

# The Use of Computer Vision in Monitoring Weaving Sections

Osama Masoud, Nikolaos P. Papanikolopoulos, Eil Kwon

## **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.

## **Index Terms--**

### I. 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 exists (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 [18,25] 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.

A large body of vision research has been targeted at vehicle tracking. One popular research direction deals with three-dimensional tracking. Three-dimensional tracking uses models for vehicles and aims to handle complex traffic situations and arbitrary configurations. A suitable application would be conceptual descriptions of traffic situations [8]. Robustness is more important than computational efficiency in such applications. Kollnig and Nagel [14] developed a model-based system. They proposed to use image gradients instead of edges for pose estimation. They fitted image gradients to synthetic model gradients which have been projected on the image. A Kalman filter was used to stabilize the tracking. Optical flow was used to generate object candidates at the initialization stage only. The selection of the model was done manually for each vehicle since the emphasis was on robust tracking as opposed to classification. In [13], the same authors increased robustness by utilizing optical

flow during the tracking process as well. Nagel *et al.* [19] and Leuck and Nagel [15] extended the previous approach to estimate the steering angle of vehicles. This was a necessary extension to handle trucks with trailers which were represented as multiple linked rigid polyhedra [19]. Experimental results in [15] compared the steering angle and velocity of a vehicle to ground truth showing good performance. They also provided qualitative results for other vehicles showing an average success rate of 77%. Tracking a single vehicle took 2-3 seconds per frame.

Three-dimensional tracking of vehicles has been extensively studied by the research group at the University of Reading. Baker and Sullivan [2] and Sullivan [21] utilized knowledge about the scene in their 3D model-based tracking. This includes knowledge that the vehicles move on a plane, the camera calibration, the vehicles, and their dynamics. The ground plane assumption reduces the tracking search parameters to a translation and a rotation. The matching is also done by comparing the projection of the model to features in the image. Sullivan *et al.* [23] extended this approach so that image features act as forces on the model. This reduced the number of iterations and improved performance. They also parameterized models as deformable templates and used principal component analysis to reduce the number of parameters. A filter that was used by Maybank *et al.* [16] to stabilize tracking was demonstrated to surpass the extended Kalman filter when vehicles undergo complex motion such as a three point turn.

Three-dimensional methods are sensitive to initial pose estimate. Some interesting work on extracting initial pose from static images was done in [24]. In addition, pose estimation and tracking can be improved by using more refined models [14] which would require a large set of models and therefore more computational requirements.

In the case of applications that require monitoring of controlled traffic situations, it is usually possible to implement a much more efficient, yet still robust, system. Sullivan *et al.* [22] developed a simplified version of their model-based tracking approach to achieve real-time performance. The method was not designed to track vehicles across lanes, however.

Another tracking approach is feature-based. Features, such as corners and edges, are tracked and grouped based on their spatial and temporal characteristics. This approach is robust with respect to occlusions. However, the density of reliable features may not be high enough in certain circumstances. If the application requires isolation of vehicles (e.g., a classification application), there may be a need for an additional component to the tracking system. Feature tracking is considered a computationally expensive operation due to the use of correlation. Smith [20] used custom hardware to achieve tracking at 25Hz. Beymer *et al.* [3] used 13 C40 DSPs to achieve up to 7.5Hz tracking in uncongested traffic. This may not remain a big disadvantage in the long run, however, considering the constant increase in computing power of ordinary PCs. Active deformable contours were also used to track vehicles [12]. However, initialization can be problematic especially for partially occluded vehicles. Combining (or switching among) different approaches has been considered as a way of dealing with diverse weather and lighting conditions.

Some algorithms have been developed to quantify scene illumination conditions (well lit, poorly lit, has shadows, etc.)[26] and road conditions (wet, dry, snow, etc.)[28]. A method for combining multiple algorithms was given in [7].

