A Graph-Based Approach to Vehicle Tracking in Traffic Camera Video Streams

Hamid Haidarian Shahri, Galileo Namata, Saket Navlakha, Amol Deshpande, Nick Roussopoulos
Department of Computer Science
University of Maryland
College Park, MD, USA
{hamid, namatag, saket, amol, nick}@cs.umd.edu

ABSTRACT

Vehicle tracking has a wide variety of applications from law enforcement to traffic planning and public safety. However, the image resolution of the videos available from most traffic camera systems, make it difficult to track vehicles based on unique identifiers like license plates. In many cases, vehicles with similar attributes are indistinguishable from one another due to *image quality* issues. Often, network bandwidth and power constraints limit the *frame rate*, as well. In this paper, we discuss the challenges of performing vehicle tracking queries over video streams from ubiquitous traffic cameras. We identify the limitations of tracking vehicles individually in such conditions and provide a novel graph-based approach using the identity of neighboring vehicles to improve the performance. We evaluate our approach using streaming video feeds from live traffic cameras available on the Internet. The results show that vehicle tracking is feasible, even for low quality and low frame rate traffic cameras. Additionally, exploitation of the attributes of neighboring vehicles significantly improves the performance.

1. INTRODUCTION

Although there are a number of traffic monitoring technologies currently being used, from magnetic strips to radar detectors, an ever greater number of cities are deploying traffic cameras as their primary way to monitor traffic. Traffic cameras provide a more flexible way of monitoring traffic and are much cheaper to install and maintain (unlike magnetic strips which require road excavation). Moreover, currently deployed cameras can be controlled remotely with the ability to pan and zoom around a given location, allowing monitoring of a greater area, compared to other monitoring techniques. With such capabilities, not only can these cameras be used in simple tasks like counting cars and controlling traffic lights, they also have the potential to be used in more complex applications like vehicle tracking. In this paper, we examine the particular application of vehicle tracking, using ubiquitous traffic cameras.

In Section 2, we begin by describing the problem of vehicle tracking from traffic camera video. We discuss the usefulness of such an application, as well as the challenges involved in its widespread deployment. In Section 3, we discuss various techniques on how to perform vehicle tracking on videos from traffic cameras. We present techniques which involve resolving the identity of vehicles independently, as well as resolving vehicles based on using the similarities of "neighboring" cars. In Section 4, we evaluate the proposed algorithms using a traffic feed, collected from the Internet. We also evaluate the effects of factors like congestion, using a traffic simulator that we developed. We discuss relevant work in Section 5, and provide the future direction and conclusions of our work in Sections 6 and 7.

Proceedings of the 4th International Workshop on Data Management for Sensor Networks (DMSN'07), Vienna, Austria, 2007.

2. VEHICLE TRACKING USING TRAFFIC CAMERAS

A set of cars, *C*, from a video frame *t* and a set of cars, *C'*, from a video frame *t'*, are given. The goal of vehicle tracking is to identify for each car *c* in *C*, a corresponding *c'* in *C'*, if they are images of the same physical vehicle. The video frames can result from the video of a single camera or can be across multiple cameras at different angles and locations.

There is interest in vehicle tracking due to its applications for public safety and traffic monitoring. Vehicle tracking can be used by law enforcement to track criminals in cases of Amber Alerts or as an alternative to high speed pursuits in crowded urban areas. Similarly, traffic planners can use vehicle tracking to identify traffic speed and alternate routes, in the roads covered by a network of traffic cameras.

Despite the many applications, vehicle tracking is still not readily done largely due to limitations with using the ubiquitous traffic cameras. The cameras are low resolution, and frame rate of video streams are low. The data collected from various cameras is lossy, noisy and heterogeneous. They are placed at various points and angles and face geographical constraints. Moreover, they are usually spaced over long distances making the tracking of vehicles from camera to camera more difficult. The cameras are manually controlled and uncalibrated, meaning colors and distance measures maybe inconsistent. Finally, usability of cameras is influenced heavily by the environment, e.g. spots on lenses, time of day, lighting or weather conditions.

