AI Magazine, Vol. 17, No. 2, pg. 11 - SO, Summer 1996

Life in the *Fast Lane*

The Evolution of an Adaptive Vehicle Control System

Todd Jochem and Dean Pomerleau

Giving robots the ability to operate in the real world has been, and continues to be, one of the most difficult tasks in AI research. Since 1987, researchers at Carnegie Mellon University have been investigating one such task. Their research has been focused on using adaptive, vision-based systems to increase the driving performance of the Navlab line of on-road mobile robots. This research has led to the development of a neural network system that can learn to drive on many road types simply by watching a human teacher. This article describes the evolution of this system from a research project in machine learning to a robust driving system capable of executing tactical driving maneuvers such as lane changing and intersection navigation.

For the past decade, researchers in the Navlab Project at The Robotics Institute at Carnegie Mellon University (CMU) have been conducting research on autonomous navigation of automotive robots. This research has included investigating route planning, obstacle avoidance, and position estimation. Perhaps the most dramatic improvement over the decade has been in the area of vision-based lane keeping—using a video camera to extract road information, such as lines and edges, that is used to keep the vehicle centered in its lane.

Early vision-based lane-keeping systems, at CMU and elsewhere, were model based: The programmer decided what information or features were important for driving and developed specific algorithms to detect these features. These systems typically looked for features such as asphalt-colored regions and lane markings to determine where the road boundaries were located. These systems worked well when the features they were programmed to detect were clearly visible but had difficulty when these features were obscured or absent. Other challenges occurred when the road's appearance changed dramatically, for example, from a city street to an interstate highway. In these cases, the system had to be reprogrammed to use the new road's features, a tedious and time-consuming task. A system that could handle noisy, real-world data and quickly adapt to new roads would clearly be useful.

A lane-keeping system with these capabilities began to take shape in the fall of 1987 as part of a project on machine learning using supercomputers. At that time, the WARP supercomputer was being developed at CMU's School of Computer Science. The parallel processing this computer could perform was well suited to the ***back-propagation learning algorithm,*** a basic neural network training technique. Dean Pomerleau, a first-year graduate student at the time, decided to use the WARP to investigate mobile robot control using neural networks, and the idea of ALVINN (autonomous land vehicle in a neural network) was born.

As ALVINN began to take shape, the method of training the system's neural network became the central research issue. Pomerleau wanted the system to be capable of learning to drive on many different road types, and over the course of the next several years, he developed a method to achieve this capability. ALVINN would learn to associate road images with steering wheel direction. It would learn by watching a person drive, using his/her actions as a training signal. If the person steered to the right in response to a particular road scene, ALVINN would associate this scene with a particular angle of the steering wheel. After watching and learning from many of these associations, it would be able

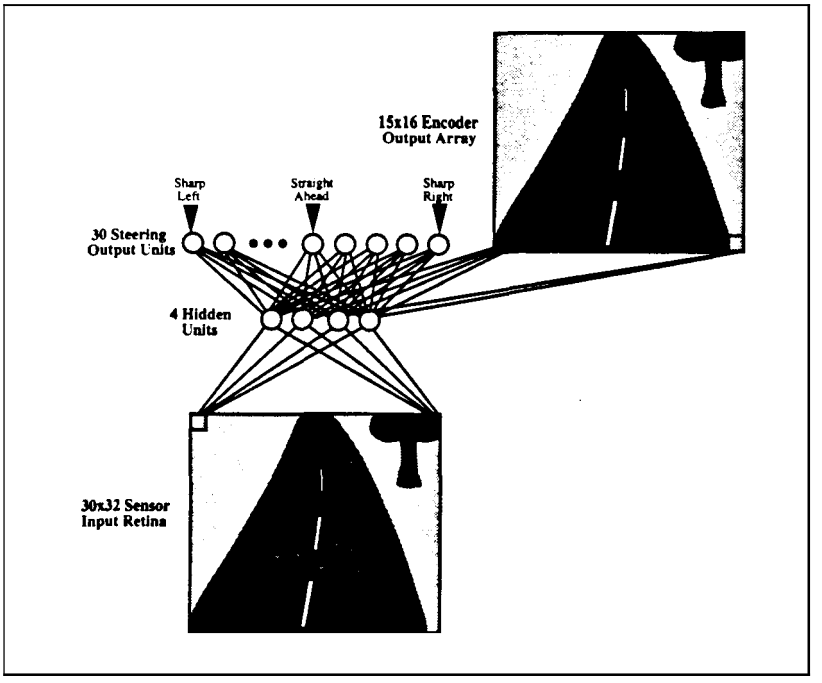

Figure 1. ALVINN's Network Architecture.

to drive by itself. When driving on a new road was desired, the system could simply be retrained by watching the person drive again.

Testing and debugging continued, and when the WARP computer was placed on the Navlab 1, a converted Chevrolet panel van, ALVINN got its "learner's permit." From this first unsteady drive down a paved bike path near campus, ALVINN has gone on to drive thousands of miles in eight different vehicles with researchers from four different institutions. ALVINN has been rewritten at least four times, going from a centralized process to a distributed architecture and back again. It no longer needs a supercomputer to run—a **486**-class machine works just fine. It has driven test-bed vehicles forward and backward, using video cameras, laser range finders, and infrared sensors.

The system, which served as the basis for Pomerleau's Ph.D., has been used by him for lane-keeping and roadway-departure warning experiments and research into neural network learning and sensitivity analysis. ***ALVINN*** was also the basis of another Ph.D. dissertation. This thesis, by Pomerleau's first graduate student, Todd Jochem, extended the capabilities of ALVINN to include tactical driving tasks such as lane changing and intersection navigation. To accomplish these tasks, Jochem used a geometric model of image formation that allowed images to be created from arbitrary locations in the world using pixels from the actual image. Because canonical, or virtu***al images,*** could be created at any location in the scene, this framework allowed ALVINN to focus its lane-finding capabilities on the areas of the scene most relevant for the task at hand. For example, if lane changing was to be performed, this enhanced version of ALVINN could be used to track each lane. Similarly, for intersection navigation, each branch could be identified.

The following sections describe Pomerleau's original ALVINN system and its accomplishments and shortfalls on the Navlab series of vehicles and Jochem's extension, which allowed it to evolve into more than just a lane-keeping system.

ALVINN

This section describes the architecture, training, and performance of the ALVINN system. It demonstrates how simple connectionist networks can learn to precisely guide a mobile robot in a wide variety of situations when trained appropriately. In particular, this section details training techniques that allow ALVINN to learn in less than five minutes to autonomously control the Navlab test-bed vehicles by watching a human driver's response to new situations. With these techniques, ***ALVINN*** has been trained to drive in a variety of circumstances, including single-lane paved and unpaved roads, multilane lined and unlined roads, and obstacle-ridden on- and off-road environments, at speeds as high as 55 miles per hour (mph).

Network Architecture

The basic network architecture used in the ALVINN system is a single hidden-layer, feed-forward neural network (figure 1). The input layer consists of a single 30 **x** 32 unit "retina" onto which a sensor image from either a video camera or a scanning laser range finder is projected. Each of the **960** input units is fully connected to the hidden layer of **4** units, which, in turn, is fully connected to the output layer. The 30-unit output layer is a linear representation of the currently appropriate steering direction that can serve to keep the vehicle on the road or prevent it from colliding with nearby obstacles, depending **on** the **type** of input sensor image and the driving situation it has been trained to handle. The centermost output unit represents the travel-straight-ahead condition, and units to the left and the right of center represent successively sharper left and right turns. The units on the extreme left and right of the out-

put vector represent turns with an approximately 30-meter radius to the left and the right, respectively, and the units in between represent turns that decrease linearly in their curvature down to the straight-ahead middle unit in the output vector.

To drive the Navlab, an image from the appropriate sensor is reduced to 30 x 32 pixels and projected onto the input layer. After propagating activation through the network, the output layer's activation profile is translated into a vehicle-steering command. The steering direction dictated by the network is taken to be the center of mass of the hill of activation surrounding the output unit with the highest activation level. Using the center of the activation mass, instead of the most active output unit, when determining the direction to steer permits finer steering corrections, thus improving ALVINN's driving accuracy.

Network Training

The network is trained to produce the correct steering direction using the back-propagation learning algorithm (Rumelhart, Hinton, and Williams 1986). In back propagation, the network is first presented with an input, and activation is propagated forward through the network to determine the network's response. The network's response is then compared with the known correct response. If the network's actual response does not match the correct response, the weights between connections in the network are modified slightly to produce a response more closely matching the correct response.

In the initial experiments, ALVINN was trained using artificially generated road images and the corresponding correct steering directions. Although this approach achieved limited success, it proved difficult to generate a-tificial images that realistically depict the variety and complexity of real-world road scenes. Instead, a method of training using real road images was required. Training on real images would dramatically reduce the human effort required to develop networks for new situations by eliminating the need for a hand-programmed training example generator. On-the-fly training should also allow the system to adapt quickly to new situations.