Finally, some researchers tracked connected regions in the image (blobs) [10,11,4,5]. Region-based tracking is perhaps the most computationally efficient but in general suffers from segmentation problems in cluttered scenes. This happens because blobs of several vehicles may merge into a single blob. Alternatively, a single vehicle may be composed of several blobs. In [4], the authors use a rule-based system to refine the tracking and resolve low-level segmentation errors. The system we describe in this paper is region-based. Our method differs from other region-based methods by the way it handles relating blobs to vehicles. This relation is allowed to be “many-to-many”, and is updated iteratively depending on the observed blobs' behavior and predictions of vehicles' behavior. Processing is done at three levels (this structure is based on our earlier work on pedestrian tracking [17]). Features are obtained at the 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.

## II. FEATURES LEVEL

Features are extracted using a recursive filtering technique (a slightly modified version of Halevi and Weinshall's algorithm [9]). 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  $\alpha$  is a fraction in the range 0 to 1 (typically 0.5). The feature image at time  $i$ ,  $F_i$ , is computed as follows:  $F = |M_i - I_i|$ . 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 it 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). Since 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 6(b) is the warped version of Figure 2) and then perform feature extraction. Apart from camera pan, rotated blobs can appear due to the diagonal motion of a weaving vehicle. However, this rotation is small and does not affect our algorithm since it occurs over a long distance compared to lane width. The warping

process uses extrinsic camera parameters such as location and orientation to perform inverse perspective projection. These parameters are estimated interactively to a high precision at initialization by utilizing ground truth measurements in the scene (such as lane width, lane marking length, etc.). The tracking algorithm is not sensitive to errors in these parameters. However, since these parameters are also used at the vehicles level to provide speed and location information of vehicles, precise estimation of these parameters is desired. In [27], a simple interface has been developed which requires minimal operator interaction and leads to good precision estimates.

### III. 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.

#### A. 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.  $E_i = \phi$ , 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  $D_i = \bigcup_{u \in P_i} N_i(u)$ .  $P_i$  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  $S_i(u) = \sum_{v \in P_i} A(v)$  be the total area occupied by the neighbors of  $u$ . The cost function that we use penalizes graphs in which blobs change significantly in size. A perfect match would be one in which blob sizes remain constant (e.g., the size of a blob that splits equals to the sum of the sizes of blobs it split into). We now write the formula for the cost function as

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

This function is a summation of ratios of size change over all parent blobs.

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. The computational complexity of this step is highly dependent on the graph being considered. If the graph already satisfies the parent structure constraint, it is  $O(1)$ . On the other

hand, if we have a fully connected graph, the complexity is exponential in the number of vertices (bounded by  $2^m$ ). Fortunately, because of the locality constraint and the high frame rate, the majority of graphs considered already satisfy the parent structure constraint. Occasionally, a small cluster of the graph may not satisfy the parent structure constraint and the algorithm will need to enumerate a few graphs. In practice, the algorithm never took more than a few milliseconds to execute even in the most cluttered scenes. Other techniques to find the optimum (or near optimum) graph (e.g., stochastic relaxation using simulated annealing) can also be used. The main concern, however, would be their efficiency which may not be appropriate for this real-time application due to their iterative nature.

At the end of this stage, we use a simple method to calculate the velocity of each blob,  $v$ , based on the velocities of the blobs at the previous stage and the computed blob graph. The blob velocity will be used to initialize vehicle models as described later. If  $v$  is the outcome of a splitting operation, it will be assigned the same velocity as the parent blob. If  $v$  is the outcome of a merging operation, it will be assigned the velocity of the largest child blob. If  $v$  is a new blob, it will be assigned zero velocity. Finally, if there is only one blob,  $u$ , related to  $v$ , the velocity is computed as

$$\mathbf{V}(v) = \beta \frac{(\mathbf{b}_v - \mathbf{b}_u)}{\delta t} + (1 - \beta)\mathbf{V}(u) \quad (3)$$

where  $\mathbf{b}_v$  and  $\mathbf{b}_u$  are the centers of the bounding boxes of  $v$  and  $u$ , respectively,  $\beta$  is a weight factor set to 0.5 (found empirically), and  $\delta t$  is the sampling interval since the last stage.

