In D. Kriegman, G. Hager, S. Morse editors, ''The Confluence of Vision and Control''
Springer Verlag, 1998. In press.

Visual Routines for Vehicle Control*

Garbis Salgian Dana H. Ballard
Computer Science Department
University of Rochester
Rochester, NY 14627

1 Introduction

Automated driving holds the promise of improving traffic safety, alleviating highway congestion and saving fuel. The continuous increase in processor speed over the last decade has led to an increased effort in research on automated driving in several countries [1]. However, autonomous tactical level driving (i.e. having the ability to do traffic maneuvers in complex, urban type environments) is still an open research problem.

As little as a decade ago, it was widely accepted that the visual world could be completely segmented into identified parts prior to analysis. This view was supported in part by the belief that additional computing cycles would eventually be available to solve this problem. However the complexity of vision's initial segmentation can easily be unbounded for all practical purposes, so that the goal of determining a complete segmentation of an individual scene in real time is impractical. Thus to meet the demands of ongoing vision, the focus has shifted to a more piecewise and on-line analysis of the scene, wherein just the products needed for behavior are computed as needed. Such products can be computed by *visual routines* [14], special purpose image processing programs that are designed to compute specific parameters that are used in guiding the vehicle.

This paper describes the development and testing of visual routines for vehicle control. It addresses the generation of visual routines from images using appearance based models of color and shape. The visual routines presented here are a major component of the perception subsystem of an intelligent vehicle. The idea of visual routines is compelling owing to the fact that being special-purpose vast amounts of computation can be saved. For this reason they have been used in several simulations (*eg.* [9]), but so far they have been used in image analysis only in a few restricted circumstances.

*This research was supported by NIH/PHS research grant 1-P41-RR09283

2 Photo-realistic simulation

Autonomous driving is a good example of an application where it is necessary to combine perception (vision) and control. However, testing such a system in the real world is difficult and potentially dangerous, especially in complex dynamic environments such as urban traffic.

Given recent advances in computer graphics, both in terms of the quality of the generated images and the rendering speed, we believe that a viable alternative to initial testing in the real world is provided by integrating *photo-realistic simulation* and real-time image processing. This allows testing the computer vision algorithms under a wide range of controllable conditions, some of which would be too dangerous to do in an an actual car. The resultant testbed leads to rapid prototyping.

Terzopoulos pioneered the use of simulated images in his animat vision architecture. However, in their approach all the processing is carried out in software, one of the motivations for the architecture being that it avoids the difficulties associated with "hardware vision" [13]. In our case, the graphical output from the simulator is sent to a separate subsystem (host computer with pipeline video processor) where the images are analyzed in real-time and commands are sent back to the simulator (figure 1). The images are generated by an SGI Onyx Infinite Reality engine which uses a model of a small town and the car. Visual routines are scheduled to meet the temporary task demands of individual driving sub-problems such as stopping at lights and traffic signs. The output of the visual routines is used to control the car which in turn affects the subsequent images. In addition to the simulations, the routines are also tested on similar images generated by driving in the real world to assure the generalizability of the simulation.

The simulator can also be used with human subjects who can drive a kart through the virtual environment while wearing head mounted displays (HMD). A unique feature of our driving simulator is the ability to track eye movements within a freely moving VR helmet which allows us to explore the scheduling tradeoffs that humans use. This provides a benchmark for the automated driver and also is a source for ideas as to priorities assigned by the human driver. In particular, the fixation point of the eyes at any moment is an indicator of the focus of attention for the human operator. Experiments show that this fixation point can be moved at the rate of one fixation every 0.3 to 1 second. Studying the motion of this fixation point provides information on how the human driver is allocating resources to solve the current set of tactical driving-related problems.

