is used to define the roadway edges. [An example of the result of pre-processing is illustrated in Figures 6 and 7 later in the paper.]
3. Standard edge detection is performed on the frames.

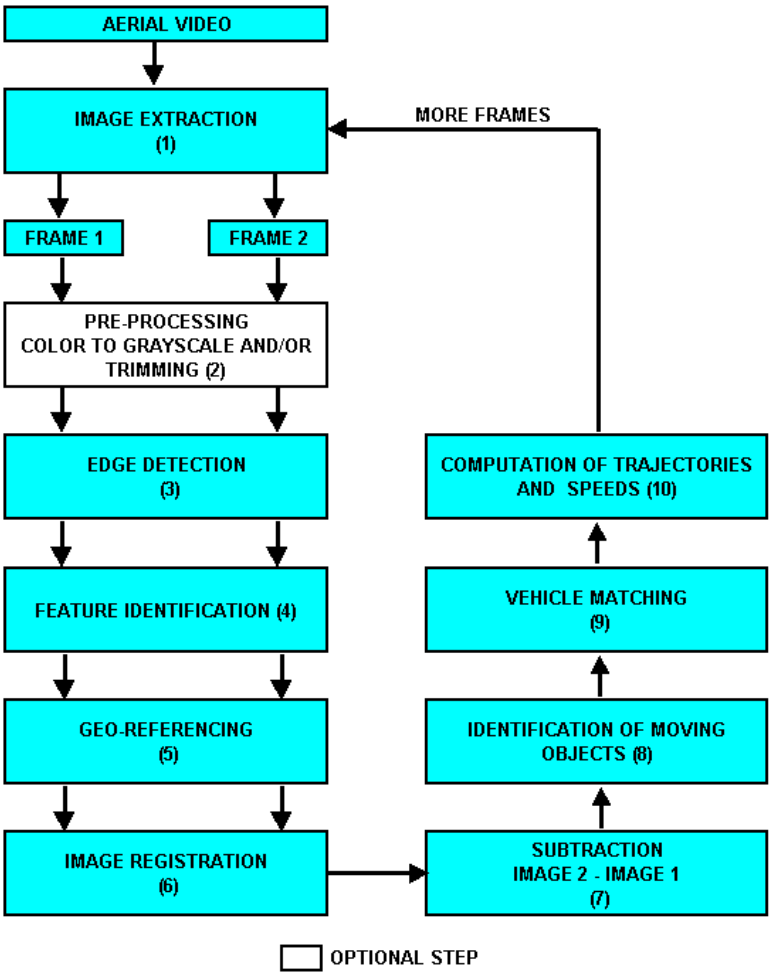


Figure 3: Flowchart of Video Image Processing

4. The frames are parsed to search for concentration of white pixels, which outline both moving and stationary objects. [Vehicle identification is also shown in Figures 6 and 7 later in the paper.]
5. The coordinates of the center of each image are found from the GPS and IMU data. The precise scale of each frame is calculated from the camera's focal length and the elevation of the helicopter with

respect to the road.

6. The second frame is registered to the first one using the location, orientation and scale information from step 5.
7. The two frames are overlapped and subtracted. The overlap between the two frames is usually, by intention, between 65% and 85%, depending on the frame sampling interval, the image scale and the speed of the helicopter.
8. After subtraction, only moving objects (vehicles) should have pixel values different from zero in the overlapping portion of the two frames. However, in many cases stationary objects may be seen as moving objects (pixel values different from zero) because of limitations in geo-referencing accuracy. This is solved by setting a minimum speed threshold for moving objects (e.g. only objects with a displacement of more than 20 pixels). The vehicles are thus identified.
9. Matching the cars in the two frames is done using a minimum cost matching algorithm. This algorithm is described in more detail in the next section.
10. After matching the vehicles, vehicle trajectories can then be derived from the time and position information in order to obtain traffic parameters such speed, density, spacing, etc. This procedure is then repeated for all the frames.

