Automatic Symbolic Traffic Scene Analysis Using Belief Networks*

T. Huang, D. Koller, J. Malik, G. Ogasawara, B. Rao, S. Russell, and J. Weber
Computer Science Division
University of California
Berkeley, CA 94720
{tthuang|koller|malik|ogasawara|bobbyrao|russell|jweber}@cs.berkeley.edu

Abstract

Automatic symbolic traffic scene analysis is essential to many areas of IVHS (Intelligent Vehicle Highway Systems). Traffic scene information can be used to optimize traffic flow during busy periods, identify stalled vehicles and accidents, and aid the decision-making of an autonomous vehicle controller. Improvements in technologies for machine vision-based surveillance and high-level symbolic reasoning have enabled us to develop a system for detailed, reliable traffic scene analysis. The machine vision component of our system employs a contour tracker and an affine motion model based on Kalman filters to extract vehicle trajectories over a sequence of traffic scene images. The symbolic reasoning component uses a dynamic belief network to make inferences about traffic events such as vehicle lane changes and stalls. In this paper, we discuss the key tasks of the vision and reasoning components as well as their integration into a working prototype.

Introduction

An important task for progress in IVHS (Intelligent Vehicle Highway Systems) is the development of methods for automatic traffic scene analysis. All three major applications of IVHS – ATIS (Advanced Traveler Information Systems), ATMS (Advanced Traffic Management Systems), and AVCS (Automated Vehicle Control Systems) – could benefit from accurate, high-level descriptions of traffic situations. For example, an ATIS and an ATMS could use information about traffic congestion and stalls to warn drivers or to direct vehicles to alternate routes. An ATMS also could analyze local traffic at intersections to identify those with higher risk of accidents. Finally, an AVCS would need information about the actions of neighboring vehicles and the condition of traffic lanes ahead to control an automated car moving along a freeway (Niehaus & Stengel 1991).

*This work was supported by the California Department of Transportation under the PATH project grant MOU-83.

In this paper, we describe a prototype system in which we have successfully combined a robust, vision-based traffic surveillance system (Koller, Weber, & Malik 1994) with a dynamic belief network dedicated to analyzing traffic scenes. Unlike conventional loop detectors, which are buried underneath highways to count vehicles, video monitoring systems are less disruptive and less costly to install. They also have greater range and allow for more detailed descriptions of traffic situations. Dynamic belief networks provide a flexible, theoretically sound framework for traffic scene analysis because they can easily model uncertainty and because they can provide high-level, symbolic descriptions by integrating low-level information from a variety of sources. They also provide a natural framework for expressing knowledge about typical traffic behavior, allowing more accurate analyses from a given sensor stream.

Symbolic traffic scene analysis using vision-based surveillance systems has been previously investigated by several research groups (Schirra *et al.* 1987; Koller, Heinze, & Nagel 1991; Heinze, Krüger, & Nagel 1991; Huang, Ogasawara, & Russell 1993). The challenges of this approach include identifying vehicles despite imprecise video data and changing lighting conditions, tracking individual vehicles despite their overlapping with each other, and efficiently providing high-level descriptions based on evidence accumulated over time. We have achieved improvements in performance, reliability, and accuracy by applying a new approach for detecting and tracking vehicles, by explicitly reasoning about vehicle occlusions (Koller, Weber, & Malik 1994), and by devising techniques for fast belief network update, localized reasoning, and flexible node semantics.

Low-Level Machine Vision-Based Surveillance

Our traffic surveillance system is based on the block diagram shown in Figure 1. This section focuses on the tasks of feature extraction and tracking, and the next section focuses on the tasks of symbolic reasoning and incident detection.