#### IV. 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. In our system we use the Extended Kalman Filter (EKF) in the tracking process. We now present this model in more detail and then describe how tracking is performed.

##### A. 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:

$$\mathbf{x}_{t+1} = \mathbf{F}\mathbf{x}_t + \mathbf{v}_t, \quad (4)$$

where the subscripts indicate time,  $\mathbf{x} = [x \ \dot{x} \ y \ \dot{y}]^T$  is the state vector consisting of the vehicle location,  $(x, y)$  and velocity,

$(\dot{x}, \dot{y})$ ,  $\mathbf{F}$  is the transition matrix of the system given by 
$$\begin{bmatrix} 1 & \delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
, and  $\mathbf{v}_t$  is a sequence of zero-mean, white, Gaussian

process noise with covariance matrix  $\mathbf{Q}$ . We compute  $\mathbf{Q}$  as in [1] (page 84) to become  $\begin{bmatrix} \mathbf{A} & 0 \\ 0 & \mathbf{A} \end{bmatrix} q$  where  $\mathbf{A} = \begin{bmatrix} \frac{(\delta t)^3}{3} & \frac{(\delta t)^2}{2} \\ \frac{(\delta t)^2}{2} & \delta t \end{bmatrix}$

and represents the variance of the acceleration.

### B. Vehicle Tracking

The next five subsections describe one tracking cycle.

#### 1) Relating vehicles to blobs

We represent the relationship between vehicles and blobs as a directed bipartite graph,  $GP_i(VP_i, EP_i)$ , where  $VP_i = B_i \cup P$ .  $B_i$  is the set of blobs computed from the  $i$ th image as in Section III.A.  $P$  is the set of vehicles. An edge  $(p, u)$ ,  $p \in P$  and  $u \in B_i$  denotes that blob  $u$  participates in vehicle  $p$ . We call  $GP_i$  a vehicle graph. Given a blob graph,  $G_i(V_i, E_i)$ , as computed in Section III.A, and a vehicle graph,  $GP_{i-1}$ ,  $EP_i$  is computed as follows:

$$EP_i = \{(p, u) \mid (u, v) \in E_i \wedge (p, v) \in EP_{i-1}\}. \quad (5)$$

In other words, 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.

#### 2) Prediction

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

$$\begin{aligned} \hat{\mathbf{x}}_{t+1} &= \mathbf{F}\mathbf{x}_t, \\ \hat{\mathbf{P}}_{t+1} &= \mathbf{F}\mathbf{P}_t\mathbf{F}^T + \mathbf{Q}. \end{aligned} \quad (6)$$

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.

### 3) Calculating vehicle positions

In this step, we use the predicted vehicle locations as starting positions and we apply the following rule to update the locations of a vehicle: Move each vehicle,  $p$ , as little as possible so that it covers as much as possible of its blobs,  $\{u \mid (p, u) \in EP_i\}$ ; and if a number of vehicles share some blobs, they should all participate in covering all these blobs. The amount by which a vehicle covers a blob implies a measure of overlap area. We have already used the bounding box as a shape representation of blobs. However, since the blob bounding box area may be quite different from the actual blob area, we will include the blob density as computed in Section III in the computation of the vehicle-blob overlap area. Let  $BB(p)$  be the bounding box of a vehicle  $p$ , and  $BB(b)$  be the bounding box of a blob,  $b$ . The intersection of  $BB(p)$  and  $BB(b)$  is a rectangle which we denote its area as  $X(BB(p), BB(b))$ . The overlap area between  $p$  and  $b$  is computed as  $X(BB(p), BB(b)) \times D(b)$ . When more than one vehicle share a blob, the overlap area is computed this way for each vehicle only if the other vehicles do not also overlap the intersection area. If they did, that particular overlap area is divided by the number of vehicles whose boxes overlap the area. Figure 4