In theory, it should be possible to teach a network to imitate a person as he/she drives using the current sensor image as input and the person's current steering direction as the desired output. This idea of training on the fly is depicted in figure 2.

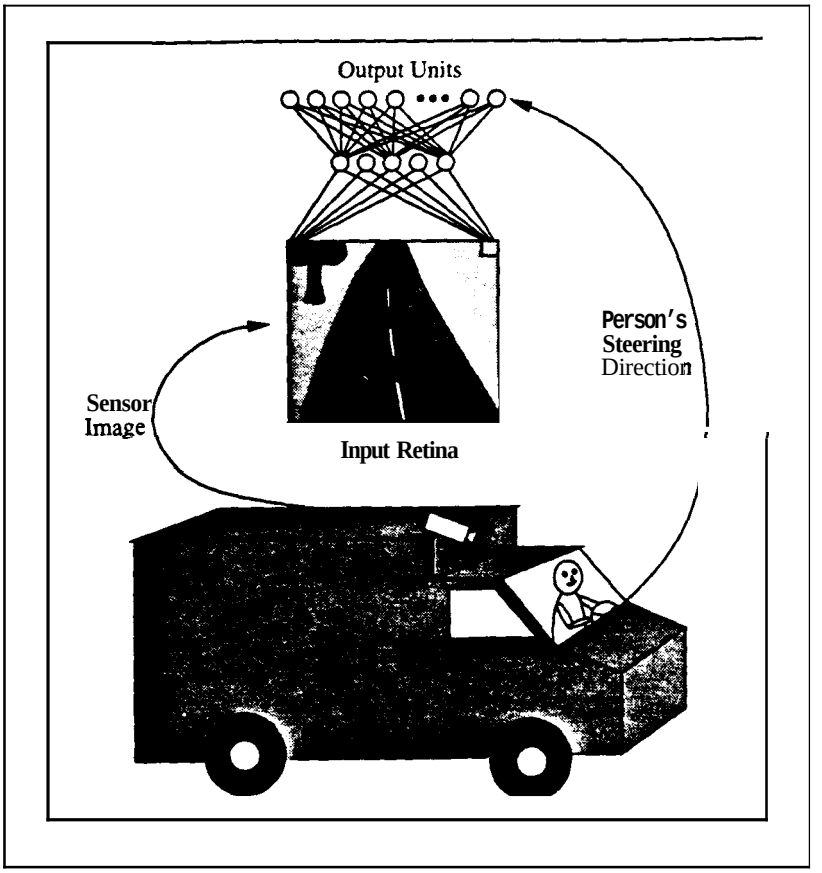

Figure 2. Schematic Representation of Training "on the Fly."
The network is shown images from the on-board sensor and trained to steer in the same direction as the human driver.

Potential Problems

There are two potential problems associated with training a network using live sensor images as a person drives. First, because the person steers the vehicle down the center of the road during training, the network will never be presented with situations where it must recover from misalignment errors. When driving for itself, the network can occasionally stray from the road center; so, it must be prepared to recover by steering the vehicle back to the middle of the road—a capability it might not possess if it never encounters this situation during training. The second problem is that naively training the network with only the current video image and steering direction might cause it to overlearn recent input. If the person drives the Navlab down a stretch of straight road at the end of training, the network will be presented with a long sequence of similar images. This sustained lack of diversity in the training set will cause the network to "forget" what it had learned about driving on curved roads and instead learn to always steer straight ahead.

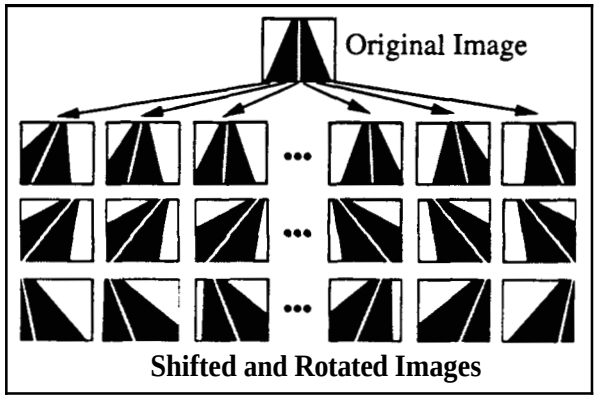

Figure 3. The Single Original Video Image Shifted and Rotated.
Multiple training exemplars are thus created in which the vehicle appears to be at different locations relative to the road.

Both problems associated with training on the fly stem from the fact that back propagation requires training data that are representative of the full task to be learned. The first approach we considered for increasing the training-set diversity was to have the driver swerve the vehicle during training. The idea was to teach the network how to recover from mistakes by showing it examples of the person steering the vehicle back to the road center. However, this approach was deemed impractical for two reasons: First, training while the driver swerves would require turning learning off while the driver steers the vehicle off the road and then back on when he/she swerves back to the road center. Without this ability to toggle the state of learning, the network would incorrectly learn to imitate the person swerving off the road as well as back on. Although possible, turning learning on and off would require substantial manual input during the training process, which we wanted to avoid. The second problem with training by swerving is that it would require swerving in many circumstances to enable the network to learn a general representation, which would be time consuming as well as dangerous when training in traffic.

Solution—Transform the Sensor Image

To achieve sufficient diversity of real sensor images in the training set, without the problems associated with training by swerving, we developed a technique for transforming sensor images to create additional training exemplars. Instead of presenting the network with only the current sensor image and steering direction, each sensor image is shifted and rotated in software to create additional images in which the vehicle appears to be situated differently relative to the environment (figure 3). The sensor's position and orientation relative to the ground plane are known, so precise transformations can be achieved using perspective geometry.

The image transformation is performed by first determining the area of the ground plane that is visible in the original image and the area that should be visible in the transformed image. These areas form two overlapping trapezoids, as illustrated by the aerial view in figure 4. To determine the appropriate value for a pixel in the transformed image, the pixel is projected onto the ground plane and then back projected into the original image. The value of the corresponding pixel in the original image is used as the value for the pixel in the transformed image. One important thing to realize is that the pixel-to-pixel mapping that implements a particular transformation is constant. In other words, based on a planar world, the pixels that need to be sampled in the original image to achieve a specific shift and translation in the transformed image always remain the same. In the actual implementation of the image-transformation technique, ALVINN takes advantage of this constant transformation by precomputing the pixels that need to be sampled to perform the desired shifts and translations. As a result, transforming the original image to change the apparent position of the vehicle simply involves changing the pixel sampling pattern during the image-reduction phase of preprocessing. Therefore, creating a transformed low-resolution image takes no more time than is required to reduce the image resolution to that required by the ALVINN network. Obviously, the environment is not always flat. However, the effects on the image from hills or dips in the road are small enough that the neural network can learn to compensate for them.

Extrapolating Missing Pixels

The less than complete overlap between the trapezoids in figure 4 illustrates the need for one additional step in the image-transformation scheme. The extra step involves determining values for pixels that have no corresponding pixel in the original image. Consider the transformation illustrated in figure 5. To make it appear that the vehicle is situated one meter to the right of its position

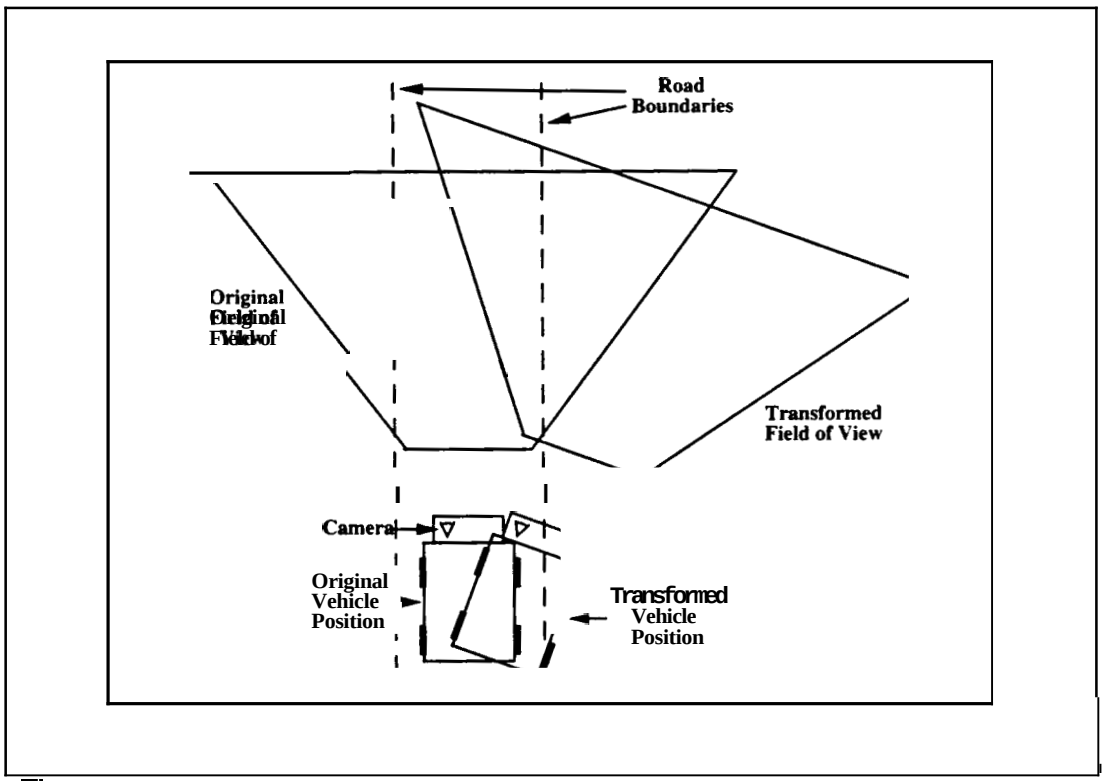

Figure 4. An Aerial View of the Vehicle at Two Different Positions, with the Corresponding Sensor Fields of View.