In this paper, we design several novel graph-based algorithms that exploit the correlation between the location of moving vehicles for tracking. We provide an experimental evaluation on streaming videos collected from different cities and conditions, as well as on simulated traffic data, generated to test the effects of factors such as congestion. We show that our graph based approaches perform *more accurately*, and are less susceptible to *low image quality and low frame rates*, in comparison to an approach using only the attributes of a vehicle. The attributes of a vehicle (represented by a blob) in a video stream, include: color, size, x and y coordinates on the image, and width to height ratio.

3. GRAPH-BASED ALGORTIHMS FOR VEHICLE TRACKING IN VIDEO STREAMS

Our algorithms consist of three main steps:

1. Identify the vehicles and their properties in a frame.
2. Create a graph, representing the proximity information about the vehicles in that frame.
3. Find a mapping between vehicles in two frames, using the graph and vehicle properties.

In this section, we first discuss the preprocessing step to extract the vehicle properties and the proximity information, and then present the matching algorithms.

Preprocessing: A basic approach for the detection of moving objects on images is background subtraction. Initially, an image background is constructed for each camera using the corresponding input video. The input videos contain many buildings, ads, traffic signs, and other independent distractions. In order to eliminate their influence, we define a region of interest for each camera, which is set manually to the area of the road that we are interested in monitoring. The background subtraction values are kept only in the region of interest.

With the background subtraction updated according to the region of interest, motion pixels are identified as the locations in the region where absolute difference in appearance is greater than a threshold value. After that, blobs which probably represent cars are found, as the connected components of the motion pixels in the image. Components with a very small area are filtered out.

Ground Truth for Algorithm Evaluation: While our algorithms can provide a mapping between vehicles in two frames, it is important to have a fast and automated procedure for determining the correct mappings, for evaluating our results with the truth. We devised an interesting method for producing the ground truth using high frame rate video. When the frame rate is high, finding the closest center of a blob in frame $t+1$, to any given center of the blob in current frame t, provides the corresponding blobs in the two frames. This serves as the correct mapping for evaluating our algorithms. Hence, *ground truth generation* requires a high frame rate, which guarantees that blobs do not move too much between two consecutive frames. Note that ground truth is only required for the *evaluation and comparison* of various algorithms, and the algorithms do not require any specific frame rate, when deployed. Although a higher frame rate increases the accuracy of results, our experiments show that the algorithms perform surprisingly well with very low frame rates.

Next, we present several algorithms for matching vehicles between the frames of a video. The first algorithm performs the matching purely based on attributes. The second algorithm creates a graph based on the formation of vehicles, at a given frame. The next four algorithms create a weighted measure for the similarity of, a vehicle and its neighbors in frame t, and compare it to a vehicle and its neighbors in frame t'. Matching is done based on which vehicle in the next frame has the most similar attributes and neighbors to the vehicle that we are trying to match.

3.1. Attribute-based Matching

In attribute based matching, a greedy matching of a car from one frame to a car in the subsequent frame is performed. For a given car in the first frame, we compare its attributes to all the cars in the second frame, looking for a matching that minimizes a given similarity measure. In this case, we use a similarity measure based on the percent difference between the color, size and length to width ratio of the cars between the two frames. This is repeated for all cars until every car in the first frame is matched to a car in the second frame. If at any point, two cars are mapped to same car in the second from, the one with the closest similarity to the car in the second frame retains the mapping while the other car is mapped to -1. A value of -1 signifies that no mapping is available since the car given in first frame may not be present in the second frame.