illustrates this situation. The overlap area for  $p_1$  is computed as  $a \times D(b_1) + b \times D(b_2) + \frac{c \times D(b_2)}{2}$ . For  $p_2$ , the overlap area is

$$d \times D(b_2) + \frac{c \times D(b_2)}{2}.$$

The problem of finding the optimum locations of vehicles can be stated in terms of the overlap area measure that we just defined. We would like to place vehicles such that the total overlap area of each vehicle is maximized. We restate the optimization problem as the problem of finding the minimum total overlap area arrangement of vehicles which has the least distances between old and new locations of the vehicle. We do not attempt to solve the problem optimally because of its complexity. Instead, we resort to a heuristic solution using relaxation. First, a large step size is chosen. Then, each vehicle is moved in all possible directions by the step size and the location which minimizes the overlap area is recorded. Vehicle locations are then updated according to the recorded locations. This completes one iteration. In each following iteration, the step size is decreased. In our implementation, we start with a step of 64 pixels and halve the step size in each iteration until a step size of 1 pixel is reached. The resulting locations form the measurements that will be fed back into the EKF to produce the new state estimates. Moreover, we use the overlap area to provide feedback about the measurement confidence by setting the measurement error standard deviation, which is described below, to be inversely proportional to the ratio of the overlap area to the vehicle area. That is, the smaller the overlap area, the less reliable the measurement is considered.

### 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

$$\mathbf{z}_t = \mathbf{h}(\mathbf{x}_t) + \mathbf{w}_t, \quad (7)$$

where  $\mathbf{h}$  is a non-linear measurement function (the inverse perspective projection function) and  $\mathbf{w}_t$  is a sequence of zero-mean,

white, Gaussian measurement noise with covariance  $\mathbf{R}_t$  given by  $\begin{bmatrix} \sigma_t^2 & 0 \\ 0 & \sigma_t^2 \end{bmatrix}$ . The measurement error standard deviation,  $\sigma_t$ ,

depends on the overlap area computed in the previous section. We let  $\mathbf{H}$  be the Jacobian of  $\mathbf{h}$ . The EKF state estimation equations become

$$\begin{aligned} \mathbf{K}_{t+1} &= \hat{\mathbf{P}}_{t+1} \mathbf{H}^T (\mathbf{H} \hat{\mathbf{P}}_{t+1} \mathbf{H}^T + \mathbf{R}_t)^{-1}, \\ \mathbf{x}_{t+1} &= \hat{\mathbf{x}}_{t+1} + \mathbf{K}_{t+1} (\mathbf{z}_{t+1} - \mathbf{h}(\hat{\mathbf{x}}_{t+1})), \\ \mathbf{P}_{t+1} &= (\mathbf{I} - \mathbf{K}_{t+1} \mathbf{H}) \hat{\mathbf{P}}_{t+1}. \end{aligned} \quad (8)$$

The estimated state vector  $\mathbf{x}_{t+1}$  is the outcome of the vehicles level.

### 5) Refinement

At the end of this stage, we perform some checks to refine the vehicle-blob relationships since vehicles have been relocated.

These can be summarized as follows:

- a. If the overlap area between a vehicle and one of its blobs becomes less than 10% of the size of both, it will no longer be considered belonging to this vehicle. This serves as a splitting procedure when one vehicle passes another.
- b. If the overlap area between a vehicle and a blob that does not belong to any vehicles becomes more than 10% of the size of either one, the blob will be added to the vehicle blobs. This makes the vehicle re-acquire some blobs that may have disappeared due to occlusion.
- c. If a cluster of blobs is found which are not related to any vehicles and whose age is larger than a threshold (i.e., have been successfully tracked for a certain number of frames), we do the following: A new vehicle may be initialized only if it will be more than 30% covered by the blobs cluster. The vehicle is given an initial velocity equal to the average of the blobs velocities. This serves as the initialization step. The requirement on the age helps in reducing chances of unstable blobs being used to initialize vehicles.
- d. Select one of the blobs which is already assigned one or more vehicles but can accommodate more vehicle patches. Create a new vehicle for this blob as in c. This handles cases in which a more than one vehicle appear in the scene while forming one big blob which does not split. If we do not do this step, only one vehicle would be assigned to this blob.
- e. If a vehicle leaves the view of the camera or has not been covered by any blobs for an extended period of time, the vehicle is deleted.

## V. 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 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 (average accuracy 85%) and provided accurate trajectory information. Ground truth was established by using manual counters. The system dealt well with partial occlusions. The success is partly due to Kalman filtering since the measurement error standard deviation is set to reflect confidence in the found features (see Section IV.B.3). However, there were some failures especially with large trucks which, depending on the camera view, may completely occlude other vehicles. Figures 5 and 6 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 7 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. An interesting probabilistic shadow handling approach is given in [6]. Wixson *et al.* [26] proposed using different parameters (or in some cases different algorithms) for different scene illumination conditions and provided an elegant condition assessment method.

By providing the location of lane boundaries, our 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 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 are shown in Table 1. In the table, periods are measured in seconds and speed is measured in miles per hour. A visualization of vehicle trajectories accumulated over an hour of heavy traffic is shown in Figure 8. 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.

## VI. 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.

## References

- [1] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*, Academic Press, 1988.
- [2] 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.
- [3] 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.
- [4] R. Cucchiara, M. Piccardi, and P. Mello, "Image analysis and rule-based reasoning for a traffic monitoring system," in *Proc. IEEE Conference on Intelligent Transportation Systems*, pp. 758-763, Tokyo, Japan, October 1999.
- [5] D. Dailey and L. Li, "Algorithm to estimate vehicle speed using un-calibrated cameras," in *Proc. IEEE Conference on Intelligent Transportation Systems*, pp. 441-446, Tokyo, Japan, October 1999.
- [6] N. Friedman, S. Russell, "Image segmentation in video sequences," in *Proc. of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, Providence, Rhode Island, 1997.
- [7] S. Gil, R. Milanese, and T. Pun, "Combining multiple motion estimates for vehicle tracking," in *Proc. of the Fourth European Conference on Computer Vision*, vol. 2, pp. 307-320, Cambridge, UK, April 1996.
- [8] M. Haag and H.-H. Nagel, "Incremental recognition of traffic situations from video image sequences," *Image and Vision Computing*, vol. 18, pp. 137-153, 2000.
- [9] 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.
- [10] 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, Netherlands, 1990.
- [11] 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.
- [12] D. Koller, J. Weber, and J. Malik, "Robust multiple car tracking with occlusion reasoning," in *ECCV*, Stockholm, Sweden, 1994.
- [13] H. Kollnig and H.-H. Nagel, "Matching object models to segments from an optical flow field," in *Proc. of the Fourth European Conference on Computer Vision*, vol. 2, pp. 15-18, Cambridge, UK, April 1996.