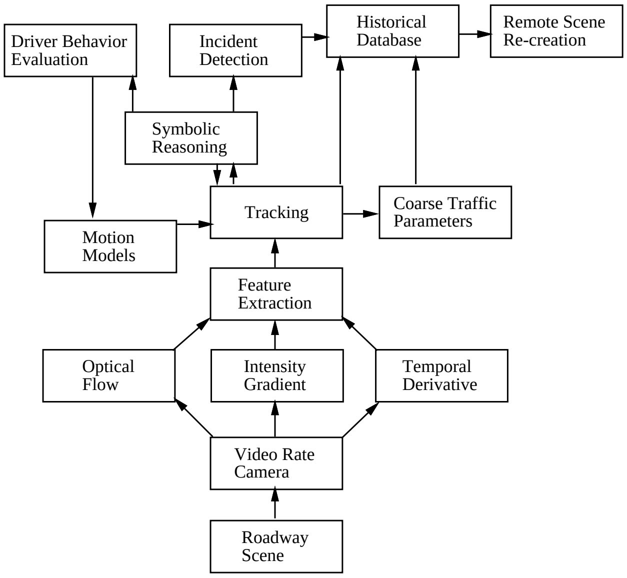

Figure 1: Block diagram of the complete traffic surveillance system. Arrows denote the flow of information.

As Figure 1 indicates, traffic scene analysis generally proceeds from low-level processing of road traffic images to high-level descriptions of the traffic situation (which can in turn be used to direct and disambiguate low-level processing). Given a sequence of traffic images, a vision-based surveillance system must identify the vehicles in the scene and track them as they progress along the image sequence. This requires not only estimation of the moving vehicle shapes and positions, but also association of these estimates from one image to the next.

Two primary factors that complicate this task are noisy sensors, which yield imprecise measurements, and vehicle occlusions, which make it more difficult to identify and disambiguate vehicles. To address these problems, we employ vehicle and motion models that are updated in a Kalman filter formalism, thus yielding most likely estimates based on accumulated observations.

Motion Segmentation

A surveillance system initiates vehicle identification and tracking by determining what parts of each image belong to moving objects and what parts belong to the background. This is accomplished by examining the difference in pixel intensities between each new frame and an estimate of the stationary background. Reliable background estimation, which is critical for accurate identification of moving 'blobs', is made more difficult as lighting conditions change. We perform this *initialization* step by using a modified version of the moving object segmentation method suggested by (Karmann & von Brandt 1990) and implemented by (Kilger 1992). Our method employs a Kalman filter-based adaptive background model. This allows the background estimate to evolve as the weather and time of day affect lighting conditions. The background is updated at each frame using the following update equation:

$$B_{t+1} = B_t + (\alpha_1(1 - M_t) + \alpha_2 M_t)D_t \qquad (1)$$

B_t is the background model at time t, D_t is the difference between the present frame and the background model, and M_t is a binary mask of hypothesized moving objects in the current frame. The gains α_1 and α_2 are based on estimates of the rate of change of the background. For a complete description, we refer the reader to (Koller, Weber, & Malik 1993).

Vehicle Identification and Shape Estimation

After identifying moving blobs, the vision system attempts to disambiguate individual vehicles and estimate their shapes. This helps with associating data over a sequence of images and with obtaining accurate vehicle trajectories. Our system performs these tasks by extracting closed contours enclosing each moving blob in each image. Contour extraction is based on motion and gray-value boundaries, which are obtained by thresholding the spatial image gradients and the time derivatives of the images. For each moving blob, points that pass a threshold test are enclosed by convex polygons, and these are used as initial object descriptions. The top row of Figure 2 shows an image section with a car, the detected moving object patch corresponding to the image of the car, and the sample points made up of image locations with acceptable spatial gradients and time derivatives. The convex polygon enclosing all these sample points is shown in the bottom row.

Our time-recursive shape estimation algorithm (Koller, Weber, & Malik 1993) cannot use convex polygons, since the number of vertices for a vehicle may change along an image sequence. We address this problem by using *snakes*, spline approximations to contours (Kass, Witkin, & Terzopoulos 1988; Curwen & Blake 1992). We use closed cubic splines with 12 *control points* to approximate each extracted convex polygon, and we obtain the locations of the control points by again employing a Kalman filter (Bartels, Beatty, & Barsky 1987; Koller, Weber, & Malik 1994). The bottom right image shows the spline approximation of the shape. Other examples of spline approximations can be found in Figure 6.