3.2. Formation-based Matching

In this algorithm, the highest car c (current car) in frame t is identified. Then car c is connected to its closest neighbor n, in frame t. Now n becomes the current car, and takes the role of c, and the process is repeated, i.e. we find the closest neighbor to current car c that has not been visited. This procedure is repeated until all cars in frame t are connected to each other through a path P1 (see Figure 1). The actual shape of the path is dependant on the local positioning of the cars. This same procedure is repeated for the subsequent frame in the video, which creates path P2. Finally, the cars are matched by walking path P1 and trying to match the cars in path P2, which are in similar positions (i.e. the first car in P1 is likely to match the first car in P2, etc.). If one car in P1 does not match, with the car under consideration in P2, we move to the next car in P2, in hope of finding a match there. We illustrate the algorithm in Figure 1.

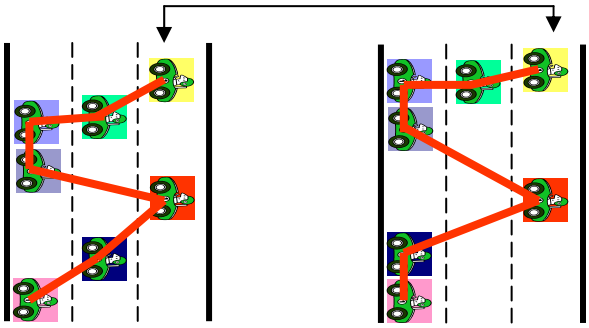

Figure 1: Formation-based Matching

Relational Weighting: The next four algorithms calculate the similarity based on similarity between the car, c, and its defined neighbors in the first frame, and a car, c', and its defined neighbors in the second frame. The similarity score between two cars is given by the equation $Score(c,c') = \alpha S_0 + (1-\alpha)S_K$ where S_0 is the attribute similarity of the cars c and c', and S_K is the sum of the similarities of all neighbors of c to the most similar car in the set of neighbors of c'. The pseudo code of our algorithm is given in Algorithm 1 and the calculation is illustrated in Figure 2. The next four algorithms follow this equation and vary only in how the neighbors are specified for a given car in a given frame. The algorithms are based on the domain knowledge about U.S. driving conventions, as well as

observations made from the traffic videos. We discuss the specifics about these motivations when applicable.

3.3. Nearest Neighbor Matching (NNM)

In Nearest Neighbor Matching (NNM) shown in Figure 3, for a given car *c* in the given frame, *c* is connected with the single neighbor whose center is the closest by Euclidean distance. There is no limitation to where the neighbor is relative to a given car, and in sparse traffic conditions, the nearest neighbor may be many lanes away.

Algorithm 1: Relational Weighting

```
1. For each pair of consecutive frames
t,t':
2.    Compute the attributes (cCol, cSize,
cRatio, cPos) for each car c in t.
3.    Compute the attributes (c'Col, c'Size,
c'Ratio, c'Pos) for each car c' in t'.
4.    Create a graph Gt with a node for each
car, and an edge  (u,v) if u is v's closest
neighbor in the same lane. Create the same
graph Gt' for cars in t'.
5.    Let cN, cN' be neighbors of c and c',
respectively.
6.    For each car c in t:
7.       Set difference = infinity,
matchingCar = -1.
8.       For each car c' in t':
9.          Score(c,c') = alpha*(cCol-c'Col +
cSize-c'Size + cRatio-c'Ratio) + (1-
alpha)*(cNCol-cN'Col + cNSize-cN'Size +
cNRatio-cN'Ratio)
10.         If Score(c,c') < difference
11.            matchingCar = c'.
12.            difference = Score(c,c')
```