- [14] H. Kollnig and H.-H. Nagel, "3D pose estimation by directly matching polyhedral models to gray value gradients," *International Journal of Computer Vision*, vol. 23, no. 3, pp. 283-302, 1997.
- [15] H. Leuck, and H.-H. Nagel, "Automatic differentiation facilitates OF-integration into steering-angle-based road vehicle tracking," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 360-365, Fort Collins, CO, June 1999.
- [16] S.J. Maybank, A.D. Worrall, and G.D. Sullivan, "Filter for car tracking based on acceleration and steering angle," in *Proc. of the Seventh British Machine Vision Conference (BMVC'96)*, vol. 2, pp. 615-624, Edinburgh, England, September 1996.
- [17] 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, June 1997.
- [18] P.G. Michalopoulos, "Vehicle detection video through image processing: The autoscope system," *IEEE Transactions on Vehicular Technology*, vol. 40, pp. 21-29, 1991.
- [19] H.-H. Nagel, T. Schwarz, H. Leuck, and M. Haag, "T3wT: tracking turning trucks with trailers," in *Proc. of the IEEE Workshop on Visual Surveillance*, pp. 65-72, Bombay, India, January 1998.
- [20] 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.
- [21] G.D. Sullivan, "Model-based vision for traffic scenes using the ground-plane constraint," *Phil. Trans. Roy. Soc. (B)*, 337, pp. 361-370, 1992.
- [22] G.D. Sullivan, K.D. Baker, A.D. Worrall, C.I. Attwood, and P.M. Remagnino, "Model-based vehicle detection and classification using orthographic approximations," *Image & Vision Computing*, vol. 15, no. 8, pp. 649-654, Aug. 1997.
- [23] G.D. Sullivan, A.D. Worrall, and J.M. Ferryman, "Visual Object Recognition Using Deformable Models of Vehicles," in *Proc. Workshop on Context-Based Vision*, pp. 75-86, Cambridge Massachusetts, June 1995.
- [24] T.N. Tan, G.D. Sullivan, and K.D. Baker, "Model-based localization and recognition of road vehicles," *International Journal of Computer Vision*, vol. 27, no. 1, pp. 5-25, 1998.
- [25] J. Versavel, "Road safety through video detection," in *Proc. IEEE Conference on Intelligent Transportation Systems*, pp. 753-757, Tokyo, Japan, October 1999.
- [26] L. Wixson, K. Hanna, and D. Mishra, "Improved illumination assessment for vision-based traffic monitoring," in *Proc. of the IEEE Workshop of Visual Surveillance*, pp. 34-41, Bombay, India, January 1998.
- [27] A.D. Worrall, G.D. Sullivan, and K.D. Baker, "A simple, intuitive camera calibration tool for natural images," in *Proc. of the 5th British Machine Vision Conference*, pp. 781-790, 1994.

[28] M. Yamada, K. Ueda, I. Horiba, and N. Sugie, "Discrimination of the road condition towards understanding of vehicle driving environments," in Proc. IEEE Conference on Intelligent Transportation Systems, pp. 20-24, Tokyo, Japan, October 1999.

# List of Figures

Figure 1. Feature image (corresponds to the image in Figure 5(c)).

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.

Figure 4. Overlap area. Vehicles  $p_1$  and  $p_2$  share blob  $b_2$  while  $b_1$  is only part of  $p_1$  (See Section IV.B.3. for overlap area computation).

Figure 5. Three snapshots from a day sequence.

Figure 6. Two snapshots from a night sequence.

Figure 7. A large vehicle tracked as two vehicles.

Figure 8. A visualization of vehicle trajectories on four lanes over a period of one hour.

# List of Tables

Table 1. Sample output.

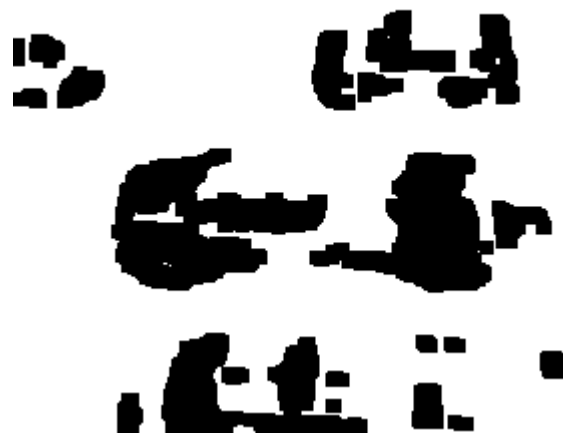


Figure 1. Feature image (corresponds to the image in Figure 5(c)).