3 Perceptual and Control Hierarchy

The key problem in driving at a tactical level is deciding what to attend to. In our system this problem is mediated by a scheduler, which decides which set of behaviors to activate. The central thesis is that, at any moment, the demands of driving can be met by a small number of behaviors. These behaviors, in turn,

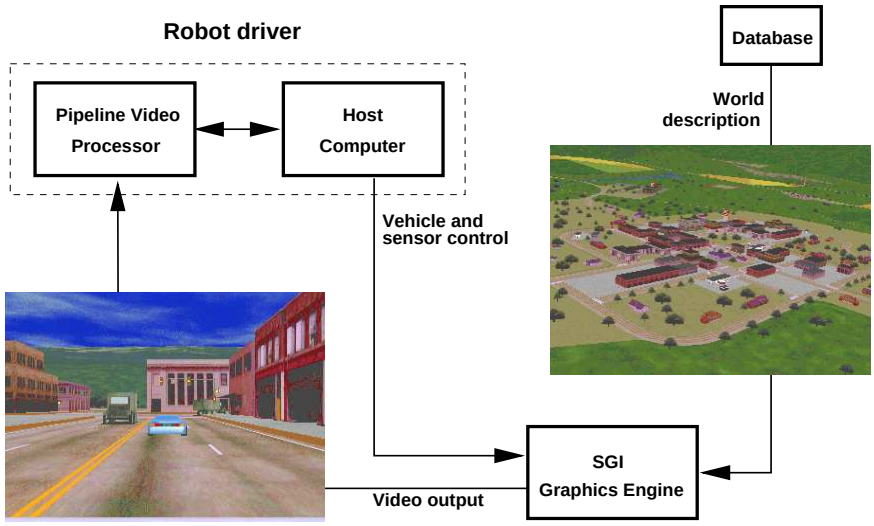

Figure 1: The graphical output of the simulator is sent to the real-time image processing hardware (Datacube color digitizer and MV200 processing board) which is connected to a host computer. The host analyzes the incoming images and sends back to the simulator controls for the vehicle and virtual camera.

invoke a library of visual routines. Recent studies have shown that humans also switch among simple behaviors when solving more complex tasks [4].

The hierarchy of perception and control levels that forms the framework for driving is presented in figure 2. While not exhaustive, all the relevant levels in implementing a behavior are represented. At the top a *scheduler* selects from a set of task-specific *behaviors* the one that should be activated at any given moment. The behaviors use *visual routines* to gather the information they need and act accordingly. Finally, the visual routines are composed from an alphabet of *basic operations* (similar to Ullman's proposal [14]).

The hierarchy in Figure 2 has many elements in common with that of Maurer and Dickmanns [6]. The main difference is one of emphasis. We have focused on the role of perception in vehicle control, seeking to compute task-related invariants that simplify individual behaviors.

To illustrate how modules on different levels in the hierarchy interact, consider the case when the scheduler activates the stop sign behavior. To determine whether there is a stop sign within certain range, the stop sign detection visual routine is invoked. The routine in its turn uses several basic operations to determine if a stop sign is present. For instance, it uses color to reduce the image area that needs to be analyzed. If there are any red blobs, they are further verified to see if they represent stop signs by checking distinctive image features. Finally, the routine returns to its caller with an answer (stop sign found or not).

If no stop sign was detected, that information is passed to the scheduler

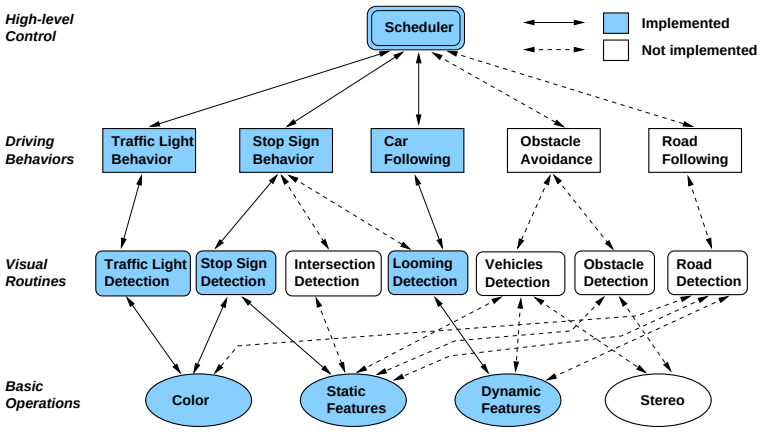

Figure 2: Hierarchy of perceptual and control modules. At the top level, the scheduler selects the behavior that is currently active. This behavior uses one or more visual routines to gather the information it needs to take the appropriate decisions. The routines are composed from a set of low-level basic operations. Shaded modules are the ones currently implemented.

which can then decide what behavior to activate next. On the other hand, if a stop sign was found, the agent has to stop at the intersection to check for traffic. For that, it needs to know where the intersection is, so the intersection detection routine will be activated. It can use static image features (*eg.* lines, corners) to determine where in the image the intersection is located. At the behavior level this information can be used for visual servoing until the intersection is reached.