To simulate the image transformation that would result from such a change in position and orientation of the vehicle, the overlap between the two field-of-view trapezoids is computed and utilized to direct resampling of the original image.

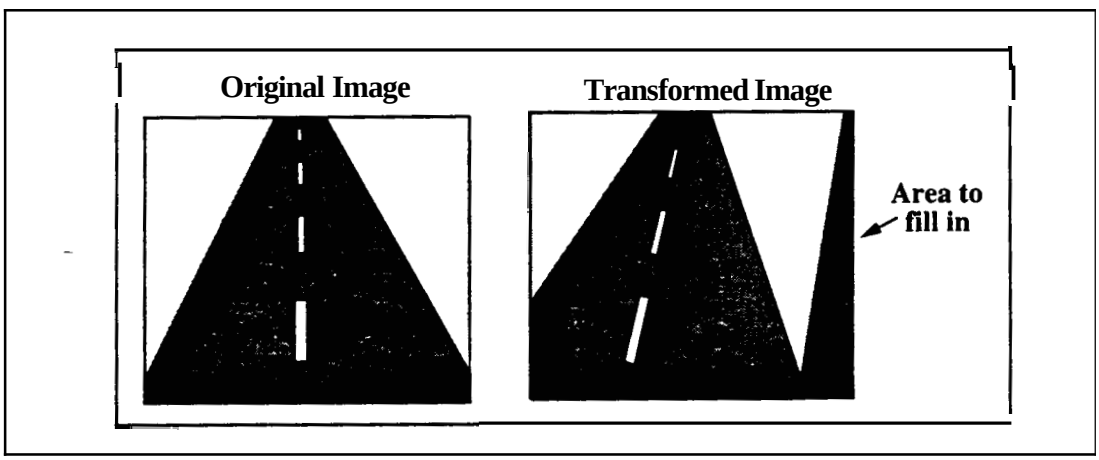

Figure 5. A Schematic Example of an Original Image and a Transformed Image in Which the Vehicle Appears One Meter to the Right of Its Initial Position.

The black region on the right of the transformed image corresponds to an unseen area in the original image. These pixels must be extrapolated from the information in the original image.

in the original image requires not only shifting pixels in the original image to the left but also filling in the unknown pixels along the right edge.

The pixels can be filled using an extrapolation technique that relies on the fact that interesting features (such as road edges and painted lane markers) typically run parallel to the road and, hence, parallel to the vehicle's current direction. With this assumption, to

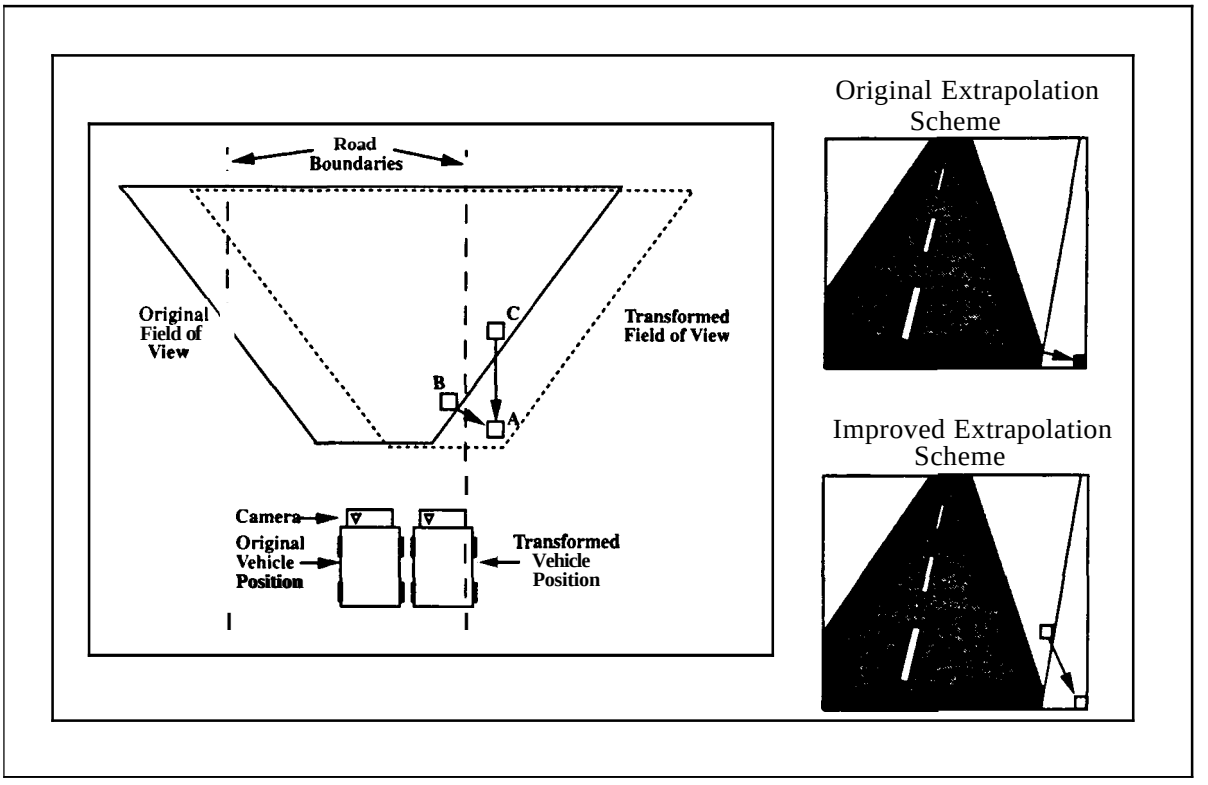

Figure 6. An Aerial View (left) and an Image-Based View (right) of the Two Techniques Used to Extrapolate the Values for Unknown Pixels.

extrapolate a value for the unknown pixel *A* in figure 6, the appropriate ground-plane point to sample from the original image's viewing trapezoid is not the closest point (point B); it is the nearest point in the original image's viewing trapezoid along the line that runs through point *A* and is parallel to the vehicle's original heading (point *C*). See Pomerleau (1993) for more complete details of this technique and the driving performance improvement it led to.

Transforming the Steering Direction

As important as the technique to transform the input images is the method used to determine the correct steering direction for each of the transformed images. The correct steering direction, as dictated by the driver, for the original image must be altered for each of the transformed images to account for the altered vehicle placement. This is done using a simple model called *pure-pursuit steering* (Wallace et al. 1985). In the pure-pursuit model, the "correct" steering direction is the one that will bring the vehicle to a desired location (usually the center of the road) a fixed distance ahead. The idea underlying pure-pursuit steering is illustrated in figure 7. With the vehicle at position *A*, driving for a predetermined distance along the person's current steering arc would bring the vehicle to a target point *T*, which is assumed to be in the center of the road.