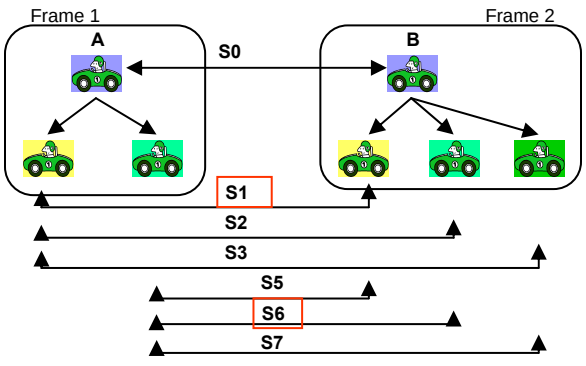

Figure 2: Relational Weighting

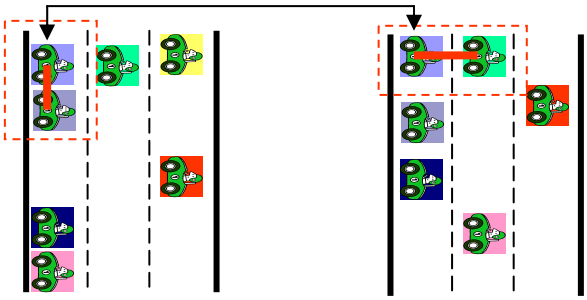

Figure 3: Nearest Neighbor Matching

3.4. Nearest Following Neighbor Matching (NFNM)

The difficulty with the Nearest Neighbor Matching is that a neighbor can be from any direction, including a car from the opposite lane. In traffic situations where one lane may be moving faster than another, as is the case in roads leading to exits and intersections, neighbors who are in another lane are not likely to remain the neighbor of a car for very long. In order to address this, we define a second algorithm similar to Nearest Neighbor, but with the added restriction that the neighbor must be a car in the same lane and must be behind the car we are trying to connect. The difference between the algorithms can be seen in Figure 3 and Figure 4.

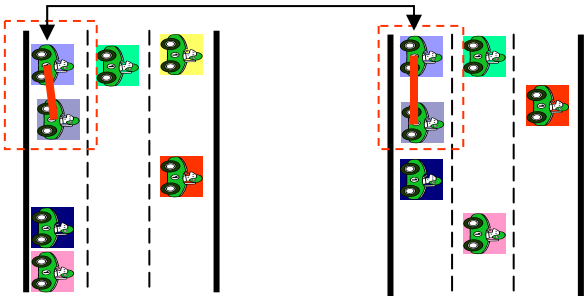

Figure 4: Nearest Following Neighbor Matching

3.5. Proximity Based Neighbors Matching (PBNM)

Proximity Based Matching is designed for major highways where cars often travel at the maximum speed limit. The motivation is that due to legal restrictions in speed in the U.S., cars will travel at the same speed and thus remain in the same position, relative to each other, down long, major roads. Thus, in PBNM, shown in Figure 5, a car is defined as neighbors with all cars in the same frame, within a given forward and backward distance from a car. In order to address the issue of perspective in these cameras, where cars further away from the camera appear smaller than closer ones, we define the distance forward and backward from a car based on the length of the car. Specifically, neighbors are defined as all cars whose centers are 1.5 times the length, in front of or behind a given car.

3.6. Same Lane Neighbors Matching (SLNM)

PBNM suffers from the same problems that NNM does. In cases where some lanes are faster than others and in cases of congestion, neighbors of a car might be changing too quickly to be useful. A modification that we evaluate is to base the neighbors on whether or not they are in the same lane as the

given car. All vehicles in the same lane are considered neighbors with each other. In our evaluations, we define lanes based on the width of the given vehicle. Specifically, all cars within 1.5 times the width of the given car, left or right, are considered neighbors. We chose to set it to 1.5 rather than just based on the width of the car in order to capture vehicle which are switching between lanes. The algorithm is demonstrated in Figure 6.

3.7. Random Matching

As a baseline for comparing the matching algorithms and making sure that they are effective, a random matching of vehicles was tested (i.e. in each frame, each car is matched with a random car in the next frame). The results of other algorithms are compared to this baseline.