Motion Estimation

The final task of the video system is to track identified vehicles from one frame to the next. To accomplish this, we estimate vehicle motion with an affine motion model. For a sufficiently small field of view and for independently moving objects, the image velocity field $\boldsymbol{u}(\boldsymbol{x})$ at some location $\boldsymbol{x}$ inside a detected image patch can be closely approximated by a linear

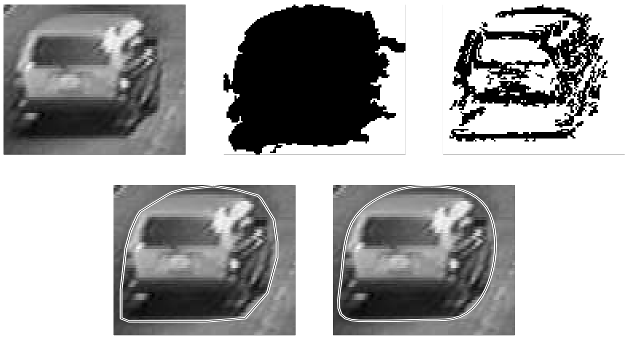

Figure 2: The top row shows an image section with a moving car, the moving object mask provided by the motion segmentation step, and the image locations with acceptable spatial gradients and temporal derivatives. The bottom row shows the convex polygon enclosing the sample points and the final contour description by cubic spline approximation of the polygon.

(affine) transformation. Since motion is constrained to the road plane and since possible rotation components along the normal of the plane are small, the degrees of freedom can be reduced to the extent that we obtain a velocity equation of only a scale parameter s and a displacement vector $\boldsymbol{u}_0$:

$$\boldsymbol{u}(\boldsymbol{x}) = s\,(\boldsymbol{x} - \boldsymbol{x}_m) + \boldsymbol{u}_0, \qquad (2)$$

For the scale parameter s, $s = 0$ indicates that there is no change in scale, while $s < 0$ and $s > 0$ indicate motion components along the optical axes away from and towards the camera, respectively. $\boldsymbol{x}_m$ denotes the center of the moving image region, and $\boldsymbol{u}_0$ denotes its displacement between two consecutive frames.

The affine motion parameters $\boldsymbol{\xi} = (\boldsymbol{u}, s)$ make up the state vector for motion estimation. We can use a third Kalman filter to estimate the motion parameters, since the measurement function can be expressed in a linear matrix equation. This tracker has been influenced by (Blake, Curwen, & Zisserman 1993), who successfully extended their real-time contour tracking system (Curwen & Blake 1992) by exploiting affine motion models. Complete details of the affine motion model can be found in (Koller, Weber, & Malik 1994).

Occlusion Reasoning

Because vehicles often overlap with each other in the road images, the extracted contours of vehicles will become distorted for some frames. This can cause artificial shifts in vehicle trajectories, since tracks are obtained by connecting centers of contours along the image sequence. To avoid these artificial shifts and to obtain reasonable tracks, we employ an explicit occlusion reasoning algorithm, which compensates for overlapping vehicles.

The occlusion reasoning algorithm works because the traffic scene geometry is known and because motion is assumed to be constrained to the ground plane (Koller, Weber, & Malik 1993). This knowledge makes it possible to determine a depth ordering among the objects in the scene, and this depth ordering defines the order in which objects are able to occlude each other.

High-Level Reasoning Using Belief Networks

We now address the task of using vehicle track information (e.g., their positions and velocities) to arrive at high-level symbolic descriptions of vehicles and the traffic scene. To accomplish this, our symbolic reasoner uses multiple, per-vehicle dynamic belief networks with fast rollup.

Concepts

Belief networks are directed acyclic graphs in which nodes represent random variables (usually discrete) and arcs represent causal connections among the variables (Pearl 1988). Associated with each node is a probability table that provides conditional probabilities of the node's possible states given each possible state of its parents. When values are observed for a subset of the nodes, posterior probability distributions can be computed for any of the remaining nodes. This updating takes place using a compiled form of the belief network that is more suitable to propagating the influence of evidence to other nodes.