The shaded modules in figure 2 are the ones that have been implemented so far. Road following has been intensely studied for more than a decade [1], [7] and it was successfully demonstrated at high speeds and over extended distances. Therefore we decided not to duplicate these efforts initially and instead to take advantage of the simulated environment. In our experiments the car is moving on a predefined track and the driving program controls the acceleration (the gas and break pedals).

3.1 Basic operations

At the lowest level in the hierarchy are basic operations. These are simple low-level functions which can be used in one or more of the higher level task-specific visual routines. The implementation uses special real-time image processing hardware, namely two Datacube boards. One is a color digitizer (Digicolor) and the other is the main processing board (MV200).

Color. The role of the *color primitive* is to detect blobs of a given color. An incoming color image is digitized in the Hue, Saturation, Value color space. Colors are defined as regions in the hue-saturation sub-space and a lookup table

is programmed to output a color value for every hue-saturation input pair. A binary map corresponding to the desired color is further extracted and analyzed using a blob labeling algorithm. The end result is a list of bounding rectangles for the blobs of that color.

Static features. The role of the static feature primitive is to detect objects of a specific appearance. It uses steerable filters, first proposed by Freeman and Adelson [3], who showed how a filter of arbitrary orientation and phase can be synthesized from a set of basis filters (oriented derivatives of a two-dimensional, circularly symmetric Gaussian function). Other researchers have used these filters for object identification [8]. The idea is to create a unique index for every image location by convolving the image at different spatial resolutions with filters from the basis set. If M filters are applied on the image at N different scales, an $M \times N$ element vector response is generated for every image position. For appropriate values of M and N, the high dimensionality of the response vector ensures its uniqueness for points of interest.

Searching for an object in an image is realized by comparing the index of a suitable point on the model with the index for every image location. The first step is to store the index (response vector) $\mathbf{r^m}$ for the chosen point on the model object. To search for that object in a new image the response $\mathbf{r^i}$ at every image point is compared to $\mathbf{r^m}$ and the one that minimizes the distance $d_{im} = ||\mathbf{r^i} - \mathbf{r^m}||$ is selected, provided that d_{im} is below some threshold.

More details about the color and static feature primitives and their real-time implementation on the Datacube hardware are given in [10].

Dynamic features. The goal of this primitive is to detect features that expand or contract in the visual field. The primitive combines three separate characteristics. Each of these have been explored independently, but our design shows that there are great benefits when they are used in combination, given the particular constraints of the visual environment during driving. The first characteristic is that of the special visual structure of looming itself. In driving, closing or losing ground with respect to the vehicle ahead creates an expansion or contraction of the local visual field with respect to the point of gaze [5]. The second one is that the expansion and contraction of the visual field can be captured succinctly with a log-polar mapping of the image about the gaze point [11]. The third characteristic is that the looming is detected by correlating the responses of multiple oriented filters.

Starting from I_t, the input image at time t, the first step is to create LP_t, the log-polar mapping at time t. This is done in real time on the pipeline video processor using the miniwarper, which allows arbitrary warps. Since dilation from the center in the original image becomes a shift in the new coordinates, detecting looming in the original input stream I_t translates into detecting horizontal shifts in the stream of transformed images LP_t, with $0 \leq t < t_{max}$

Another reason for using a log-polar mapping is that the space-variant sampling emphasizes features in the foveal region while diminishing the influence of those in the periphery of the visual field (figure 4 left). This is useful in the car following scenario, assuming fixation is maintained on the leading vehicle,

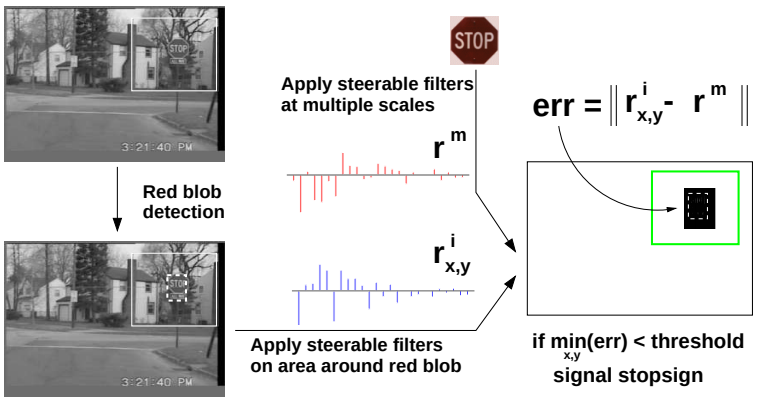

Figure 3: Stop sign detection routine.

since it reduces the chance of false matches in the periphery.