After transforming the image with a horizontal shift *s* and rotation θ to make it appear that the vehicle is at point B, the appropriate steering direction according to the pure-pursuit model would also bring the vehicle to target point T. Mathematically, the formula to compute the radius of the steering arc that will take the vehicle from point B to point T is

$$r = (l_2 + d_2)/2d \ ,$$

where *r* is the steering radius, l is the look-ahead distance, and *d* is the distance from point T that the vehicle would end up if driven straight ahead from point B for distance *l*. The displacement d can be determined using the following formula:

$$d = \cos(\theta) \cdot [dp + s + l \cdot \tan(\theta)] \ ,$$

where *dp* is the distance from point T that the vehicle would end up if it drove straight ahead from point *A* for the look-ahead distance *l*, *s* is the horizontal distance from point *A* to point *B*, and θ is the vehicle rotation from point *A* to point *B*. The quantity *dp*

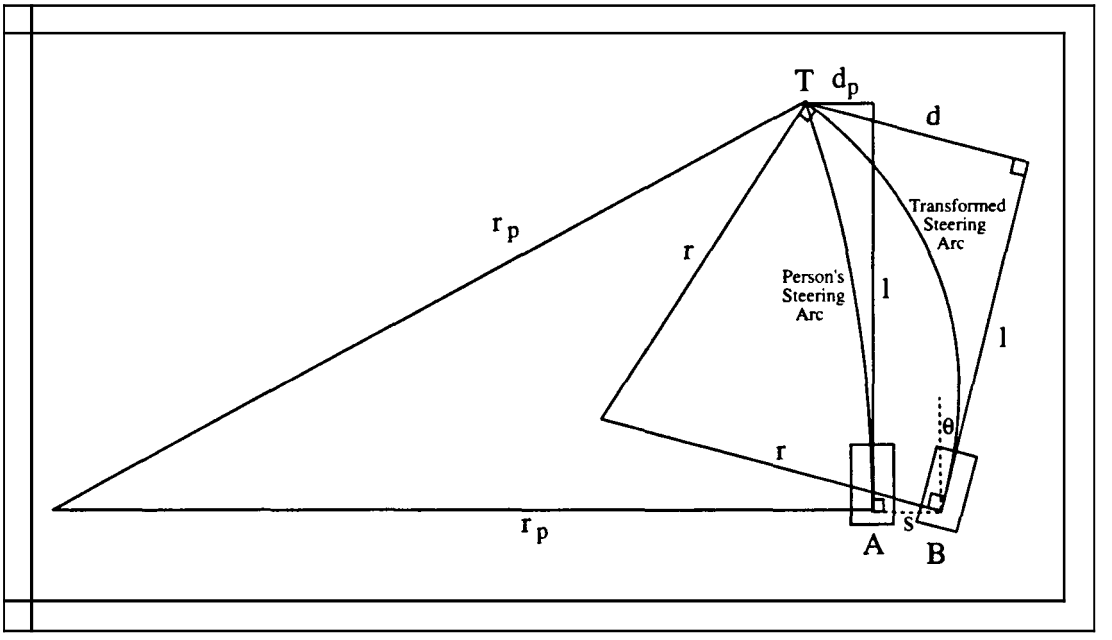

Figure 7. Illustration of the "Pure-Pursuit" Model of Steering.

can be calculated using the following equation:

$$dp = rp - sqrt(rp_2 - l_2) \ ,$$

where *rp* is the radius of the arc the person was steering along when the image was taken. The only remaining unspecified parameter in the pure-pursuit model is l, the distance ahead of the vehicle to select a point to steer toward. Empirically, we have found that over the speed range of 5 to **55** mph, accurate and stable vehicle control can be achieved using the following rule: Look ahead the distance that the vehicle will travel in 2 to 3 seconds.

Interestingly, with this empirically determined rule for choosing the look-ahead distance, the pure-pursuit model of steering is a fairly good approximation of how people actually steer. Research has found that at **50** kilometers/hour, human subjects responded to a 1-meter lateral vehicle displacement, with a steering radius ranging from 511 to **1194** meters (Reid, Solowka, and Billing **1981).** With a look ahead equal to the distance that the vehicle will travel in **2.3** seconds, the pure-pursuit model dictates a steering radius of **594** meters, within the range of human responses. Similarly, human subjects reacted to a I-degree heading error relative to the current road direction, with a steering radius ranging from **719** to **970** meters. Again, using the 2.3-second travel distance for look ahead, the pure-pursuit steering model's dictated radius of **945** meters falls within the range of human responses.

Like the image-transformation scheme, the steering-direction transformation technique uses a simple model to determine how a change in the vehicle's position or orientation would affect the situation. In the image-transformation scheme, a planar-world hypothesis and rules of perspective projection are used to determine how changing the vehicle's position or orientation would affect the sensor image of the scene ahead of the vehicle. In the steering-direction transformation technique, **a** model of how people drive is used to determine how a particular vehicle transformation should alter the correct steering direction. In both cases, the transformation techniques are independent of the driving situation. The person could be driving on a single-lane dirt road or a multilane highway: The transformation techniques would be the same.

Anthropomorphically speaking, transforming the sensor image to create more training images is equivalent to telling the network that "I don't know what features in the image are important for determining the correct direction to steer, but whatever they are, here are some other positions and orientations you might see them in." Similarly, the technique for transforming the steering direction for each of these new training images is equivalent to telling the network that "whatever the important features are, if you see them in this new position and orientation, here is how your response should change." Because it does not rely on a strong model of

what important image features look like, instead acquiring this knowledge through training, the system is able to drive in a wide variety of circumstances, as is seen later.

These weak models are enough to solve the two problems associated with training in real time on sensor data. Specifically, using transformed training patterns allows the network to learn how to recover from driving mistakes that it would not otherwise encounter as the person drives. Also, overtraining on repetitive images is less a problem because the transformed training exemplars maintain variety in the training set.

... using transformed training patterns allows the network to learn how to recover from driving mistakes that it would not otherwise encounter as the person drives.

Adding Diversity through Buffering

As additional insurance against the effects of repetitive exemplars, the training-set diversity is further increased by maintaining a buffer of previously encountered training patterns. When new training patterns are acquired through digitizing and transforming the current sensor image, they are added to the buffer, and older patterns are removed. We experimented with four techniques for determining which patterns to replace. The first is to replace oldest patterns first. With this scheme, the training pattern buffer represents a history of the driving situations encountered recently. However, if the driving situation remains unchanged for a period of time, such as during an extended right turn, the buffer will lose its diversity and become filled with right-turn patterns. The second technique is to randomly choose old patterns to be replaced by new ones. With this technique, the laws of probability help ensure somewhat more diversity than the oldest pattern-replacement scheme, but the buffer will still become biased during monotonous stretches.

The next solution we developed to encourage diversity in the training set was to replace the patterns on which the network was making the lowest error, as measured by the sum-squared difference between the network's output and the desired output. The idea was to eliminate from the training set the patterns the network was performing best on and to leave in the training set the images the network was still having trouble with. The problem with this technique is that the human driver doesn't always steer in the correct direction. Occasionally, he or she might have a momentary lapse of attention and steer in an incorrect direction for the current situation. If a training exemplar was collected during this momentary lapse, under this replacement scheme, it will remain in the training buffer for a long time because the network will have trouble learning a steering response to match the person's incorrect steering command. In fact, with this replacement technique, the only way the pattern would be removed from the training set would be if the network learned to duplicate the incorrect steering response-obviously, not a desired outcome. We considered replacing both the patterns with the lowest error and the patterns with the highest error but decided against it because high network error on a pattern might also result in novel input image with a correct response associated with it. A better method to eliminate this problem is to add a random replacement probability to all patterns in the training buffer. This method ensures that ever? if the network never learns to produce the same steering response as the person on an image, the image will eventually be eliminated from the training set.

Although this augmented lowest-error replacement technique did a reasonable job of maintaining diversity in the training set, we found a more straightforward way of achieving the same result. To make sure the buffer of training patterns does not become biased toward one steering direction, we added a constraint to ensure that the mean steering direction of all the patterns in the buffer remains as close to straight ahead as possible. When choosing the pattern to replace, we selected the pattern whose replacement will bring the average steering direction closest to straight. For example, if the training-pattern buffer had more right turns than left, and a left-turn image was just collected, one of the right-turn images in the buffer would be chosen for replacement to move the average steering direction toward straight ahead. If the buffer already had a straight-ahead average steering direction, then an old pattern requiring a steering direction similar to the new one would be replaced to maintain the buffer's unbiased nature. By actively compensating for steering bias in the training buffer, the network never learns to consistently favor one steering direction over another. This active bias compensation is a way to build into the network a known constraint about steering: In the long run, right and left turns occur with equal frequency.

Details

The final details required to specify the training on-the-fly process are the number and magnitude of transformations to use for

training the network. The following quantities were determined empirically to provide sufficient diversity to allow networks to learn to drive in a wide variety of situations. The original sensor image is shifted and rotated 14 times using the technique described earlier to create 14 training exemplars. The size of the shift for each of the transformed exemplars is chosen randomly from the range -0.6 to +0.6 meters, and the amount of rotation is chosen from the range –6.0 to +6.0 degrees. In the image formed by the camera on the Navlab, which has a 42-degree horizontal field of view, an image with a maximum shift of 0.6 meters results in the road shifting approximately one-third the way across the input image at the bottom.

Before the randomly selected shift and rotation are performed on the original image, the steering direction that would be appropriate for the resulting transformed image is computed using the formulas given previously. If the resulting steering direction is sharper than the sharpest turn representable by the network's output (usually a turn with a 20-meter radius), then the transformation is disallowed, and a new shift distance and rotation magnitude are randomly chosen. By eliminating extreme and unlikely conditions from the training set, such as when the road is shifted far to the right and the vehicle is heading sharply to the left, the network is able to devote more of its representation capacity to handling plausible scenarios.