Belief networks offer a mathematically sound basis for making inferences under uncertainty. The conditional probability tables provide a natural way to represent uncertain events, and the semantics of the updated probabilities are well-defined. Knowledge of causal relationships among variables is expressed by the presence or absence of arcs between them. Furthermore, the conditional independence relationships implied by the topology of the network allow exponentially fewer probabilities to be specified than the full joint probability distribution for all the variables in the network.

Dynamic belief networks allow for reasoning in domains where variables take on different values over time. Typically, observations are taken at regular 'time slices', and a given network structure is replicated for each slice. Nodes can be connected not only to other nodes within the same time slice but also to nodes in the previous or subsequent slice. As new slices are added to the network, older slices are removed. Before a slice is removed, its influence is 'rolled-up' into the next slice by recomputing probability tables for certain nodes in that slice. Thus, evidence accumulated over time is always integrated into the current belief network model (Nicholson 1992; Kjaerulff 1993).

Traffic network structure

The symbolic reasoning component for our system is built on the HUGIN inference engine for belief networks (Andersen *et al.* 1989). Figure 3 shows an example belief network fragment for a single vehicle. Figure 4 shows the fragment projected over one time slice. For each vehicle in a traffic scene, there is a separate belief network corresponding to it.

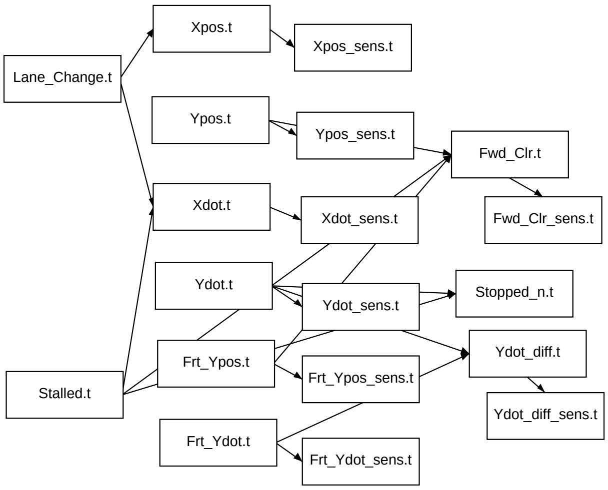

Figure 3: Belief network fragment for a single vehicle.

Some of the nodes in Figure 3, such as Xpos_sens.t and Xdot_sens.t, correspond to discretized sensor values that are set in each new slice when the slice is added to the network. For instance, the Xpos_sens.t node represents a vehicle's left-right position among the lanes of a highway and can take on one of ten states indicating the vehicle's distance from the right edge of the lanes. Other nodes, such as Stalled.t and LaneChange.t, correspond to high-level events. For example, the LaneChange.t node can take on one of three different states indicating if a vehicle is going straight, changing lanes to the left, or changing lanes to the right. The posterior probability distributions for these high-level events are affected by the sensor values in the current slice as well as the posterior probabilities of nodes in the previous slice. These distributions are then used to provide symbolic descriptions of the traffic scene.

Figure 4 shows how nodes are replicated from time slice 0 to time slice 1, as well as how some variables in time slice 1 depend on variables in the previous time slice. For example, Ypos.t1 (representing a vehicle's forward position on the highway) depends on Ypos.t0 (its previous position) and Ydot.t0 (its previous velocity).

The probabilities associated with each node provide a natural framework to encode knowledge about traffic behavior and rules. For example, the probability table for Ydot.t1 in Figure 4 contains probabilities for each

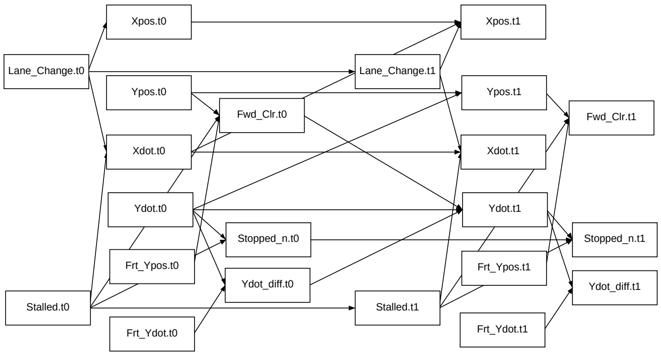

Figure 4: Belief network fragment for a single vehicle projected over one time slice. Some nodes have been omitted for simplicity.

