

Vision-Based Monitoring of Weaving Sections

Osama Masoud
Artificial Intelligence, Robotics, and Vision Laboratory
Department of Computer Science,
University of Minnesota,
4-192 EE/CS Building, 200 Union St. SE,
Minneapolis, MN 55455, USA
{masoud,npapas}@cs.umn.edu

Eil Kwon*
*Center for Transportation Studies
University of Minnesota,
200 Transportation and Safety Building,
511 Washington Ave. SE,
Minneapolis, MN 55455, USA
kwonx001@maroon.tc.umn.edu

Abstract

This paper presents algorithms for vision-based monitoring of weaving sections. These algorithms have been developed for the Minnesota Department of Transportation in order to acquire data for several weaving sections in the Twin Cities Area. Unlike commercially available systems, the proposed algorithms can track and count vehicles as they change lanes. Furthermore, they provide the velocity and the direction of each vehicle in the weaving section. Experimental results from various weaving sections with various weather conditions are presented. The proposed methods are based on the establishment of correspondences among blobs and vehicles as the vehicles move through the weaving section. The blob tracking problem is formulated as a bipartite graph optimization problem.

1. Introduction

Traffic control at weaving sections is a challenging problem since vehicles are crossing paths, changing lanes, or merging with through traffic as they enter or exit a highway. There are two types of weaving sections depending on the number of points of entry and the number of points of exits (single weaving section and multiple weaving section). Sensors which are based on lane detection or tripline detection often fail to monitor weaving sections since they cannot track vehicles which cross lanes. The fundamental problem that needs to be addressed is the establishment of correspondence between a traffic object “A” in lane “x” and the same object “A” in lane “y” at a later time. For example, vision systems which depend on multiple detection zones simply cannot establish correspondences since they assume that the vehicles stay on the same lane.

The motivation behind this work is to compensate for inefficiencies in existing systems as well as to provide more comprehensive data about the weaving section being monitored. In this work, we use visual information as our sensory input. The rich information provided by vision sensors is essential for extracting information that may cover several

lanes of traffic. The information we are considering includes:

1. Extraction of vehicles and thus a count of vehicles.
2. Velocity of each vehicle in the weaving section.
3. Direction of each vehicle. This is actually a trajectory versus time rather than a fixed direction (since vehicles may change direction while in a highway weaving section).

Our goal is to build a portable system which can gather data in various settings (something similar to the workzone variable message sign units). Such a system can have a wide range of uses including:

1. Data-collection for weaving sections,
2. Optimal traffic control at weaving sections,
3. Real-time traffic simulation, and,
4. Traffic prediction.

Our system uses a single fixed camera mounted in an arbitrary position. We use simple rectangular patches with a certain dynamic behavior to model vehicles. Overlaps among vehicles and occlusions are dealt with by allowing vehicle models to overlap in the image space and by maintaining their existence in spite of the disappearance of some features.

Vision-based tracking of vehicles has been attempted in the past through several approaches. Some researchers used three-dimensional models for vehicles [1]. The disadvantage of such an approach is that a model is needed for every type of vehicle that could appear in the scene. Active contours were also used to track vehicles [6]. However, initialization is problematic especially for partially occluded vehicles. Another tracking approach is feature-based. Features, such as corners and edges, are tracked and grouped based on their spatial and temporal characteristics. Such systems, however, are computationally expensive due to correlation in feature tracking. Smith [8] used custom hardware to achieve tracking at 25Hz. Beymer *et al.* [2] used 13 C40 DSPs to achieve up to 7.5Hz tracking in uncongested traffic. Finally, some researchers tracked connected regions in the image (blobs) [4,5]. Region-based tracking suffers from segmentation problems in cluttered scenes because blobs of several vehicles may merge into a single blob. Also, a single vehicle may be composed of several blobs. Our system, which can be classified as region-based, deals with this problem by allowing the relation between vehicles and blobs to be many-to-

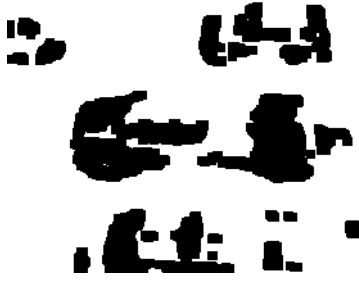


Figure 1. Feature image (corresponds to the image in Figure 4(c)).

many. This relation is updated iteratively depending on the observed blobs behavior and predictions of vehicles behavior. Three levels of abstractions are used (this structure is based on our earlier work on pedestrian tracking [7]). Features are obtained at lowest level by a simple recursive filtering technique. In the second level, which deals with blobs, feature images are segmented to obtain blobs which are subsequently tracked. Finally, tracked blobs are used in the vehicles level where relations between vehicles and blobs as well as information about vehicles is inferred.