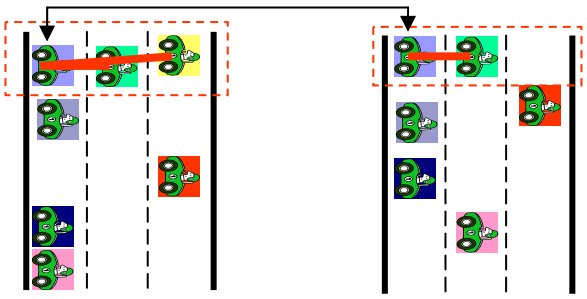

Figure 5: Proximity Based Neighbors Matching

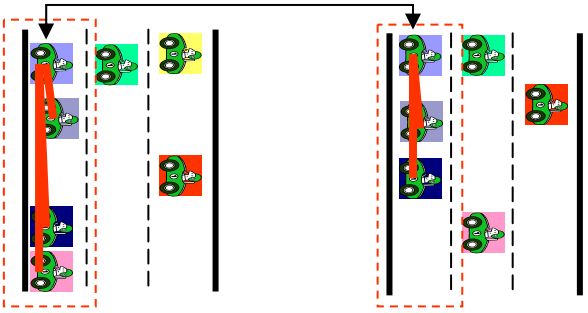

Figure 6: Same Lane Neighbors Matching

4. EXPERIMENTAL EVALUATION

In this section, we evaluate our algorithms on streaming videos from different live cameras. We have run our algorithms on many cameras, but only report some representative experiments due to space constraints. In our analysis, we also vary the camera's frame rate to see how our accuracy degrades as less and less frames are provided (i.e. for a frame step of five, we run our matching algorithms on frames 1 and 6, then 6 and 11, etc.) A sensitivity analysis on the alpha parameter is provided, and we also consider the problem of tracking at night time, when only headlights are visible. Moreover, we developed a traffic simulator to test our algorithms under different conditions, as explained below.

Zinfandel Daytime, San Francisco: This video is from a highway in California taken during the day (see Figure 7). In Figure 8, the vertical axis shows the fraction of error in matching and the horizontal axis shows the number of frames that were skipped, to experiment with low frame rates. We can see in Figure 8 that the graph-based relational weighting algorithms do much better than the attribute and formation based matching algorithms. Results for random matching are nearly constant at 90% error rate (not shown). The Nearest Following Neighbor Matching (NFNM) algorithm produces very good results with a 35% error rate at frame step 20 (skipping 20 frames, i.e. very low frame rate). For high frame rates, all our relational matching algorithms result in an error below 9%. The formation approach measures the extent to which the overall formation of cars remains constant, and seems to suffer from fluctuations in the speed of cars, at high frame steps. Similar results have been achieved on different types of roads, and imply that our algorithms are quite invariant to the type of road.

Figure 7: Zinfandel, San Francisco Traffic Camera (day)

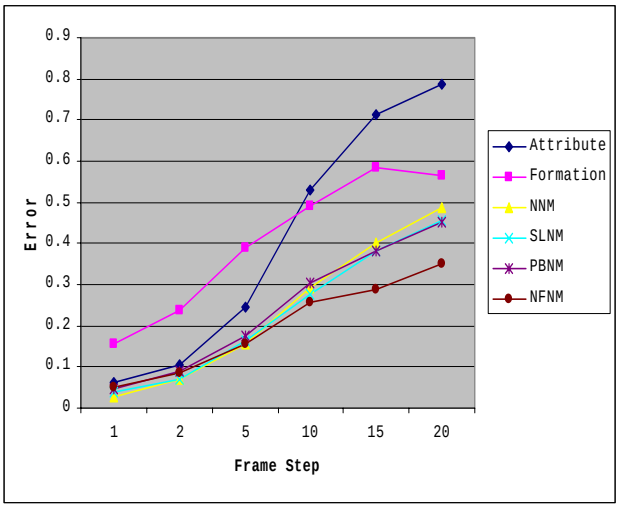

Figure 8: Algorithm Comparison, at Zinfandel (day)

Sensitivity Analysis of Alpha Parameter: In the four graph-based relational weighting algorithms, a parameter alpha can be set to control the amount of influence that the relational attributes of the vehicles have in the overall matching. We evaluated our algorithms for different values of alpha and present the results in Figure 9. Note that when alpha=1, the relational attribute is completely ignored and the algorithms work exactly like attribute based matching. The results show that our algorithms do not perform well for alpha less than 0.5. This means that when doing the matching, the similarity between the attributes of two cars must be weighted more than the similarity of their neighbors. Moreover, we see that in general, the best performance is achieved at an alpha value of 0.8. We used this alpha value in all experiments in this paper, when setting alpha as constant.