of Ydot.t1's possible states (e.g., 21-30 km/hr, 31-40 km/hr, etc.) given the states of Ydot.t0, Fwd_Clr.t0 (the space in front of a vehicle), and Ydot_diff.t0 (the difference in speed between a given vehicle and the vehicle in front of it). A driver is likely to slow down if there isn't much distance between his vehicle and the vehicle in front and if his vehicle is going faster than the vehicle in front. Thus, the appropriate entries in the probability table will indicate a high probability that the vehicle's speed at time t1 will be lower than its speed at time t0. Similarly, the other entries in the table encode probability distributions for the new velocity given the combinations of parent states. Additional traffic knowledge that is or will be encoded includes knowledge about lane-changing and braking behavior, the effect of road geometry and weather on driving behavior, and the significance of brake, hazard, and signal lights.

Network Issues

Handling multiple vehicles. As mentioned earlier, in each time slice the structure in Figure 3 is replicated for each tracked vehicle in the traffic scene. Clearly, the positions and velocities of different vehicles will affect each other. Thus, determining globally consistent probability distributions for each vehicle involves a large network consisting of changing interconnections between vehicle subnetworks. We have investigated this approach and found the cost of modifying and recompiling the network at each time slice too computationally expensive. Nevertheless, we plan to pursue this avenue further, perhaps using approximation methods.

Our current approach is to assign each vehicle its own dynamic belief network. We incorporate the influence of nearby vehicles on the current vehicle by assigning some nodes to those vehicles. For example, Front_Ypos.t and Front_Ydot.t in Figure 3 refer to "the vehicle in front of the current vehicle". Since the actual vehicle in front may change, these indexical nodes

(Agre & Chapman 1987) do not correspond to a specific vehicle. Instead, a preprocessing step uses sensor data to determine which vehicles are currently in front of each other and then sets those node states accordingly. Using multiple, per-vehicle belief networks with indexical nodes has yielded a reasonably inexpensive approach to achieving locally consistent high-level descriptions for each vehicle while considering the affect of nearby vehicles.

Nodes with variable semantics. When a vehicle first stops on the highway, it could be for any number of reasons. The probability that the vehicle is stalled may be small at first, but it increases over time if the vehicle continues to remain stopped while no vehicles are stopped in front of it. To allow flexible representation of how a vehicle being stalled relates to it being stopped for some time, we made it possible for nodes to have variable semantics, i.e. a node refers to a different event in different time slices. This is accomplished by modifying the node's conditional probability table from one time slice to another. For example, the Stopped_n.t node has some value n associated with it, and the node refers to the event that the vehicle has been stopped for n time slices. The probability table for the node is modified according to the value of n associated with it. This can be computed with a simple function to simulate a counter (e.g., we can give vehicles positive probability of being stalled only if they've been stopped for over 50 time slices) or with any arbitrarily complex function.

Rolling the network forward. Because the HUGIN system is geared toward standard rather than dynamic belief networks, we developed the facilities necessary for rolling the network forward. Essentially, this involves adding the capability to add new time slices to the network and to incorporate information from old slices to the rest of the network so that the old slices can be deleted.

We developed two approaches to this problem. In the first approach, we generated and compiled a new network for each time slice, and we used a new network every time a slice was added. This approach seemed adequate and offered the opportunity to dynamically alter the actual network structure (which would be necessary for a global network of all the vehicles), but it suffered from the poor performance noted earlier.

We currently use our second approach, which employs two precompiled networks, each with two slices. As shown in Figure 5, the system alternates between the two networks. To introduce sensor information from a new time slice, the system incorporates the evidence from the oldest slice into the rest of the model through a series of straightforward matrix multiplications. The resulting probability tables are stored in the first slice of the other network. The new sensor information is then added to the second slice of this network, and their influence is propagated to obtain new posterior probabilities. This other network is then used until the next time slice, when the rollup procedure is repeated back to the first network. This approach does not allow dynamic alteration of the belief network structure, but it greatly improves performance by eliminating the need for network recompilation after every time slice.