Vehicle Matching and Speed Estimation

Steps 9 and 10 of the algorithm presented above determine the motion of cars and the speeds of the various cars detected. There are two main algorithms that were used for the matching of the cars in consecutive frames:

- Greedy algorithm
- Minimum cost matching for a weighted bipartite graph (the assignment problem)

The cars detected in the first frame are to be matched with the cars in the second frame. A greedy algorithm is implemented for matching, as follows:

1. Take a car in the first frame if it has not yet been selected and initialize a variable minimum_distance to a very high value.
2. Assume the speed at which this car should travel and determine the predicted location of this car in the second frame. The algorithm is fairly robust to the assumed speed, when the speeds are consistent across vehicles (e.g., on a freeway). One might expect performance to be degraded for less consistent speeds (e.g., on arterial roadways with signalized intersections).
3. Take each car of frame 2 in turn and determine the Euclidian distance between the predicted location of the car in the first frame and the current car in the second frame.
4. If this distance is below a certain threshold limit, and it is less than minimum_distance, then record that value and the car number as the possible match for the car in frame 1. Update the value of minimum_distance to this new value.
5. After all the cars in frame 2 have been checked, and if a match is found for the car in frame 1, then make this tentative match as permanent.
6. Repeat the above steps till all the cars in frame 1 are evaluated, given a successful or unsuccessful match.

This matching algorithm was carried out on various images taken at different intervals. The various resolutions and time intervals along with the performance of this algorithm are given below:

- *Resolution 720 x 480 pixels, Time interval: 2 seconds.* For this, the greedy algorithm worked very well and gave results that were very close (about 90-100% correct matching) to the actual matching of the cars. The actual matching was done manually and compared with the results obtained by the greedy algorithm.
- *Resolution 720 x 480 pixels, Time interval: 5 seconds.* For this, the greedy algorithm worked very well and gave results that were very close (about 90-100% correct matching) to the actual matching of the cars.
- *Resolution 2265 x 300 pixels, Time interval: 10 seconds.* For this, the greedy algorithm did not work well. It gave about 50-60% correct matching.

A second method using minimum cost matching was also investigated. To match the cars in the consecutive frames, a weighted bipartite network was generated; this network was then used for matching the cars. The algorithm to accomplish this is as follows:

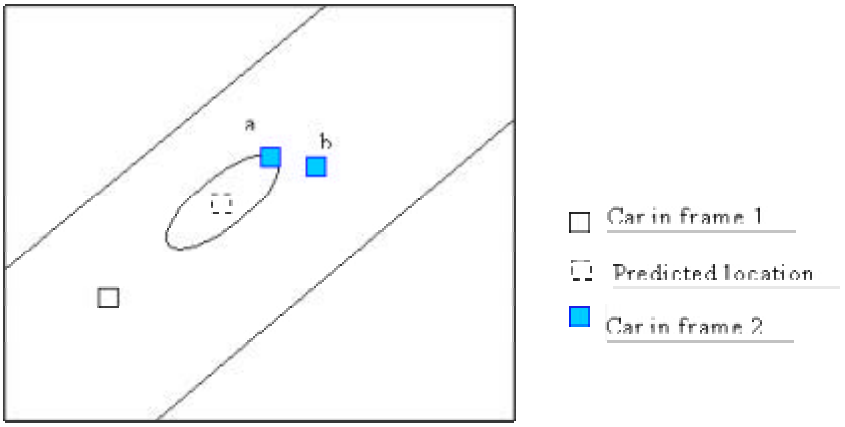
1. Take a car in the first frame if it is not yet been selected.
2. Assume the speed at which this car should travel and determine the possible location of this car.

3. Build a bipartite graph that has arcs from the predicted location of the cars in frame 1 to all of the cars in frame 2. Take each car of frame 2 in turn and make the weight assigned to this arc a function of the distance between the predicted location and the observed location (explained below).
4. Repeat the above steps for each of the cars of frame 1.
5. Using the network so formed, perform the successive shortest path algorithm to do the minimum cost matching of this weighted bipartite graph.

The determination of the arc weights was carried out by two different approaches, and the results of each of these were compared. The two techniques are explained below.