Zinfandel Night Time, San Francisco: Cars in this video were hardly visible, and mostly identified by their tail lights, though some were missed entirely due to darkness (see Figure 10). In Figure 11 the error rate is higher than Figure 8, as expected, but considering the quality of the video, the results are still impressive and comparable to day time for high frame rates. Furthermore, we believe that using better vision tools would greatly aid our algorithms, which we discuss in future work.

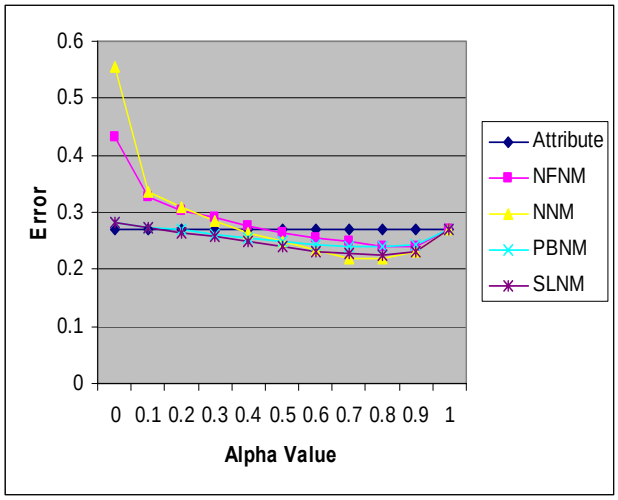

Figure 9: Sensitivity Analysis of the Alpha Parameter

Figure 10: Zinfandel, San Francisco Traffic Camera (night)

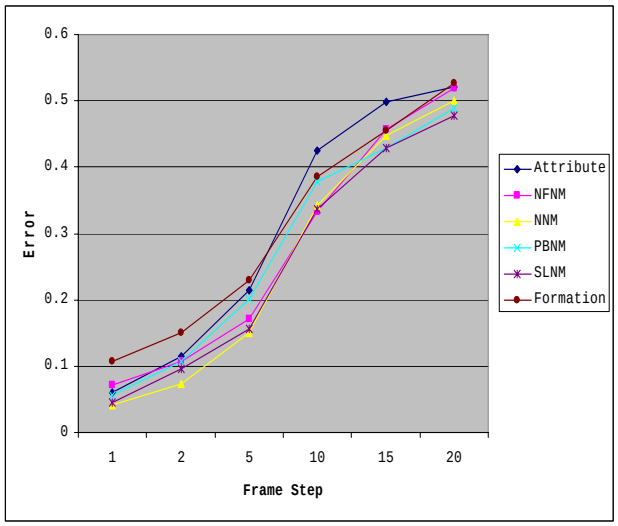

Figure 11: Algorithm Comparison, at Zinfandel (night)

Analyzing the Effect of Road Congestion, using Traffic Simulations: One problem with using videos from real cameras is that there is no control over several traffic parameters, such as the speed of cars or density of traffic. Hence, we developed a traffic simulator, which allowed us to not only isolate and study certain conditions in detail, but also test our algorithms under more diverse and extreme conditions, to ensure their generality.

We do not explain the simulator details here due to lack of space. To experiment with different road traffics, we varied road congestion from 5 to 25 cars per frame (with a window height and width of 400 pixels). The results of the simulation (Figure 12) show that the NNM approach worked astoundingly well, with an error rate below 10% for all different congestion conditions. Note that the frame step is set to one.