The *Dynamic Feature Map* (DFM) indicates the regions in the image where a specified shift is present between LP_t and LP_{t-1}. $DFM_{s,t}$ denotes the map at time t with a shift value s and is obtained by correlating LP_t with LP^s_{t-1} (where the superscript indicates the amount of shift).

In order to reduce the number of false matches, the correlation is performed in a higher dimensional space by analyzing the responses of five different filters (from the same basis set as in the static feature case).

3.2 Visual Routines

Basic operations are combined into more complex, task-specific routines. Since the routines are task-specific, they make use of high level information (*eg.* a geometric road model, known ego-motion, *etc.*) to limit the region of the image that needs to be analyzed, which leads to reduced processing time. We have implemented routines for stop light, stop sign and looming detection.

Stop light detection. The stop light detection routine is an application of the color blob detection primitive to a restricted part of the image. Specifically, it searches for red blobs in the upper part of the image. If two red blobs are found within the search area, then a stop light is signaled.

Currently, the search window is fixed a priori. Once we have a road detection routine, we will use that information to adjust the position and size of the window dynamically.

Stop sign detection. The area searched for stop signs is the one on the right side of the road (the white rectangle in the right side of every image in figure 3). First, the color primitive is applied to detect red blobs in this area, which are candidates for stop signs. Since other red objects can appear in this region (such as billboards, brick walls, etc.) the color test alone is not enough for detecting the stop signs, being used just as a "focus of attention" mechanism to further limit the image area that is analyzed.

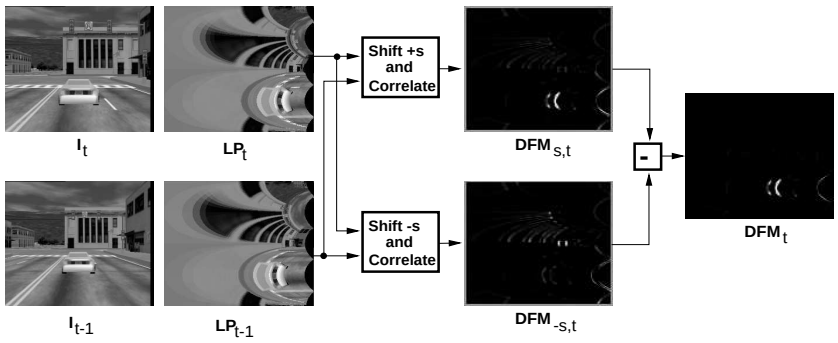

Figure 4: The overall looming detection routine combines the results of two dynamic feature maps, one using a positive shift, another using a negative shift.

Once a red blob is detected, the static feature primitive is applied to determine whether any of the filter responses $\mathbf{r^i_{x,y}}$ in that area (dashed white rectangle) matches the previously stored response for a stop sign $\mathbf{r^m}$. If the error (difference) is below some predetermined threshold, a stop sign is reported.

The two routines have been tested both in simulation and on real world video sequences. Sample results are presented in [10]

Looming detection. The *looming detection* routine applies two instances of the dynamic feature primitive (for two equal shifts of opposite signs) on consecutive frames in log-polar coordinates. Figure 4 illustrates the main steps and some intermediate results for the case when the leading car is approaching (expanding from I_{t-1} to I_t). Consequently, features on the car shift to the right from LP_{t-1} to LP_t and show up in $DFM_{s,t}$ but not in $DFM_{-s,t}$.

A single dynamic feature map DFM_t is computed as the difference of $DFM_{s,t}$ and $DFM_{-s,t}$. By taking the difference of the two maps, the sensitivity to speeds in the region where the distributions for s and $-s$ overlap is reduced. This is visible in figure 4, where features from the building in the background are present in both $DFM_{s,t}$ and $DFM_{-s,t}$, but cancel each other in DFM_t, which contains only the features corresponding to the car.