- *Single ellipse method:* Assume that the location of the car in frame 1 is determined using some predicted speed at which the car must travel. For a given car in frame 2, the problem is now to assign a weight to the link that joins these two locations. In this method, we consider that it is more likely for a car to gain or drop speed and remain in the same lane rather than change lanes. Thus we consider that around each car there are equal-likelihood ellipses that define the likelihood of the car's predicted location. These likelihood ellipses are related to the direction of motion of the car; the width is flatter that the length of the ellipse. This is demonstrated in Figure 4.

Figure 4: Single Ellipse Method



In Figure 4, it is seen that in this method, it is more likely for the car in frame 1 to be matched with car *a* of frame 2 than with car *b* of frame 2. This is because it assigns more cost to the link that is incident on car *b* than to car *a*. This algorithm was tested on the same set of images as with the greedy algorithm, with the following results. For the 2-sec images, the algorithm worked very well and gave results that were very close (about 90-100% correct matching) to the actual matching of the cars. For the 5-sec images, the algorithm also performed very well, with about 90-100% correct matching) to the actual matching of the cars. Finally, with the high-resolution 10-sec images, the algorithm also gave results that were very close (about 70-90% correct matching) to the actual matching of the cars.

- *Two ellipse method:* In this method, we assume, from the earlier images, that we know whether the vehicles are slowing (platoon contraction) or increasing speed (platoon dispersion). For example, for platoon dispersion, we consider that it is more likely for a car to gain speed rather than drop speed and also remain in the same lane rather than change lanes. Thus we consider that around each car there are two different ellipses that define the likelihood function for the location of the car. One ellipse is in the forward direction, while the other is in the backward direction. The ratio of the major axis to the minor axis for the ellipse in the backward direction is lesser than that in the forward direction. This is demonstrated in Figure 5.

In this method, it is more likely for the car in frame 1 to be matched with car *a* of frame 2 than with car *b* of frame 2. This is because it assigns more cost to the link that is incident on car *b* than to car *a*. Performance of these arc weights was similar to the single ellipse method.

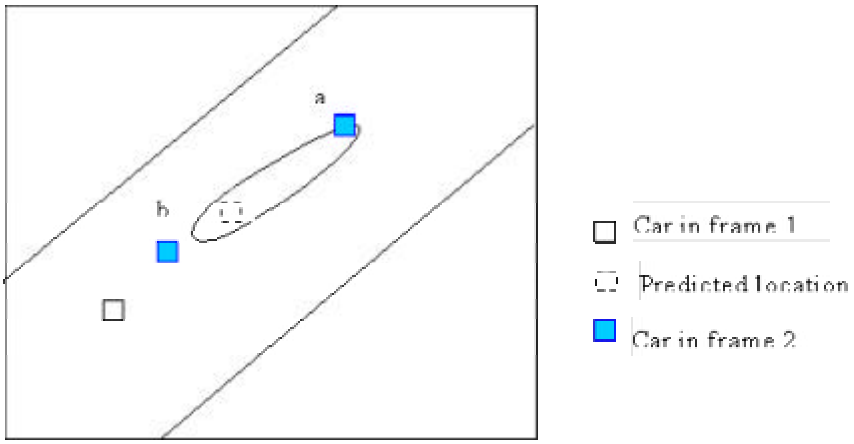


Figure 5: Two Ellipse Method

In all of the matching techniques, the method uses an initial estimate of the speed to determine a probability density function for the vehicle location. The values from the density function are used as the "costs" in the matching algorithm. In the tests described above, this algorithm seems to perform reasonably well on freeway segments, where the speeds are generally uniform. For arterials, large speed variations take place, and it is considerably more difficult to match vehicles by speed.

Once the vehicles are matched, the speed of each vehicle is computed by taking the distance between matched vehicles (i.e., pixels multiplied by the image scale) and dividing by the time interval between the two images. An example of this process is described in the following section.