Our simulator is also capable of generating data for the multiple-camera case, where we try to track cars as they move from camera to camera. Our initial results are promising, though we do not present them here.

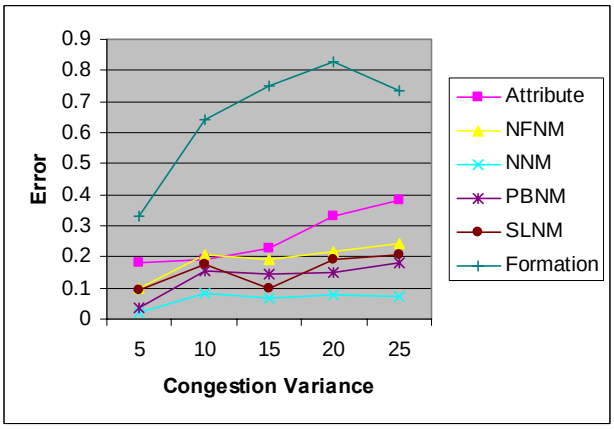

Figure 12: Effect of Congestion Variance

Analyzing the Effect of Speed Variance, using Traffic Simulations: In this experiment, each car was assigned a random speed, in the given range, i.e. a greater range implies a greater speed variance. In Figure 13, each point on horizontal axis shows the speed range. For different speed variances, the relational weighting algorithms perform better than attribute and formation based matching, which is consistent with our previous results.

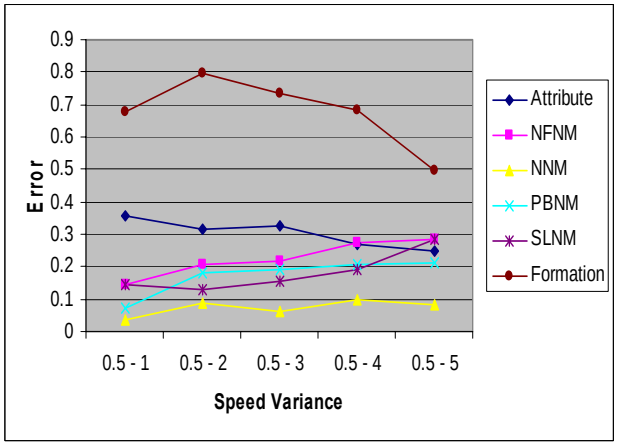

Figure 13: Effect of Speed Variance

5. RELATED WORK

There has been much work in the area of traffic monitoring; due to space constraints, we briefly discuss only the most related work to ours. Different sensing mechanisms have been investigated in the literature for traffic monitoring. These include the use of cell phones, inductive strip probes on the road, GPS-based monitoring [Hul06], and traffic cameras [Smi04]. Our algorithms use simple cameras with relatively low quality video and the main focus of our research is on developing vehicle-tracking algorithms that exploit the relative positions of vehicles in relation to one another. There is a large body of work on detecting unique tuples in a database and various adaptive and rule-based systems have been proposed for this task [Hai06]; our approach is similar in spirit to relational clustering-based entity resolution [Bha07].

A common approach to scale-invariant object recognition in images involves Scale-Invariant Feature Transformations (SIFT) [Low04]. Briefly, SIFT is an algorithm which extracts scale-rotation-and-perspective-invariant features from images, using key point localization and assignment. One possible approach to using SIFT in our problem is to extract images of each car in each frame and use SIFT to see if each of these images appear in the subsequent frame. This would in effect allow us to tell which cars in which frames match. Soh et al. attempt to count traffic in a similar manner to ours [Soh95]. First, they create an "extracted road structure" similar to our background subtraction, and then detect vehicles using the Prewitt edge detection operator. To define a region of interest they first compute the average speed of the cars for the first few frames, and set the region (automatically) such that each car is only in the region once. However, they do not use relational information to aid in vehicle tracking. Traffic simulators are also very well-studied, and there exist several methods for doing traffic simulation; e.g. VATSIM [Lei01] models automated vehicles with different sensors, controllers, and traffic networks with real-time traffic control and route guidance.