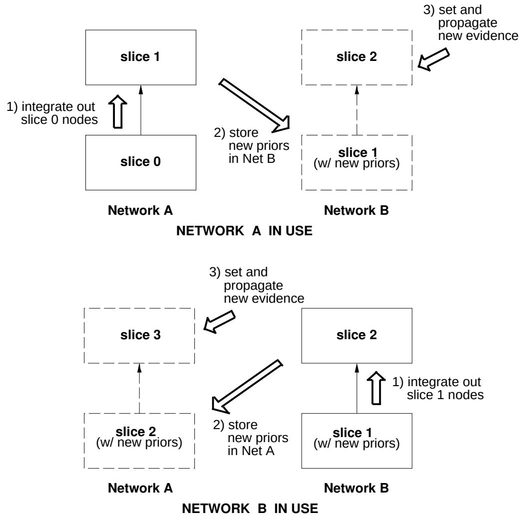

Figure 5: Steps for rolling the dynamic network forward.

The dHUGIN package (Kjaerulff 1993) provides extensions to HUGIN for dynamic belief networks, but it does not provide the flexibility of our first approach for changing the network structure from one time slice to the next, and it does not provide the performance speedup of our second approach.

Results with Real-World Traffic Scenes

We have tested our system on real-world image sequences, and we present here the results of one 270-frame sequence of a divided four-lane freeway. The image at the top of Figure 6 shows frame #40 of the sequence overlaid with contour estimates of the vehicles. The image at the bottom shows only the vehicle contour estimates and their tracks (starting from frame #0). The image at the top of Figure 7 shows frame #64 of the sequence, and the graphic on the bottom shows a reconstruction in the SmartPath traffic simulator[1] of the image (the geometry is slightly different due to the display of the SmartPath simulator). In the graphic, one vehicle has been identified by the symbolic reasoner as changing lanes, and the number in the signpost correctly indicates the number

[1]SmartPath is a microscopic three-dimensional automated highway simulator developed at UC Berkeley as part of the PATH (Partners for Advanced Transit and Highways) program of the Institute for Transportation Studies.

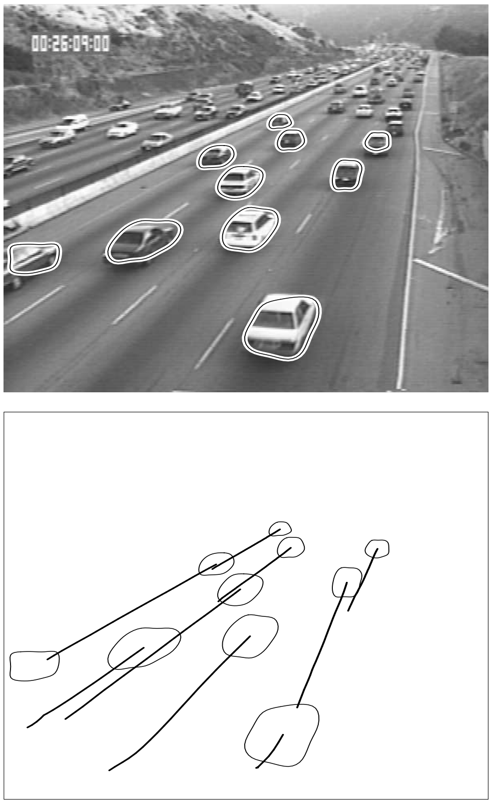

Figure 6: The upper image shows frame #40 of the image sequence with overlaid contour estimates of the cars. The bottom image shows the contour estimates with their tracks (starting from frame #0).