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

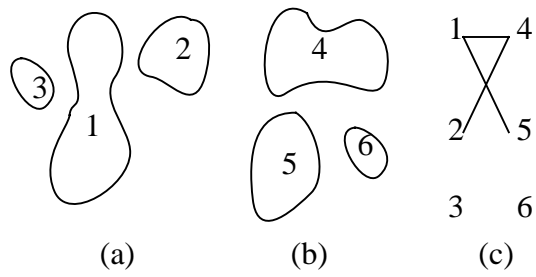


Figure 3. (a) Blobs in frame  $(i-1)$ . (b) Blobs in frame  $i$ . (c) Relationship among blobs.

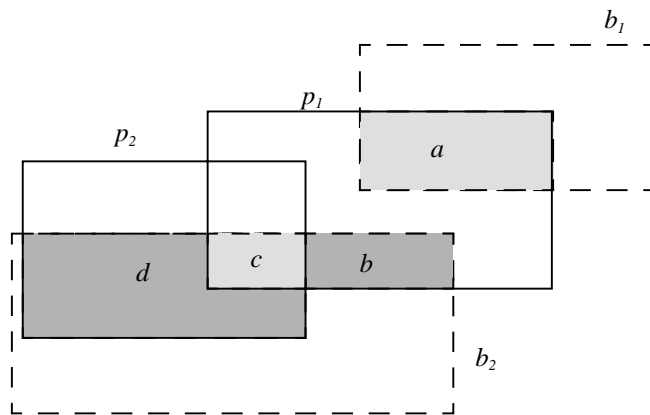


Figure 4. Overlap area. Vehicles  $p_1$  and  $p_2$  share blob  $b_2$  while  $b_1$  is only part of  $p_1$  (See Section IV.B.3. for overlap area computation).



(a)



(b)

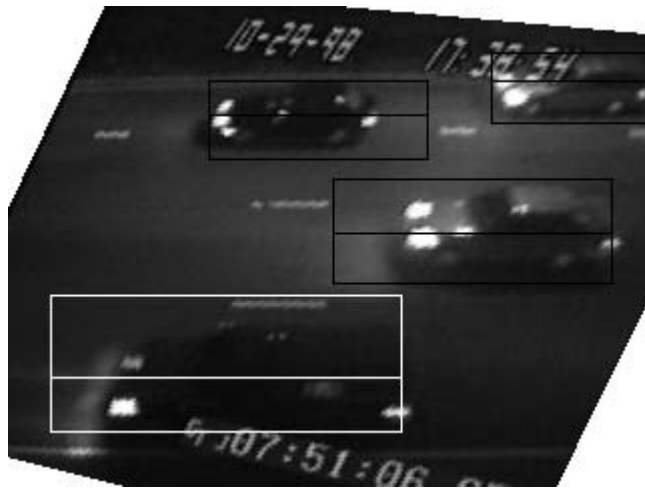


(c)

Figure 5. Three snapshots from a day sequence.



(a)

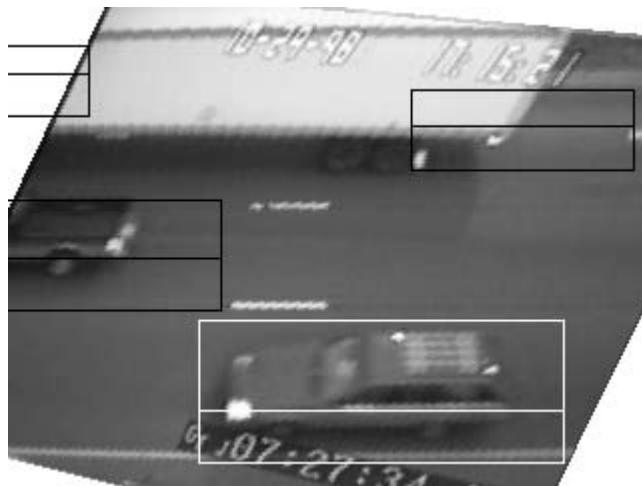


(b)

Figure 6. Two snapshots from a night sequence.