DFM_t is analyzed for blobs and the list of blobs (with size, centroid and sign) is returned. The sign indicates whether it is a dilation or a contraction. If there is more than one blob, the correspondence is determined across frames based on the distance between them. The tracking can be further simplified by analyzing only a sub-window of the dynamic feature map corresponding to a region of interest in the original image (*eg.* the lower part if the shadow under the lead vehicle is tracked).

3.3 Driving Behaviors

Visual routines are highly context dependent, and therefore need an enveloping construct to interpret their results. For example, the stop light detector

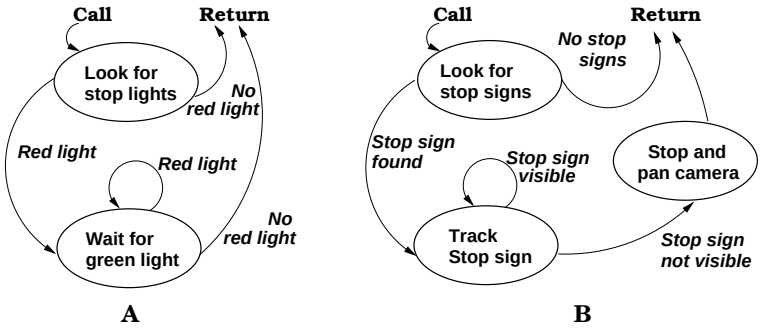

Figure 5: Finite state machines used for two driving behaviors (**A** traffic light behavior and **B** stop sign behavior).

assumes the lights are in a certain position when gaze is straight ahead, thus the stop light behavior has to enforce this constraint. To do this, the behaviors are implemented as finite state machines, presented in figure 5.

Traffic light behavior. The initial state is "Look for stop lights", in which the traffic light detection routine is activated. If no red light is detected the behavior returns immediately. When a red light is detected, the vehicle is instructed to stop and the state changes to "Wait for green light" in which the red light detector is executed. When the light changes to green, the routine will return "*No red light*" at which time the vehicle starts moving again and the behavior completes.

Stop sign behavior. In the "Look for stop signs" state the stop sign detection routine is activated. If no sign is detected the behavior returns immediately. When a stop sign is detected, the agent needs to stop at the intersection. Since we don't have an intersection detector yet, once the stop sign is detected, the state changes to "Track stop sign" in which the vehicle moves forward while tracking the sign. When the sign is no longer visible, a new state is entered in which the agent stops and pans the camera left and right.

Car following behaviors. The looming detection routine can be used to build a car following behavior. Two such behaviors are presented, one purely reactive and another one that tries to maintain a constant distance to the leading vehicle.

Reactive behavior. This behavior does not model the motion of the leading vehicle. It has a default speed V_{def}, at which the vehicle is moving if nothing is detected by the looming routine. When there is something looming in front of the vehicle, the routine returns the horizontal coordinate of the corresponding blob centroid in the DFM and its sign. Based on these two inputs, the desired speed V_{des} is computed to ensure that the maximum brake is applied when the leading vehicle is close and approaching and the maximum acceleration is applied when the distance to the lead vehicle is large and increasing. The actual vehicle speed is determined by the current speed, the desired speed and vehicle dynamics.

Constant distance behavior. This behavior tries to maintain a constant

distance to the leading vehicle by monitoring the position of the blob centroid x_c in the dynamic feature map. The desired relative distance is specified by the corresponding horizontal position in log-polar coordinates x_{des}. The error signal $x_{err} = x_{des} - x_c$ is used as input to a proportional plus integral (PI) controller whose output is the vehicle desired speed.

3.4 Scheduling

Given a set of behaviors, and a limited processing capacity, the next problem to address is how to schedule them in order to ensure that the right behavior is active at the right time. This issue has been addressed by other researchers and several solutions have been proposed: inference trees [9], and more recently, distributed architectures with centralized arbitration [12].

We are currently investigating different alternatives for the scheduler design. So far our principal method is to alternate between the existing behaviors, but there are important subsidiary considerations. One is that the performance of difficult or important routines can be improved by scheduling them more frequently. Another is that the performance of such routines can be further improved by altering the behavior, for example by slowing down. The effect of different scheduling policies is addressed in [10].

4 Experiments

The two car following behaviors have been tested in simulation. The leading vehicle is moving at a constant speed of 48 km/h and the initial distance between vehicles was around 20 meters.