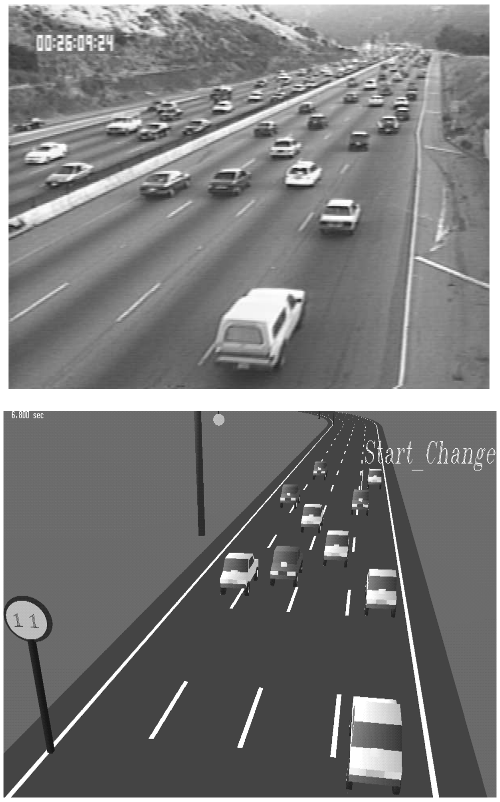

Figure 7: The upper image shows frame #64 of the sequence. The bottom graphic shows a reconstruction in the SmartPath traffic simulator of this image.

of vehicles that have passed since the beginning of the image sequence.

Running on a Sun SparcStation 10, the performance of the vision component reaches about two seconds per frame for simultaneous tracking of about 10 vehicles. A high-speed implementation on special purpose hardware using C-40 digital signal processors is in progress. The performance of the belief network varies greatly with the network design, but generally requires about one second per vehicle per frame. We expect to improve the performance of both components by an order of magnitude with various optimizations. Operation in real-time would require sampling image frames quickly enough for the affine tracker to associate vehicles between frames and for the symbolic reasoner to detect short traffic events such as lane changes. We expect that operation at 10 Hz for the vision system and 3 Hz for the symbolic reasoner will be sufficient for real-time performance.

Conclusions / Future Work

In this paper we have described the successful combination of a low-level, vision-based surveillance system with a high-level, symbolic reasoner based on dynamic belief networks. This prototype system provides robust, high-level information about traffic scenes, such as lane changes, stalled vehicles, and overall vehicle counts. We believe that the required accuracy can in the long run only be obtained using high-level reasoning under uncertainty.

The symbolic reasoner is already capable of using other vehicle features, such as vehicle type, turn signals and brake lights, to improve its analytical performance. We are currently upgrading the vision system to detect these features, as well as to handle vehicle shadows (Kilger 1992). Furthermore, the inferences of the symbolic reasoner can be fed back to the tracker's Kalman filter to further increase its reliability. For example, if a vehicle is signalling left, its expected motion update should be biased toward leftward acceleration rather than a random perturbation. This allows for reduced variance, and hence greater reliability in tracking. In the extreme case, if the low-level tracker loses a vehicle (for example, in heavy rain), the high-level system can automatically "track" its most likely position by a combination of extended projection and inference from the behavior of other vehicles.

Another benefit is the robust data fusion provided by Bayesian inference. This is especially important at dusk or dawn, when the surveillance system will see both vehicle outlines and vehicle tail lights. Finally, by including a simple sensor failure model, the network can detect and diagnose sensor failure, while continuing to track vehicles using remaining sensor inputs (Nicholson 1992).

Besides continuing to refine the network design and to optimize its performance, we are investigating methods for enabling the symbolic reasoner to handle mixed networks with both continuous and discrete variables (Lauritzen 1992; Shachter & Kenley 1989). This offers the opportunity for greater performance over purely discrete networks, and it seems reasonable, since sensor variables such as vehicle positions and velocities are adequately modelled as Gaussians. The symbolic reasoner can also be enhanced to provide other types of descriptions, such as driver behaviors. Machine learning techniques applied to a library of image sequences can be used to generate detailed probabilistic models of driver behavior, which are useful both in our own work and in analytical and simulation studies of highway designs.

We are currently moving the implementation of the prototype (running on single Sun SparcStations) to a heterogeneous system consisting of a host Sun SparcStation and special purpose hardware. This will improve the setup for large-scale experimentation and will improve performance to about 5Hz, which we believe will be adequate for traffic surveillance in sunny California weather. To better assess the system's usefulness and accuracy, we plan to measure its performance on a more extensive collection of video sequences.