(a)



(b)

Figure 7. A large vehicle tracked as two vehicles.

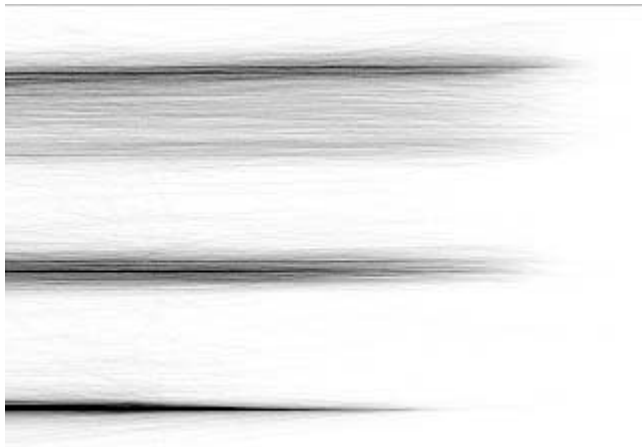


Figure 8. A visualization of vehicle trajectories on four lanes over a period of one hour.

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.

# Biosketches

**Osama Masoud** was born in Riyadh, Saudi Arabia in 1971. He received his B.S. and M.S. degrees in computer science from King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia, in 1992 and 1994, respectively, and his Ph.D. degree in computer science from the University of Minnesota, Minneapolis, MN, in 2000. He is currently a post-doctoral associate at the department of Computer Science at the University of Minnesota. His research interests include computer vision, robotics, transportation applications, and computer graphics. He is a recipient of a Research Contribution Award from the University of Minnesota, the Rosemount Instrumentation Award from Rosemount Inc., and the Matt Huber Award for Excellence in Transportation Research.

**Nikolaos P. Papanikolopoulos** (S'88-M'93) was born in Piraeus, Greece, in 1964. He received the Diploma degree in electrical and computer engineering from the National Technical University of Athens, Athens, Greece, in 1987, the M.S.E.E. in electrical engineering from Carnegie Mellon University (CMU), Pittsburgh, PA, in 1988, and the Ph.D. in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, in 1992. Currently, he is an Associate Professor in the department of Computer Science at the University of Minnesota. His research interests include robotics, sensors for transportation applications, control, and computer vision. He has authored or co-authored more than 100 journal and conference papers in the above areas (twenty nine refereed journal papers). He was finalist for the Anton Philips Award for Best Student Paper in the 1991 IEEE Robotics and Automation Conference. Furthermore, he was recipient of the Kritski fellowship in 1986 and 1987. He was a McKnight Land-Grant Professor at the University of Minnesota for the period 1995-1997 and has received the NSF Research Initiation and Early Career Development Awards. Finally, he has received grants from DARPA, Sandia National Laboratories, NSF, USDOT, MN/DOT, Honeywell, and 3M.

**Eil Kwon** was born in Seoul, Korea, in 1953. He received his B.S. degree in Civil Engineering from SungKyunKwan University, Seoul, Korea, in 1978 and the M.S.C.E. in transportation engineering from University of Minnesota, Minneapolis, MN, in 1986 and the Ph.D. in transportation engineering from University of Minnesota, Minneapolis, MN, in 1990. He is currently the advanced traffic systems program director and an adjunct professor in the Center for Transportation Studies, University of Minnesota. His research interests include traffic flow modeling/simulation, adaptive control of traffic flows at freeways/ intersections and on-line operational strategies for traffic management. He has authored or co-authored 30 journal and conference papers in the above areas (fifteen refereed journal papers). He was a 3M McKnight distinguished visiting professor at the University of Minnesota, Duluth, in 2000.