For the reactive case, the default speed was set to 58 km/h. The results are shown in the center column of figure 6: the upper plot is the vehicle speed, and the lower one is the relative distance. The reactive characteristic of the behavior is noticeable in the speed profile, which has a seesaw pattern. The distance to the leading car varies significantly, which is to be expected since the controller has no model of the relative position of the vehicles.

In the case of the constant distance behavior, the desired position was set initially to correspond to a relative distance of about 20 meters, and after 10 seconds it was changed to a relative distance of about 11 meters. The upper right plot in figure 6 shows the speed profile of the vehicle, which is closer to the speed of the leading vehicle than in the reactive case. Also, the relative distance (lower right plot) varies significantly less around the desired value. The response to the step change is relatively slow (about 10 seconds to get at the new relative distance), but this is determined by the parameters of the controller. We have not extensively experimented with the possible parameter values, the main focus so far being to show that the looming detection routine provides a robust enough signal that can be used in a car following behavior.

The leftmost column in figure 6 shows the results for a human driving in the same virtual environment. The fact that humans also exhibit a characteristic sawtooth pattern in speed change may suggest that they rely on the looming

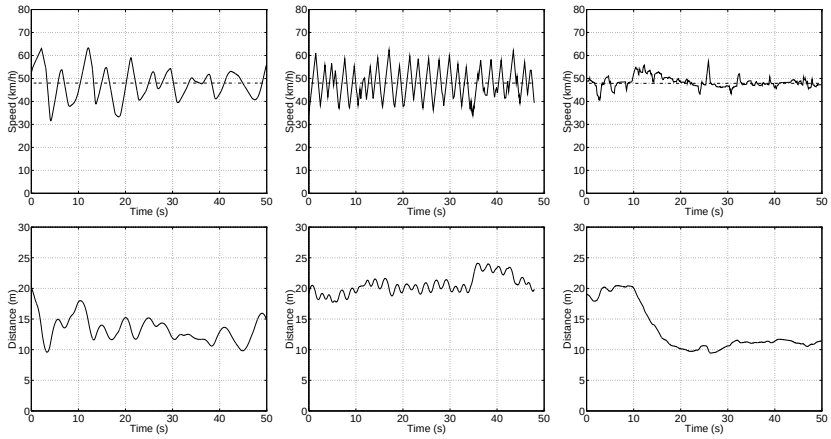

Figure 6: Vehicle speed (up) and distance between vehicles (down) for a human driver (left) and two instances of the robot driver with different car following behaviors: reactive (center) and constant distance (right). The results show that the former has an absolute error of about 5 meters and the latter about 1.5 meters.

cue for car following (as opposed to using other image cues to estimate the relative distance).

The tests here have assumed the functioning of the tracking system that can identify the rough position of the lead vehicle during turns. This information is in the optic flow of the dilation and contraction images in that vertical motion of the correlation images indicates turns. Figure 7 shows the real angular offset (dotted line) and the value recovered from the vertical position of the blob corresponding to the lead vehicle in the dynamic feature map (solid line). The right side shows the same data, after removing the lag. Our future plan is to use the measured angular offset to control the panning of the virtual camera in order to maintain fixation on the lead vehicle when it turns.

5 Conclusions

Driving is a demanding dynamically changing environment that places severe temporal demands on vision. These demands arise owing to the need to do a variety of things at once. One way to meet them is to use specially-designed visual behaviors that are geared to sub-parts of the task. Such visual behaviors in turn use visual routines that are specialized to location and function. Our hypothesis is that:

1. The complex behavior of driving can be decomposed into a large library of such behaviors, and

2. At any moment the tactical demands of driving can be meet by special

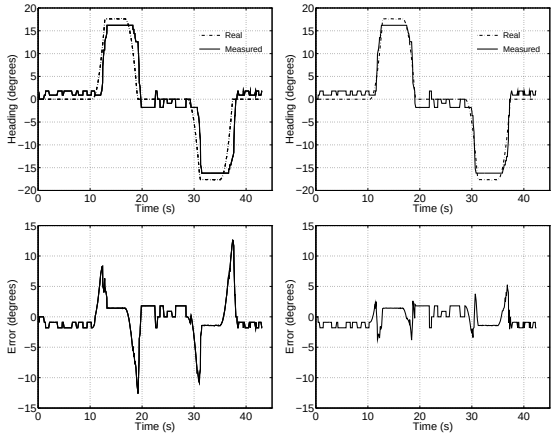

Figure 7: *Left:* Angular offset of the lead vehicle in the visual field of the follower for a road segment with two turns (the camera is looking straight ahead); *Right:* Same data, after removing the lag due to rendering and image processing.

purpose hardware that runs just a collection of these behaviors, and

3. The appropriate subset of such routines can be selected by a scheduling algorithm that requires a much slower temporal bandwidth.