The next three sections describe the processing done at the three levels mentioned above. Finally, experimental results and conclusions are presented.

2. Features level

Features are extracted using a recursive filtering technique (a slightly modified version of Halevi and Weinshall's algorithm [3]). This technique is simple, time-efficient and therefore, suitable for real-time applications. A weighted average at time i , M_i , is computed as

$$M_i = \alpha \times I_{i-1} + (1 - \alpha) \times M_{i-1}, \quad (1)$$

where I_i , is the image at time i , and α is a fraction in the range 0 to 1 (typically 0.5). The feature image at time i , F_i , is computed as follows: $F_i = |M_i - I_i|$. The feature image captures temporal changes (features) in the sequence. Moving objects result in a fading trail behind them. The feature image is thresholded to remove insignificant changes. Figure 1 shows a typical thresholded feature image.

As will be explained below, our algorithm makes use of bounding boxes in blob tracking. The bounding boxes track most effectively when the targeted blobs fill the box, which is achieved when they are aligned horizontally or vertically (i.e., diagonal blobs are not desirable). Because the camera pan may be large as shown in Figure 2, we warp the original image so that vehicles appear perpendicular to a virtual camera's optical axis (Figure 5(b)) and then perform feature extraction.



Figure 2. Original image. Camera axis is not perpendicular to the street.

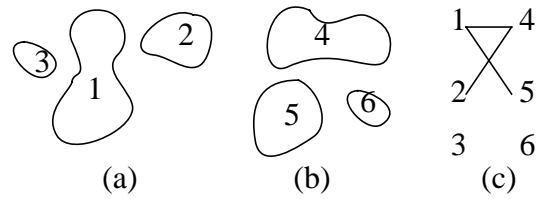


Figure 3. (a) Blobs in frame $(i - 1)$. (b) Blobs in frame i . (c) Relationship among blobs.

3. Blobs level

At the blobs level, blob extraction is performed by finding connected regions in the feature image. A number of parameters is computed for each blob. These parameters include perimeter, area, bounding box, and density (area divided by bounding box area). We then use a novel approach to track blobs regardless of what they represent. Our approach allows blobs to merge, split, appear, and vanish. Robust blob tracking was necessary since the vehicles level relies solely on information passed from this level.

3.1. Blob tracking

When a new set of blobs is computed for frame i , an association with frame $(i - 1)$'s set of blobs is sought. The relation between the two sets can be represented by an undirected bipartite graph, $G_i(V_i, E_i)$, where $V_i = B_i \cup B_{i-1}$. B_i and B_{i-1} are the sets of vertices associated with the blobs in frames i and $i - 1$, respectively. We will refer to this graph as a *blob graph*. Figure 3 shows an example where blob 1 split into blobs 4 and 5, blob 2 and part of blob 1 merged to form blob 4, blob 3 disappeared, and blob 6 appeared.

The process of blob tracking is equivalent to computing G_i for $i = 1, 2, \dots, n$, where n is the total number of frames. We do this by modeling the problem as a constrained graph optimization problem where we attempt to find the graph which minimizes a cost function.

Let $N_i(u)$ denote the set of neighbors of vertex $u \in V_i$,

$N_i(u) = \{v | (u, v) \in E_i\}$. To simplify graph computation, we will restrict the generality of the graph to those graphs which do not have more than one vertex of degree more than one in every connected component of the graph. This is equivalent to saying that from one frame to the next, a blob may not participate in a splitting and a merging at the same time. We refer to this as the *parent structure constraint*. According to this constraint, the graph in figure 3(c) is invalid. If, however, we eliminate the arc between 1 and 5 or the arc between 2 and 4, it will be a valid graph. This restriction is reasonable assuming a high frame rate where such simultaneous split and merge occurrences are rare.

To further reduce the number of possible graphs, we use another constraint which we call the *locality constraint*. With this constraint, vertices can be connected only if their corresponding blobs have a bounding box overlap area which is at least half the size of the bounding box of the smaller blob. This constraint, which significantly reduces possible graphs, relies on the assumption that a blob is not expected to be too far from where it was in the previous frame. This is also reasonable to assume if we have a relatively high frame rate. We refer to a graph which satisfies both the parent structure and locality constraints as a *valid* graph.