The 14 transformed training patterns, along with the single pattern created by pairing the current sensor image with the current steering direction, are inserted into the buffer of 200 patterns using the replacement strategy described earlier. After this replacement process, one iteration of the back-propagation algorithm are performed on the 200 exemplars to update the network's weights, using a learning rate of 0.01 and a momentum of 0.8. The entire process is then repeated. The network requires approximately 100 iterations through this digitize-replace-train cycle to learn to drive in the domains that have been tested. At 2.5 seconds a cycle, training takes approximately 4 minutes of human driving over a sample stretch of road. During the training phase, the person drives at approximately the speed at which the network will be tested, which ranges from 5 to 55 mph.

Performance Improvement Using Transformations

The performance advantage that this technique of transforming and buffering training patterns offers over the more naive methods of training on real sensor data is illustrated in figure 8. This graph shows the vehicle's displacement from the road center that was measured as three different networks drove at 4 mph over a 100-meter section of a single-lane paved bike path that included a straight stretch and turns to the left and right. The three networks were trained on a 150-meter stretch of the path that was disjoint from the test section and that ended in an extended right turn.

The first network, labeled -trans -buff, was trained using just the images coming from the video camera. That is, during the training phase, an image was digitized from the camera and fed into the network. One iteration of back propagation was performed on the training exemplar and then the process was repeated. The second network, labeled +trans -buff, was trained using an image that was digitized from the camera and then transformed 14 times to create 15 new training patterns, as described earlier. One iteration of back propagation was performed on each of these 15 training patterns, and then the process was repeated. The third network, labeled +trans +buff, was trained using the same transformation scheme as the second network but with the addition of the image-buffering technique described earlier to prevent overtraining on recent images.

Note that all three networks were presented with the same number of images. The transformation and buffering schemes did not influence the quantity of data that the networks were trained on, only their distribution. The -trans -buff network was trained on closely spaced actual video images. The +trans -buff network was presented with 15 times fewer actual images, but its training set also contained 14 transformed images for every "real" one. The +trans +buff network collected even fewer live images because it performed a training iteration through its buffer of 200 patterns before digitizing a new one.

The accuracy of each of the three networks was determined by manually measuring the vehicle's lateral displacement relative to the road center as each network drove. The network trained on only the current video image quickly drove off the right side of the road, as indicated by its rapidly increasing displacement from the road center. The problem was that the network overlearned the right turn at the end of training and became biased toward turning right. Because of the increased diversity provided by the image-transformation scheme, the second network per-

The problem was that the network overlearned the right turn at the end of training and became biased toward turning right.

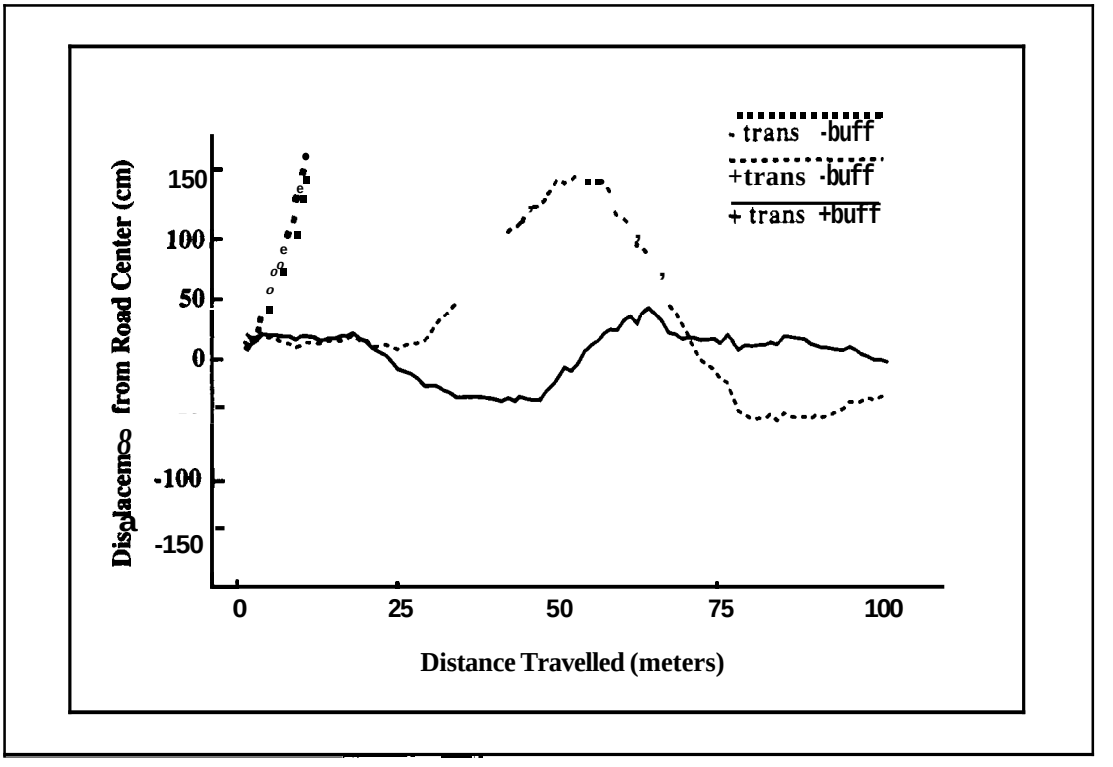

Figure 8. Vehicle Displacement from the Road Center as the Navlab Was Driven by Networks Trained Using Three Different Techniques.

formed much better than the first. It was able to follow the entire test stretch of road. However, it still had a tendency to steer too much to the right, as illustrated in figure 8 by the vehicle's positive displacement over most of the test run. In fact, the mean position of the vehicle was 28.9 centimeters right of the road center during the test. The variability of the errors made by this network was also quite large, as illustrated by the wide range of vehicle displacement in the +trans -buff graph in figure 8. Quantitatively, the standard deviation of this network's displacement was 62.7 centimeters.

The addition of buffering previously encountered training patterns eliminated the right bias in the third network and also greatly reduced the magnitude of the vehicle's displacement from the road center, as evidenced by the +trans +buff graph in figure 8. When the third network drove, the average position of the vehicle was 2.7 centimeters right of center, with a standard deviation of only 14.8 centimeters. This vehicle position error represents a 423-percent improvement in driving accuracy.

A separate test was performed to compare the steering accuracy of the network trained using both transformations and buffering with the steering accuracy of a human driver. This test was performed over the same stretch of road as the previous test, but the road was less obscured by fallen leaves in this test, resulting in better network performance. Over three runs, with the network driving at 5 mph along the 100-meter test section of road, the average position of the vehicle was 1.6 centimeters right of center, with a standard deviation of 7.2 centimeters. Under human control, the average position of the vehicle was 4.0 centimeters right of center, with a standard deviation of 5.47 centimeters. The average distance the vehicle was from the road center while the person drove was 5.70 centimeters. It appears that the human driver, although more consistent than the network, had an inaccurate estimate of the vehicle's center line; therefore, it drove slightly right of the road center. Studies of human driving performance have found similar steady-state errors and variances in vehicle lateral position. Researchers have found that consistent displacements of as much as 7 centimeters were not uncommon when people drove on highways (Blaaum 1982). Also for highway driving, standard deviations in lateral error as great as 16.6 centimeters have been reported.

Network Confidence

In addition to learning to produce the correct output steering direction, ALVINN can also estimate its confidence in the steering output. This ability is important because it gives outside, typically symbolic, knowledge sources insight into how the neural system is performing. Because the confidence value is a measure of ALVINN's familiarity with the current road, it can be used to determine if proper driving performance is possible. In another context, this same information can be used as a metric to determine the existence of roads in input images. This ability becomes especially important if tactical driving maneuvers, like those described in Tactical Driving, are to be executed.

The confidence metric that is primarily used is called ***input reconstruction reliability estimation*** (IRRE). IRRE is a measure of the familiarity of the input image to the neural network. In IRRE, the network's internal representation is used to reconstruct the input pattern being presented. The more closely the reconstructed input matches the actual input, the more familiar the input and, hence, the more reliable the network's response.

IRRE utilizes an additional set of output units to perform input reconstruction, as depicted in figure 1. This second set of output units is half the size of the input retina—15 rows by 16 columns. The desired activation for each of these additional output units is the average of the activation on four corresponding input units. For example, IRRE unit (0,0) contains the average activation of input units (0,0), (0,1), (1,0), and (1,1). In essence, these additional output units turn the network into an autoencoder.

The network is trained using back propagation to produce both the correct steering response on the steering output units and reconstruct the input image as accurately as possible on the reconstruction output. During testing, images are presented to the network, and activation is propagated forward through the network to produce a steering response and a reconstructed input image. The reliability of the steering response is estimated by computing the correlation coefficient between the activation levels of units in the actual input image and the reconstructed input image. The higher the correlation between the two images, the more reliable the network's steering response is estimated to be.