We demonstrated this design by implementing three such behaviors, a stop sign behavior, a traffic light behavior and a car following behavior. All three take advantage of special purpose video pipeline processing to execute in approximately 100 milliseconds, thus allowing real-time behavior.

The tests of the looming behavior show that it is extremely robust, and is capable of following cars over a wide range of speeds and following distances. The obvious alternate strategy for car-following would be to track points on the lead car. This has been tried successfully [2] but requires that the tracker identify points on the vehicle over a wide variety of illumination conditions. In contrast the method herein does not require that the scene be segmented in any way. It only requires that the visual system can track the lead vehicle during turns and that the relative speeds between them are slower than their absolute speeds.

As of this writing, the various behaviors have only been tested under simple conditions. Future work will test the robustness of the scheduler under more complicated driving scenarios where the demands of the visual behaviors interact. One such example is that of following a car while obeying traffic lights.

The demonstration system is a special design that allows the output of a Silicon Graphics Onyx Infinite Reality to be sent directly to the video pipeline computer. The results of visual processing are then sent to the car model and appropriate driving corrections are made. This design is useful for rapid

prototyping, allowing many situations to be explored in simulation that would be dangerous, slow or impractical to explore in a real vehicle.

References

[1] E. D. Dickmanns. Performance improvements for autonomous road vehicles. In *Proceedings of the 4th International Conference on Intelligent Autonomous Systems*, pages 2–14, Karlsruhe, Germany, March 27-30 1995.

[2] U. Franke, F. Böttiger, Z. Zomotor, and D. Seeberger. Truck platoonong in mixed traffic. In *Proceedings of the Intelligent Vehicles '95 Symposium*, pages 1–6, Detroit, USA, September 25-26 1995.

[3] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, September 1991.

[4] M. F. Land and S. Furneaux. The knowledge base of the oculomotor system. Sussex Centre for Neuroscience, School of Biological Sciences, University of Sussex, Brighton BN1 9QG, UK, 1996.

[5] D. N. Lee. A theory of visual control of braking based on information about time-to-collision. *Perception*, 5:437–459, 1976.

[6] M. Maurer and E. Dickmanns. An advanced control architecture for autonomous vehicles. In *Navigation and Control Technologies for Unmanned Systems II*, volume 3087 of *SPIE*, pages 94–105, Orlando, FL, USA, 23 April 1997.

[7] D. Pomerleau. Ralph: rapidly adapting lateral position handler. In *Proceedings of the Intelligent Vehicles '95 Symposium*, pages 506–511, New York, NY, USA, September 1995. IEEE.

[8] R. P. Rao and D. H. Ballard. Object indexing using an iconic sparse distributed memory. In *ICCV-95*, pages 24–31, June 1995.

[9] D. A. Reece. Selective perception for robot driving. Technical Report CMU-CS-92-139, Carnegie Mellon University, 1992.

[10] G. Salgian and D. H. Ballard. Visual routines for autonomous driving. In *Proceedings of the 6th International Conference on Computer Vision (ICCV-98)*, pages 876–882, Bombay, India, January 1998.

[11] E. L. Schwartz. Anatomical and physiological correlates of visual computation from striate to infero-temporal cortex. *IEEE Transactions on systems, man and cybernetics*, SMC-14(2):257–271, April 1984.

[12] R. Sukthankar. *Situation Awarness for Tactical Driving.* PhD thesis, Robotics Institute, CMU, Pittsburg, PA 15213, January 1997. CMU-RI-TR-97-08.

[13] D. Terzopoulos and T. F. Rabie. Animat vision: Active vision in artificial animals. In *ICCV-95*, pages 801–808, June 1995.

[14] S. Ullman. Visual routines. *Cognition*, (18):97–160, 1984.