Acknowledgments

We gratefully acknowledge the help of C. McCarley and his group at Cal Poly, San Luis Obispo, for providing us with video tapes of various traffic scenes. We also thank HUGIN Expert A/S for their generous doctoral student license to use the HUGIN system.

References

P. Agre, D. Chapman. Pengi: An Implementation of a Theory of Activity, in *Proceedings of the Sixth National Conference on Artificial Intelligence*, 1987.

S. Andersen, K. Olesen, F. V. Jensen, F. Jensen. HUGIN* – a Shell for Building Bayesian Belief Universes for Expert Systems, in *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, 1989.

R. Bartels, J. Beatty, B. Barsky. *An Introduction to Splines for use in Computer Vision*, Morgan Kaufmann, 1987.

A. Blake, R. Curwen, A. Zisserman. Affine-invariant contour tracking with automatic control of spatiotemporal scale, in *Proc. Int. Conf. on Computer Vision*, Berlin, Germany, May. 11-14, 1993, pp. 66–75.

R. Curwen, A. Blake. *Active Vision*, MIT Press, Cambridge, MA, 1992, chapter Dynamic Contours: Real-time Active Snakes, pp. 39–57.

N. Heinze, W. Krüger, H.-H. Nagel. Berechnung von Bewegungsverben zur Beschreibung von aus Bildfolgen gewonnenen Trajektorien in Straßenverkehrsszenen, *Informatik – Forschung und Entwicklung* **6** (1991), pp. 51–61.

T. Huang, G. Ogasawara, S. Russell. Symbolic Traffic Scene Analysis Using Dynamic Belief Networks, in *AAAI Workshop on AI in IVHS*, Washington D.C., 1993.

Klaus-Peter Karmann, Achim von Brandt. Moving Object Recognition Using an Adaptive Background Memory, in V Cappellini (ed.), *Time-Varying Image Processing and Moving Object Recognition, 2*, Elsevier, Amsterdam, The Netherlands, 1990.

M. Kass, A. Witkin, D. Terzopoulos. Snakes: Active Contour Models, *International Journal of Computer Vision* **1** (1988) 321–331.

M. Kilger. A Shadow Handler in a Video-based Real-time Traffic Monitoring System, in *IEEE Workshop on Applications of Computer Vision*, Palm Springs, CA, 1992, pp. 1060–1066.

U. Kjaerulff. User's Guide to dHUGIN, Institute of Electronic Systems, Aalborg University, 1993.

D. Koller, N. Heinze, H.-H. Nagel. Algorithmic Characterization of Vehicle Trajectories from Image Sequences by Motion Verbs, in *IEEE Conf. Computer Vision and Pattern Recognition*, Lahaina, Maui, Hawaii, June 3-6, 1991, pp. 90–95.

D. Koller, J. Weber, J. Malik. *Robust Multiple Car Tracking with Occlusion Reasoning*, technical report UCB/CSD-93-780, University of California at Berkeley, October 1993.

D. Koller, J. Weber, J. Malik. Robust Multiple Car Tracking with Occlusion Reasoning, in *Proc. Third European Conference on Computer Vision*, Stockholm, Sweden, May 2-6, 1994, J.-O. Eklundh (ed.), Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg, New York (to appear), 1994.

S. Lauritzen. Propagation of Probabilities, Means, and Variances in Mixed Graphical Association Models, in *Journal of the American Statistical Association*, vol. 87, no. 420, 1992.

A. Nicholson. Monitoring Discrete Environments Using Dynamic Belief Networks, PhD thesis, Oxford University, 1992.

A. Niehaus, R. F. Stengel. Rule-Based Guidance for Vehicle Highway Driving in the Presence of Uncertainty, in *Proceedings of the 1991 American Control Conference*, 1991.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, San Mateo, CA, 1988.

J. R. J. Schirra, G. Bosch, C. K. Sung, G. Zimmermann. From Image Sequences to Natural Language: A First Step towards Automatic Perception and Description of Motion, *Applied Artificial Intelligence* **1** (1987) 287–307.

R. Shachter, C. Kenley. Gaussian influence diagrams, in *Management Science* **35**, 1989.