To find the optimum G_i , we need to define a cost function, $C(G_i)$, so that different graphs can be compared. A graph with no edges, i.e. $E_i = \emptyset$, is one extreme solution in which all blobs in V_{i-1} disappear and all blobs in V_i appear. This solution has no association among blobs and should therefore have a high cost. In order to proceed with our formulation of the cost function, we define two disjoint sets, which we call *parents*, P_i , and *descendents*, D_i , whose union is V_i such that $D_i = \bigcup_{u \in P_i} N_i(u)$. P can be easily constructed by selecting from V_i all vertices of degree more than one, all vertices of degree zero, and all vertices of degree one which are only in B_i . Furthermore, let

$S_i(u) = \sum_{v \in N_i(u)} A(v)$ be the total area occupied by the

neighbors of u . We now write the formula for the cost function as

$$C(G_i) = \sum_{u \in P_i} \frac{|A(u) - S_i(u)|^2}{\max(A(u), S_i(u))}. \quad (2)$$

This function favors associations in which blobs do not change much in size. It also favors changes in large blobs' sizes to changes in small blobs' sizes which is intuitively reasonable.

Using this cost function, we can proceed to compute the optimum graph. First, we notice that given a valid graph $G(V, E)$ and two vertices $u, v \in V$, such that $(u, v) \notin E$, the graph $G'(V, E \cup \{(u, v), (v, u)\})$ has a lower cost than

G provided that G' is a valid graph. If it is not possible to find such a G' , we call G *dense*. Using this property, we can avoid some useless enumeration of graphs which are not dense. In fact, this observation is the basis of our algorithm to compute the optimum G .

Our algorithm to compute the optimum graph works as follows: A graph G is constructed such that the addition of any edge to G makes it violate the locality constraint. There can be only one such graph. Note that G may violate the parent structure constraints at this moment. The next step in our algorithm systematically eliminates just enough edges from G to make it satisfy the parent structure constraint. The resulting graph is valid and also dense. The process is repeated so that all possible dense graphs are generated. The optimum graph is the one with the minimum cost.

4. Vehicles level

The input to this level is tracked blobs and the output is the spatio-temporal coordinates of each vehicle. The relationship between vehicles and blobs in the image is determined at this point. Each vehicle is associated with a list of blobs. Moreover, a blob can be shared by more than one vehicle. Vehicles are modeled as rectangular patches with a certain dynamic behavior. We found that for the purpose of tracking, this simple model adequately resembles the vehicle shape and motion dynamics. We now present this model in more detail and then describe how tracking is performed.

4.1. Vehicle model

Our vehicle model is based on the assumption that the scene has a flat ground. A vehicle is modeled as a rectangular patch whose dimensions depend on its location in the image. The dimensions are equal to the projection of the dimensions of an average size vehicle at the corresponding location in the scene. The patch is assumed to move with a constant velocity in the scene coordinate system. The patch acceleration is modeled as zero-mean, Gaussian noise to accommodate for changes in velocity. The discrete-time dynamic system for the vehicle model can be described by the following equation:

$$\mathbf{x}_{t+1} = \mathbf{F}\mathbf{x}_t + \mathbf{v}_t, \quad (3)$$

where the subscripts indicate time, $\mathbf{x} = [x \ x' \ y \ y']^T$ is the state vector consisting of the vehicle location, (x, y) and velocity, (x', y') , \mathbf{F} is the transition matrix of the system, and \mathbf{v}_t is a sequence of zero-mean, white, Gaussian process noise with covariance matrix \mathbf{Q} .

4.2. Vehicle tracking

The next five subsections describe one tracking cycle.

4.2.1. Relating vehicles to blobs. We use simple rules to refine the relationship between vehicles and blobs. If a vehicle was related to a blob in frame $(i - 1)$ and that blob is related to another blob in the i th frame (through a split, merge, etc.), then the vehicle is also related to the latter blob. More details can be found in [7].

4.2.2. Prediction. Given the system equation as in the previous section, the prediction phase of the Kalman filter is given by the following equations:

$$\begin{aligned}\hat{\mathbf{x}}_{t+1} &= \mathbf{F}\mathbf{x}_t, \\ \hat{\mathbf{P}}_{t+1} &= \mathbf{F}\mathbf{P}_t\mathbf{F}^T + \mathbf{Q}.\end{aligned}\quad (4)$$

Here, $\hat{\mathbf{x}}$ and $\hat{\mathbf{P}}$ are the predicted state vector and state error covariance matrix, respectively. \mathbf{x} and \mathbf{P} are the previously estimated state vector and state error covariance matrix, respectively.

4.2.3. Calculating vehicle positions. This step provides the measurements and the associated error standard deviation for the Kalman filter. We use a heuristic in which each vehicle patch is moved around its current location to cover as much as possible of the blobs related to this vehicle. More details can be found in [7].