Results and Comparison

The success of the ALVINN system is demonstrated by the range of situations in which it has successfully driven. The training on-the-fly scheme gives ALVINN a flexibility that is novel among autonomous navigation systems. It has allowed us to successfully train individual networks to drive in a variety of situations, including a single-lane dirt access road, a single-lane paved bicycle path, a two-lane suburban neighborhood street, and a lined two-lane highway (figure 9). Using other sensor modalities as input, including laser range images and laser reflectance images, individual ALVINN networks have been trained to follow roads in total darkness, avoid collisions in obstacle-rich environments, and follow alongside railroad tracks. ALVINN networks have driven without intervention for distances as great as 90 miles. In addition, because determining the steering direction from the input image merely involves a forward sweep through the network, the system is able to process 15 images a second, allowing it to drive as fast as 55 mph.

Figure 9. Video Images Taken on Three of the Road Types ALVINN Modules Have Been Trained to Handle.

They are, from left to right, a single-lane dirt access road, a single-lane paved bicycle path, and a lined two-lane highway.

The level of flexibility across driving situations exhibited by ALVINN would be difficult to achieve without learning. It would require the programmer to (1) determine what features are important for the particular driving domain, (2) program detectors (using statistical or symbolic techniques) for finding these important features, and (3) develop an algorithm for determining which direction to steer from the location of the detected features. As a result, although hand-programmed systems have been developed to drive in some of the individual domains that ALVINN can handle (Kluge 1993; Crisman 1990; Turk et al. 1988; Dichmanns and Zapp 1987), none have duplicated ALVINN's flexibility.

ALVINN is able to learn, for each new domain, what image features are important, how to detect them, and how to use their position to steer the vehicle. Analysis of the

hidden unit representations developed in different driving situations shows that the network forms detectors for the image features that correlate with the correct steering direction. When trained on multilane roads, the network develops hidden-unit feature detectors for the lines painted on the road, but in single-lane driving situations, the detectors developed are sensitive to road edges and road-shaped regions of similar intensity in the image.

This ability to use arbitrary image features can be problematic, for example, when ALVINN was trained to drive on a poorly defined dirt road with a distinct ditch on its right side. The network had no problem learning and then driving autonomously in one direction, but when driving the other way, the network was erratic, swerving from one side of the road to the other. After analyzing the network's hidden representation, the reason for its difficulty became clear. Because of the poor distinction between the road and the nonroad, the network had developed only weak detectors for the road itself and instead relied heavily on the position of the ditch to determine the direction to steer. When tested in the opposite direction, the network was able to keep the vehicle on the road using its weak road detectors but was unstable because the ditch it had learned to look for on the right side was now on the left. Individual ALVINN networks have a tendency to rely on any image feature consistently correlated with the correct steering direction. Therefore, it is important to expose them to a wide enough variety of situations during training to minimize the effects of transient image features.

*Because of the **poor** distinction between the road and the nonroad, the network had developed only weak detectors for the road itself and instead relied heavily on the position of the ditch to determine the direction to steer.*

Experience has shown that it is more efficient to train several domain-specific networks for circumstances such as one-lane versus two-lane driving than to train a single network for all situations. To prevent this network specificity from reducing ALVINN's generality, we have implemented connectionist and nonconnectionist techniques for combining networks trained for different driving situations. Using a simple rule-based priority system similar to the subsumption architecture, we combined a road-following network and an obstacle-avoidance network (Brooks **1986).** The road-following network used video camera input to follow a single-lane road. The obstacle-avoidance network used laser range-finder images as input. It was trained to swerve appropriately to prevent a collision when confronted with obstacles and to drive straight when the terrain ahead was free of obstructions. The arbitration rule gave priority to the road-following network when determining the steering direction, except when the obstacle-avoidance network outputs a sharp steering command. In this case, the urgency of avoiding an imminent collision takes precedence over road following, and the steering direction was determined by the obstacle-avoidance network. Together, the two networks and the arbitration rule made up a system capable of staying on the road and swerving to prevent collisions.

To facilitate other rule-based arbitration techniques, we added a nonconnectionist module to ALVINN that maintains the vehicle's position on a map (Pomerleau, Gowdy, and Thorpe **1992).** Knowing its map position allows ALVINN to use arbitration rules such as "when on a stretch of two-lane highway, rely primarily OR the two-lane highway network." This symbolic mapping module also allows ALVINN to make high-level, goal-oriented decisions such as which way to turn at intersections and when to stop at a predetermined destination.

ALVINN Discussion

A truly autonomous mobile vehicle must cope with a wide variety of driving situations and environmental conditions. As a result, it is crucial that an autonomous navigation system possess the ability to adapt to novel domains. Supervised training of a connectionist network is one means of achieving this adaptability. However, teaching an artificial neural network to drive based on a person's driving behavior presents a number of challenges. Prominent among these challenges is the need to maintain sufficient variety in the training set to ensure that the network. develops a sufficiently general representation of the task. Two characteristics of real sensor data collected as a person drives that make training-set variety difficult to maintain are (1) temporal correlations and **(2)** the limited range of situations encountered. Extended intervals of nearly identical sensor input can bias a network's internal representation and reduce driving accuracy. The human trainer's high degree of driving accuracy severely restricts the variety of situations covered by the raw sensor data.

The techniques for training on the fly described earlier solve these difficulties. The key idea underlying training on the fly is that a model of the process generating the live training data can be used to augment the training set with additional realistic patterns. By modeling both the imaging process and

the steering behavior of the human driver, training on the fly generates patterns with sufficient variety to allow artificial neural networks to learn a robust representation of individual driving domains. The resulting networks are capable of driving accurately in a wide range of situations.

Tactical Driving

The basic ALVINN system described earlier was initially used as just a stand-alone lane-keeping, or lane-departure warning, system. However, beginning in early **1994,** when Jochem began his Ph.D. dissertation, the system began to evolve into more. Jochem's goal was to use a robust lane-keeping system to explore tactical driving. The challenge was to create a system that maintained the performance of the existing lane-keeping system and added the ability to execute tactical driving tasks such as changing lanes and negotiating intersections.

The ability to execute tactical actions was becoming important for several reasons: First, lane-keeping systems had matured to a level where tactical control algorithms, previously demonstrated only in simulation, could realistically be tested on vehicles that operate in the real world. Second, tactical reasoning algorithms had improved to the point where operation in the real world was becoming feasible. Finally, specifications and concepts of new intelligent-vehicle programs in Europe, Asia, and the United States required the ability to execute tactical actions.

Although there are many ways to add tactical functions to an autonomous navigation system, the most desirable solution is to develop. a robust, lane-keeper–independent control scheme that provides the functions to execute tactical actions. These functions can be provided through intelligent control of the visual information presented to the lane-keeping system. Specifically, the techniques described in the following sections use the inherent lane-keeping ability of ALVINN to perform tactical driving tasks.

Although ALVINN has demonstrated robust lane-keeping performance in a wide variety of situations, it is generally limited to this task because of its lack of geometric models, which are typically required to execute tactical actions. Grafting geometric reasoning onto a nongeometric base would be difficult and would dilute ALVINN's capabilities. A much better approach was to leave the basic neural network intact, preserving its real-time performance and generalization capabilities, and apply geometric transformations to the input image and the output steering vector. These transformations form a new set of tools and techniques called *virtual* active ***vision. Vir-*** tual because all the techniques use artificially created imaging sensors that can be manipulated to suit the needs of the task and ***active*** because the techniques move sensors to locations where the images they create will enhance system performance. The idea of virtual active vision—and, specifically, the virtual camera—became the basis for Jochem's dissertation **(1996)** and served as the tool that allowed ALVINN to evolve into more than just a lane-keeping system.

A virtual camera is simply an artificial imaging sensor that can be placed at any location and orientation in the world-reference frame. It creates images using a technique similar to the one used to create transformed images during ALVINN's training—resampling the actual camera image to make it appear as if the camera is situated at a different position or orientation. **By** knowing the location of both the actual and the virtual camera and assuming a flat-world model, accurate image reconstructions can be created from the virtual camera location.

For a particular tactical driving task, the location of the virtual camera is chosen so that it creates images from relevant parts of the scene and from the same vantage as the images that were used to train the network. Because the images look familiar, the network will respond properly. Virtual camera views from many orientations have been created using images from several different actual cameras. The images produced by these views have proven to be accurate enough for the ALVINN system to navigate successfully. Figure **10** shows some typical virtual camera views and the images that they create.

Descriptions of the application of virtual active vision tools to two tactical driving tasks—(1) lane changing and **(2)**intersection navigation—are presented in the next subsections. For both tasks, the basic ALVINN system is used, but by intelligently controlling its input and interpreting its output, enhanced tactical functions were possible.