6. FUTURE WORK

In our work, we have used simple vision techniques to extract car data from traffic videos. For one, we use background subtraction to identify blobs and simple heuristics to do segmentation of those blobs. The result is that there are cases where a whole car is not captured in a blob and where a single blob actually represents several cars. There are more sophisticated vision techniques that we can use to address this problem, including frame differencing. Similarly, attributes of blobs that are exploited in our work are: length-width ratio, color, and size. There are other features that can be extracted from the image, for example the span of headlights, which might increase the accuracy of our algorithms. It is also possible to try different clustering and classification techniques from the machine learning literature, though the search space is huge and proper attribute selection needs to be addressed first. Next, the creation of the graph is the crux of all the relational-matching algorithms, implemented in this project. We are unaware of any other work which uses graph-based techniques to perform matching. Although, the techniques we used were mostly developed based on our intuition of how cars collectively move on the road, other graph-creation algorithms could be considered.

Our traffic simulator is capable of generating data from multiple cameras, where one camera is followed by another camera with a blind spot in between. We plan on pursuing the multi-camera case and using other traffic simulation software to improve our synthetic data. Finally, we are currently working with the University of Maryland's Center for Advanced Transportation Technology Laboratory (CATT Lab) in order to acquire more video feeds from the Washington DC area. The ability to capture several feeds directly from the cameras will be invaluable in furthering our research.

7. CONCLUSIONS

In this paper, we discuss the application of vehicle tracking using low quality and low frame rate traffic camera video streams. We discuss the challenges which need to be overcome for this problem and propose novel solutions using both the individual vehicle attributes, as well as the attributes of neighbors. We present various ways of defining what a neighbor is, taking into account the traffic laws and conventions. We have collected a relatively large video stream archive, from traffic cameras in geographically diverse locations. The dataset is above one Gigabyte in size and spans over three weeks. We are planning to release this archive, which could prove to be valuable for continuing this research. Using both real-world data, as well as simulated data for various conditions, we show that vehicle tracking is feasible even in *low resolution*, *low frame rate* video streams. Moreover, we show that using attributes of neighboring vehicles, particularly our Nearest Neighbor Matching algorithm, significantly improves the accuracy and is more resilient to the low quality of available feeds, from resource-constrained traffic cameras.

REFERENCES

[Bha07] Bhattacharya I., and Getoor L., “Collective entity resolution in relational data”, ACM Transactions on Knowledge Discovery from Data, 2007.

[Bun00] Bunke, H, “Graph matching: Theoretical foundations, algorithms, and applications.” Proc. Vision Interface, 2000.

[Hai06] Haidarian Shahri, H., et al., “Eliminating Duplicates in Information Integration: An Adaptive, Extensible Framework.” IEEE Intelligent Systems, Vol. 21, No. 5, pp. 63-71, 2006.

[Hul06] Hull, B., Bychkovsky, V., Chen, K., Goraczko, M., Miu, A., Shih, E., Zhang, Y., Balakrishnan, H., Madden, S., “CarTel: A Distributed Mobile Sensor Computing System.” SenSys, 2006.

[Lei01] Lei, et. al, “VATSIM: a simulator for vehicles and traffic,” Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE.

[Low04] Lowe, D.G, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, 60(2), pp. 91-110, 2004.

[Smi04] Smith, B., Zhang, H., Fontaine, M., and Green, M., “Wireless Location Technology-Based Traffic Monitoring: Critical Assessment and Evaluation of an Early-Generation System.” Journal of Transportation Engineering, October, 2004.

[Soh95] Soh, Jung, et. al, “Analysis of Road Image Sequences for Vehicle Counting.” IEEE Proc. International Conference of Systems, Man, and Cybernetics, 1995.