4.2.4. Estimation. A measurement is a location in the image coordinate system as computed in the previous subsection, \mathbf{z} . Measurements are related to the state vector by

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t) + \mathbf{w}_t, \quad (5)$$

where \mathbf{h} is a non-linear measurement function and \mathbf{w}_t is a sequence of zero-mean, white, Gaussian measurement noise. The extended Kalman filter had to be used because \mathbf{h} is non-linear. The standard state estimation equations for the extended Kalman filter are applied.

4.2.5. Refinement. At the end of the cycle, we perform some checks to refine the vehicle-blob relationships since vehicles have been relocated. This step provides mechanisms for vehicle initialization and re-acquiring blobs due to occlusions. Vehicle initialization is done by scanning unlabeled blobs from left to right (or right to left depending on vehicles direction) and finding the first blob that satisfies a condition: *If a vehicle box is placed over the blob, at least 30% of the box will be covered by this blob and other unlabeled blobs.* The vehicle dynamics are initialized according to the composite velocity of all these blobs. Blobs can be acquired by vehicles if they happen to be covered enough (about 30%) by the vehicle.

5. Experimental results

The system was implemented on a dual Pentium 200MHz PC equipped with a C80 Matrox Genesis vision board. Tracking was achieved at 15 frames/second even for the most

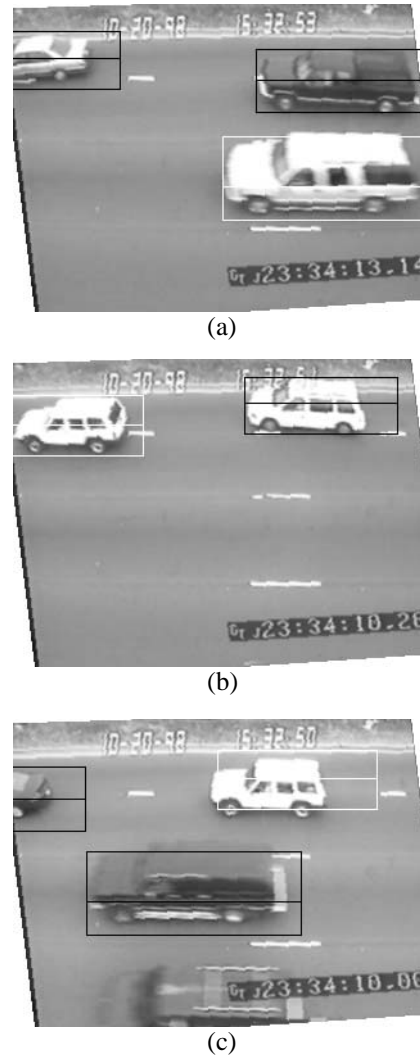


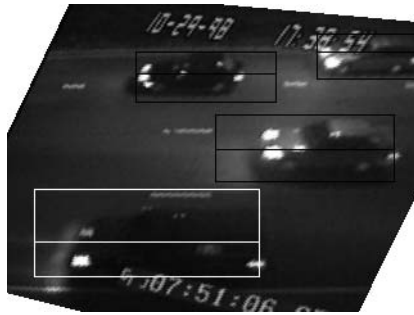
Figure 4. Three snapshots from a day sequence.

cluttered scenes. Several weaving sequences were used to test the system. Variations included different lighting conditions, camera view angles, and vehicle speeds (e.g., rush hour). The system was able to track most vehicles successfully and provided accurate trajectory information. Figures 4 and 5 show some snapshots of the tracking process performed on scenes with two different camera view angles and during different times of the day. One drawback of the current implementation is the use of a fixed size vehicle model for all vehicles. Even though this model works for most vehicles, it does not correctly track very large vehicles. Figure 6 shows how a large vehicle is tracked as two separate vehicles. Another drawback is present in the feature extraction component of the system. Large shadows are extracted as features and can confuse the tracking process. We are currently working on ways to deal with these two problems.

By providing the location of lane boundaries, the system was used to do data collection for use by the Minnesota Department of Transportation. The data included the count and average speed of vehicles in each lane as well as vehicles



(a)



(b)

Figure 5. Two snapshots from a night sequence.

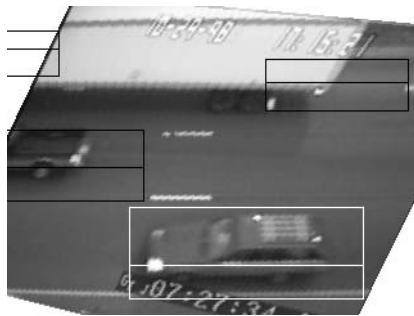


Figure 6. A large vehicle tracked as two vehicles.

that move between the two lanes farthest from the camera (weaving vehicles).