Lane Changing

There has been a significant amount of research published describing how people change lanes as well as identifying theoretically optimal control strategies that could be used to autonomously control a vehicle in a lane-change maneuver (Grupen et al. **1995;** Hatipoglu, Ozguner, and Unyelioglu **1995;**

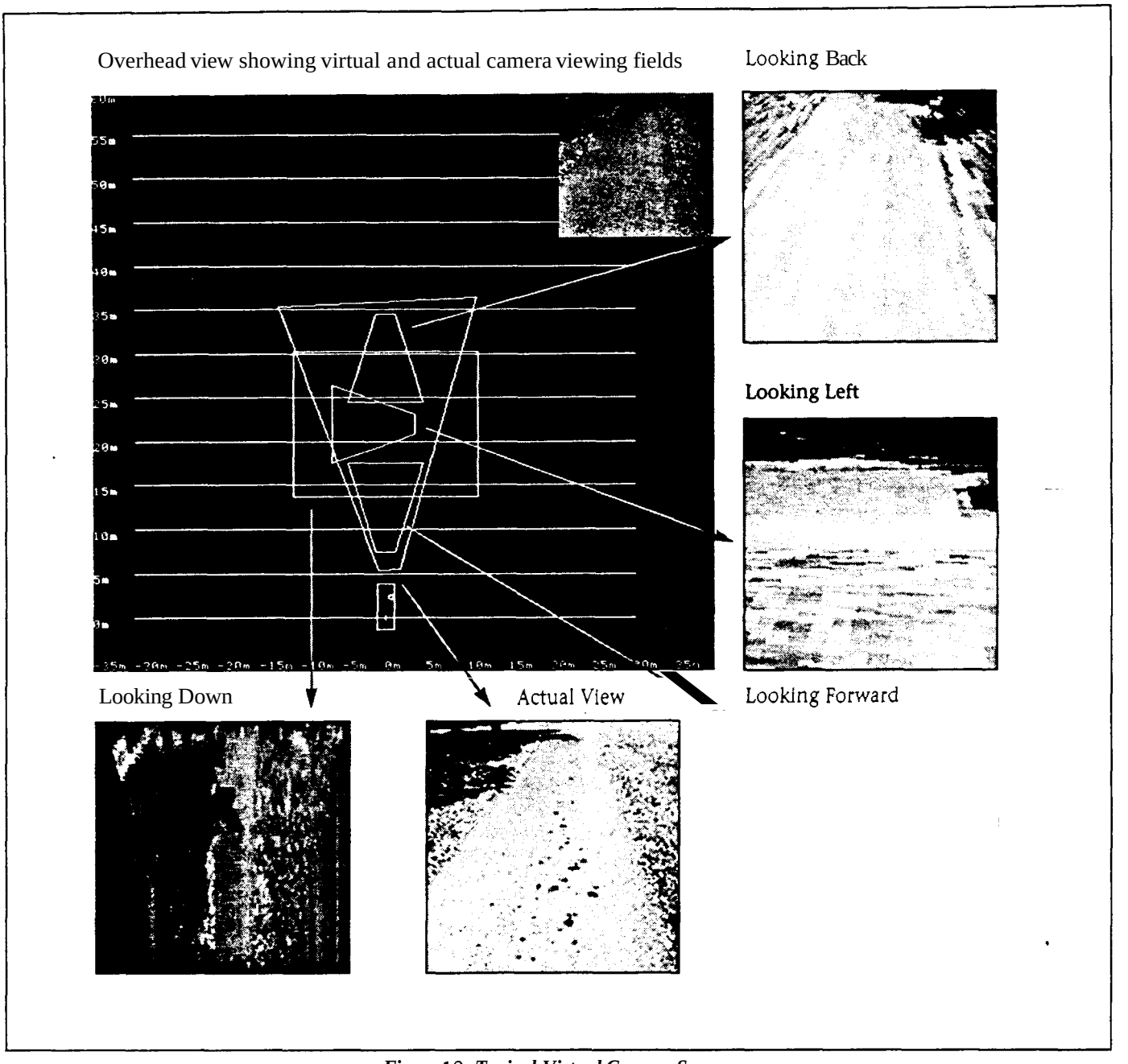

Figure 10. Typical Virtual Camera Scenes.

Nelson ***1989).*** Unfortunately, only a few of these researchers had the facilities or equipment to test their results outside the lab (Jochem ***1996;*** Rosenblum ***1995;*** Behringer, Holt, and Dickmanns ***1992).***

It is obvious to think about lane changing only in terms of moving a vehicle from one driving lane to another, adjacent lane. Although this task is the most evident, there are others similar in nature that require a subset of, or a minor extension to, the capabilities required for lane changing. In addition to presenting techniques for lane changing, this section introduces algorithms for exit-ramp detection and traversal and short-distance obstacle-avoidance maneuvering.

Technically, these tasks are important because they represent tactical driving tasks that have not been studied as closely as low-level tasks such as lane keeping and obstacle detection. To be successfully accomplished, all the tasks require a sequence of actions to be taken—a plan. This type of behavior is difficult to coax from low-level, reactive sys-

tems such as ALVINN, which are characterized by fast response to features in their surroundings. Additionally, plans are often rigid and difficult to integrate into autonomous systems that function in the real world. The algorithms presented in this section circumvent these problems by framing the task plan in a way that preserves the flexibility and robustness of the low-level lane-keeping system yet exhibits enough control to permit successful execution.

Lane change and related functions are implemented using active placement and control of virtual cameras, intelligent interpretation of the lane-keeping system's response to the images created from the virtual cameras, and a simple road model that guides virtual camera placement and vehicle control. The algorithms are not strictly tied to any specific lane-keeping system; they only require that the system take images as input and produce a point on the road to drive over and a measure of its internal confidence in this point as output. When implemented using ALVINN, the techniques have been able to autonomously navigate one of our test-bed vehicles, the Navlab 5 (Jochem et al. 1995), between lanes of a rural interstate highway, onto exit ramps, and around obstacles.

As mentioned earlier, a simple road model is needed to guide the lane-change maneuvers. The model that the techniques use is that the centers of adjacent lanes of the road have constant separation, implying that the lanes are parallel and have a constant width.

The most successful method Jochem developed to change lanes is called the ***dual-view lane-transition method*** (DVLT) Uochem 1996). In the DVLT algorithm, ALVINN is presented with images from two virtual cameras, one tracking each lane. ALVINN's responses to the images from these two views are smoothly merged to move the vehicle from one lane to the next. The placement of the views is controlled by ALVINN's response on the two virtual images as well as by a high-level control algorithm that biases the system to transition from one lane to the other.

This technique is a bottom-up approach to lane changing. Although it can use input from high-level modules to initiate the maneuver, the geometry of the situation is what drives this method: The system locates the center of both lanes and then moves the vehicle based on these locations.

The DVLT method requires networks that are trained for the driving and destination lanes. Initially, only one view, called the ***source-lane view*** (SLV), and the associated network, are used to control the vehicle in the driving lane. When a DVLT is initiated, the road model is used to laterally offset a second view, called the ***destination-lane view*** (DLV), so that it is centered over the destination lane. The network that was trained to drive in the destination lane is associated with the DLV (figure 11).

After the DLV has been created, images are generated from both the SLV and the DLV and are passed to their respective networks, which produce an output steering displacement along with an **IRRE** confidence value. Both output displacements are converted to vehicle-relative points. These points, called ***lane*** center ***points*** (LCPs), specify where ALVINN believes the center of each lane is located in front of the vehicle.

The LCPs are used to calculate the ***modified look-ahead point*** (MLP). The MLP is the point that the vehicle will actually drive toward. The MLP is between the two LCPs, along the line connecting them. The distance between the MLP and either LCPs is related to the step in the DVLT process. For example, in DVLT experiments that used 16 iterations to transition between lanes, the first MLP would be 1/16 the total distance (along the line connecting the LCPs) away from the driving lane LCP and 15/16 from the destination LCP. On the second step, the MLP would be 2/16 and 14/16 away, respectively.

Because the vehicle is instructed to steer toward the MLP, the virtual views become misaligned with their respective lanes, and their position must be updated. If the vehicle is moving to the left, both views are repositioned to the right. This process is continued until the vehicle has transitioned completely into the destination lane (figure 12). The black dot in each of the diagrams in figure 12 is the MLP. The combination of moving toward an MLP that is continually closer to the destination lane center and updating the location of the virtual camera views results in a smooth, controlled transition.

During the lane transition, the IRRE confidence metric and the constant lane-separation constraint are used to determine if the system is confident in its current driving ability. For the transition to continue, the IRRE confidence measure for each image must be above a threshold value, typically 0.40, and the distance between the LCPs is required to be within 40 percent of the lane-separation distance, specified by the lane model.

Figure 13 shows two images taken at the beginning and the end of a left-to-right

...plans are often rigid and difficult to integrate into autonomous systems that function in the real world.