EXAMPLE RESULTS

Two frames taken from a video captured in May 2002 are shown in Figures 6 and 7. These frames were taken two seconds apart, as the helicopter moved along the freeway. For the purposes of faster image processing, the images were converted to grayscale. With these images, edge detection and feature (vehicle) identification was performed, resulting in the blobs identified with bounding boxes in Figures 6 and 7. One may note that not all vehicles were recognized in each frame, particularly in Frame 2. Note also that the precise location of each bounding box is also not directly over the vehicle, due to long shadows that appeared in the imagery – the images were taken around 5 pm, with the evening sun from the west (i.e., from the tops of the images).

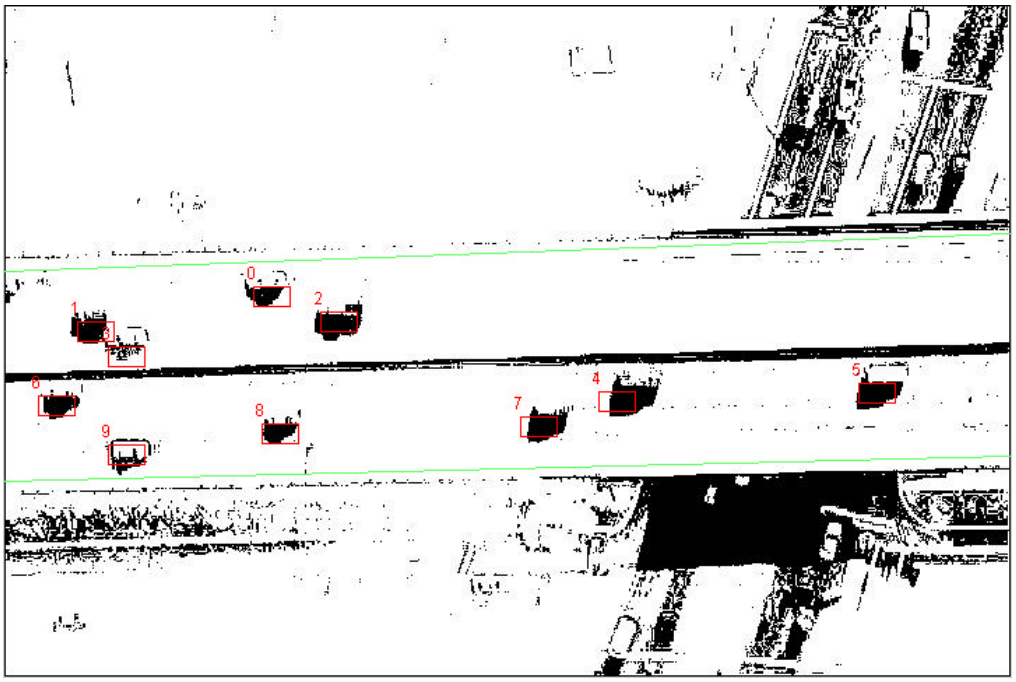


Figure 6: Video Frame 1

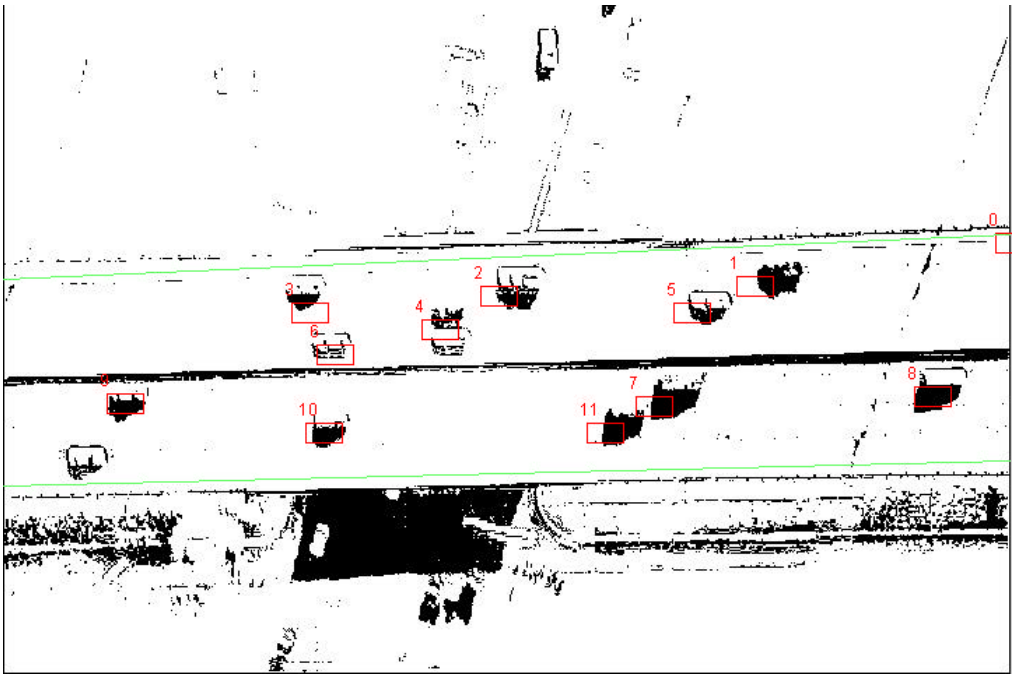


Figure 7: Video Frame 2, taken 2 sec after Frame 1

The process of registering the images, and image subtraction, was also performed on the images. The result of this process, for the two frames, is shown in Figure 8. In this example, the single ellipse method with the minimum cost matching algorithm was