The data was provided accumulatively every 10, 30, 60, and 300 seconds. Sample data for the two lanes farthest from the camera is shown in Table 1. In the table, periods are measured in seconds and speed is measured in miles per hour. A

Time/ period	Vehicles no. (no. in period)	Lane 1->2 Spd (no.)	Lane 2->1 Spd (no.)	Lane 2->2 Spd (no.)
16:30:40 /10	1243 (15)	34.5 (3)	37.6 (2)	42.1 (3)
16:30:50 /10	1265 (22)	32.6 (1)	25.2 (3)	42.3 (4)
16:31:00 /10	1289 (23)	28.4 (4)	0 (0)	38.4 (5)
16:31:00 /30	1289 (60)	31.2 (8)	30.2 (5)	40.6 (12)
16:31:00 /60	1289 (110)	28.8 (18)	29.2 (7)	38.3 (23)

Table 1. Sample output.

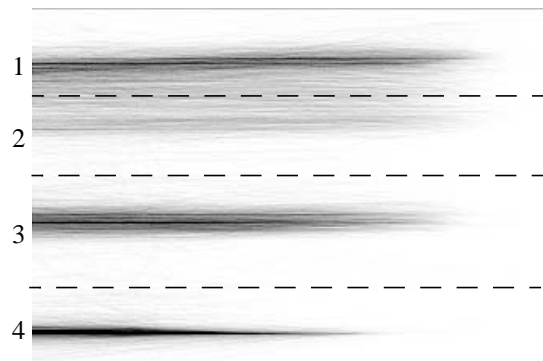


Figure 7. A visualization of vehicle trajectories on four lanes over a period of one hour.

visualization of vehicle trajectories accumulated over an hour of heavy traffic is shown in Figure 7. Notice the crisp vehicle paths for the two lanes closer to the camera. Also notice the gray area between the other two lanes which shows that a great deal of weaving took place. The reason lane 4 trail appears shorter is that the system starts the tracking a bit later than other lanes because vehicles on this lane move faster in the image.

6. Conclusions

We presented a real-time model-based vehicle tracking system capable of working robustly under many difficult circumstances. The main goal of the system is to provide data on weaving sections. For each vehicle in the view of the camera, the system produces location and velocity information as long as the vehicle is visible. There are some issues that still need to be addressed. Dealing with large shadows and large vehicles are two such issues.

Acknowledgements

This work has been supported by the Minnesota Department of Transportation through Contract MNDOT/74708-W.O. #14, the Center for Transportation Studies through Contract #USDOT/DTRS 93-G-0017-01, and the National Science Foundation through Contracts #IRI-9410003 and #IRI-9502245.

References

- [1] K.D. Baker and G.D. Sullivan, "Performance assessment of model-based tracking," in *Proc. of the IEEE Workshop on Applications of Computer Vision*, pp. 28-35, Palm Springs, CA, 1992.
- [2] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik, "A real-time computer vision system for measuring traffic parameters," in *Proc. of IEEE Conf. Computer Vision and Pattern Recognition*, pp. 496-501, June 1997, Puerto Rico.
- [3] G. Halevi and D. Weinshall, "Motion of disturbances: detection and tracking of multi-body non-rigid motion," in *Proc. of IEEE Conf. Computer Vision and Pattern Recognition*, pp. 897-902, June 1997, Puerto Rico.
- [4] K.P. Karmann and A. Brandt, "Moving object recognition using an adaptive background memory," V. Cappellini, editor, *Time-Varying Image Processing and Moving Object Recognition*, Amsterdam, Netherland, 1990.
- [5] M. Kilger, "A shadow handler in a video-based real-time traffic monitoring system," in *Proc. of IEEE Workshop on Applications of Computer Vision*, pp. 1060-1066, Palm Springs, CA, 1992.
- [6] D. Koller, J. Weber, and J. Malik, "Robust multiple car tracking with occlusion reasoning," in *ECCV*, Stockholm, Sweden, 1994.
- [7] O. Masoud and N. P. Papanikolopoulos, "A robust real-time multi-level model-based pedestrian tracking system," in *Proc. of the ITS America Seventh Annual Meeting*, Washington, DC, Jun 1997.
- [8] S.M. Smith and J.M. Brady, "ASSET-2: real-time motion segmentation and shape tracking," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(8):814--820, 1995.