used to match vehicles. The results of the matching are shown in Table 1. In this case, a total of five vehicles were matched between the two frames. One might note from Figures 6 and 7 that vehicle 8 in frame 1 should have been matched with vehicle 10 from frame 2; also, vehicle 9 in frame 1 should not have been matched with vehicle 10 in frame 2. It appears that vehicle 9 in frame 1 was not re-identified in frame 2, due to a failure of the edge detection / feature identification process. Also, because of the movement of the helicopter, some vehicles in the first frame were not observed in the second frame, and vice versa.

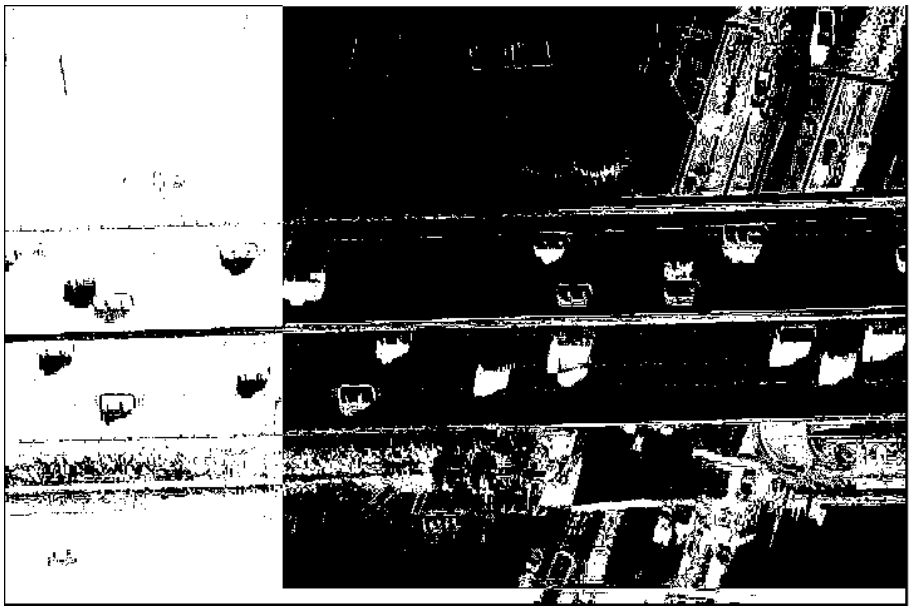


Figure 8: Images after Registration and Subtraction

The estimated speeds of the matched vehicles were calculated using the image scale and the time between frames. Vehicle speeds were fairly well clustered around 48-53 mph, as one might expect along a freeway. The lone exception was vehicle 9 (from frame 1), which had an observed speed of over 70 mph. This anomalous speed was due to the erroneous matching of vehicle 9 with vehicle 10 in frame 2. Such an erroneous speed could be discovered by a simple filter.

Table 1: Example Vehicle Matching Results

Vehicle in Frame 1	Matched Vehicle in Frame 2	Estimated Speed
0	None	
1	None	
2	None	
3	None	
4	7	48.2 mph
5	8	50.7 mph
6	9	52.7 mph
7	11	52.2 mph
8	None	
9	10	70.7 mph
None	5	
None	6	

Finally, to demonstrate the real-time nature of the proposed algorithm, the processing time for each of the major steps are shown in Figure 9. For one run of the algorithm (i.e. to match two frames, detect cars in the two frames, match the cars and determine their speeds) requires the following operations:

- Displaying the picture twice (this step is not necessary)
- Rotating the picture once
- Edge detection once
- Scaling the picture once
- Translation once
- Detecting cars once
- Image subtraction once

Note that we need to perform each operation only once, since under normal conditions one might need to process only one *additional* image in a series of images. The algorithm was coded in Java and run on a 1.8 GHz Pentium 4 personal computer with 256 MB of RAM. The total CPU time for a 500x500 pixel (trimmed) image is about 1.2 seconds. This meets the initial requirements for processing in near real time for sample images (taken every 2 seconds). Further tests suggest that the CPU time is approximately linear in the number of pixels, meaning that the full image from the video camera (720x540) could be processed in real time. To achieve similar results for higher resolution imagery, frames would need to be sampled less frequently, or else some additional processing power is needed.

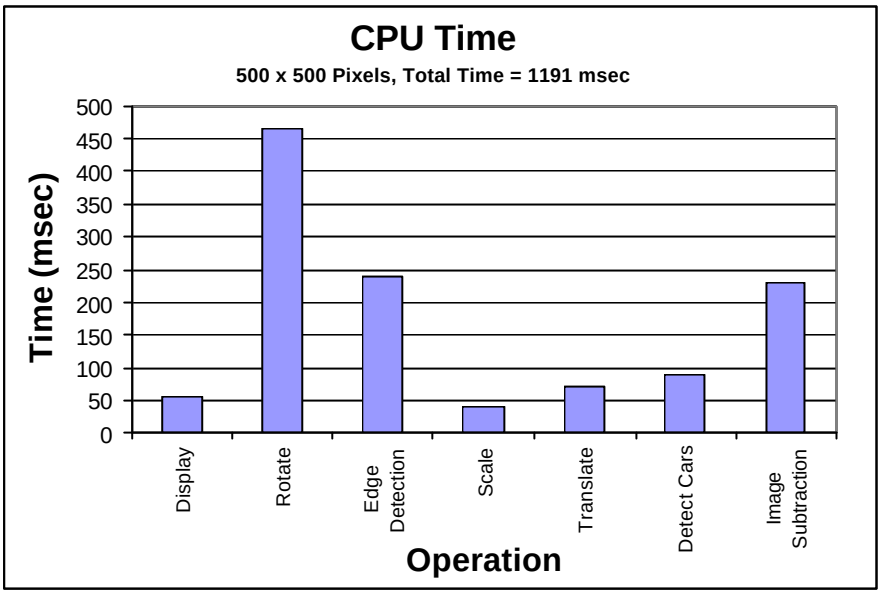


Figure 9: Example Processing Times

CONCLUSIONS

In this paper a simple technique for vehicle identification and speed estimation is given. This technique, which we are continuing to refine, shows promise that it may be run in real time in the near future, allowing one to estimate individual vehicle speeds from aerial imagery on a more or less continuous basis. Ongoing testing and refinement is now being conducted to explore the performance of the algorithms in terms of:

- CPU (image processing) times as a function of image resolution, scale, and field of view.
- Communication requirements to send the results to a ground station;
- Detection error rates; and,
- Tracking error rates.

ACKNOWLEDGMENTS

This research was supported by the National Consortium on Remote Sensing in Transportation – Flows. The support of the Research and Special Programs Administration at the US Department of Transportation, and of NASA, is gratefully acknowledged.