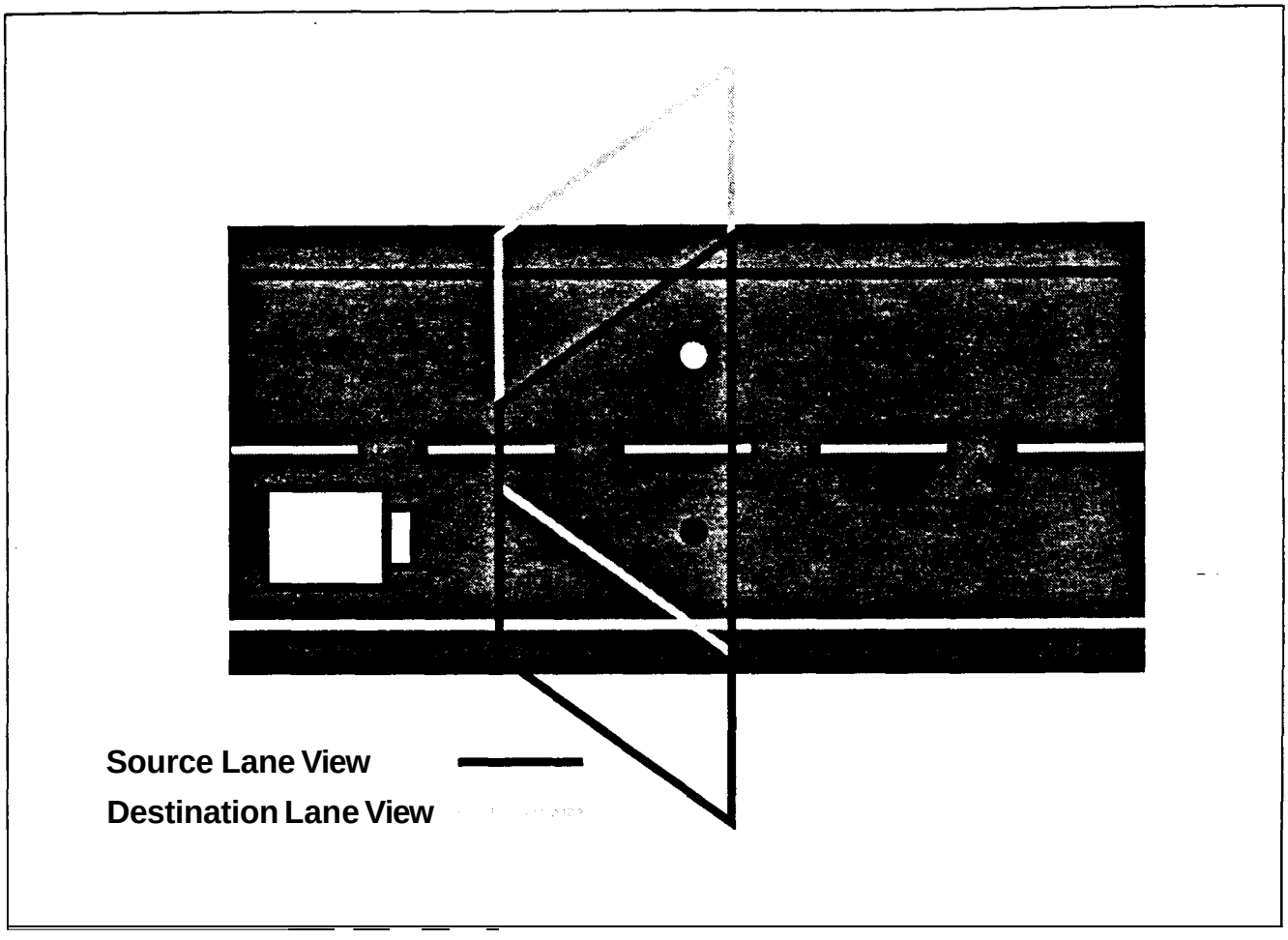

Figure 11. Initial Virtual View Placement for the Dual-View Lane-Transition Technique.

DVLT. In the top image, the vehicle is still in the initial driving lane. The SLV footprint is the darker trapezoid immediately in front of the vehicle. The preprocessed image associated with the SLV is shown in the lower left corner. The DLV footprint is also visible as the lighter trapezoid, offset to the right of the SLV footprint. The preprocessed image associated with it is shown to the right of the preprocessed image from the SLV. Just above each preprocessed image is ALVINN's driving response to the image. For each view, the driving response is almost the same, indicating that the constant separation distance road model and virtual camera imaging were accurate. Also shown in front of the vehicle is a grey dot that represents the MLP. Next to each preprocessed image is a bar graph representation of the IRRE confidence value associated with each. For both the SLV and the DLV, the confidence is high—above 0.70.

The image on the bottom was taken as the vehicle approached the destination lane. In this image, the DLV footprint is almost centered in front of the vehicle, and the original SLV is offset to the left. The preprocessed images are shown in the same location, along with ALVINN's driving responses and IRRE confidence values. The responses and confidence values are similar to those in the top image, even though the vehicle is now in the destination lane.

All 42 DVLTs that were logged to verify this lane-transition method were successfully completed. One-half the transitions were left lane-to-right lane transitions; the rest were right to left. The results of the experiments are shown in figure 14. (The average distance for completion of a DVLT was 138 meters, and the average vehicle speed was about 21.9 meters/second.)

There are two characteristics to notice in figure 14. First, the lane transitions are symmetric across the center line of the road as well as with respect to the time in the lane transition. Second, the DVLT method doesn't

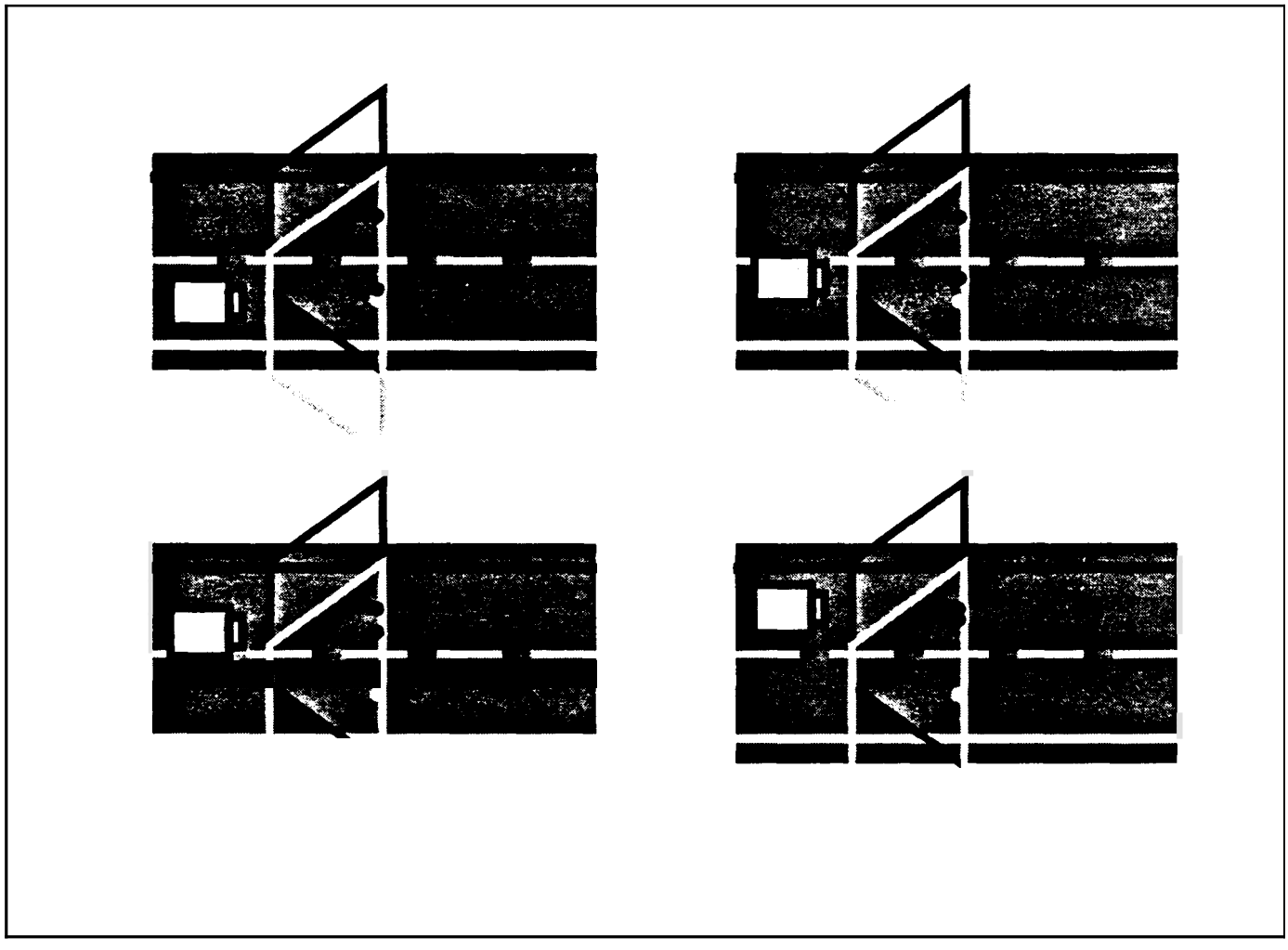

Figure 12. Movement of the Vehicle and the Views in the Dual-View Lane-Transition Method.

relinquish control until the vehicle is at the proper-driving position in the destination lane. Also notice, however, that at the end of the transition, the vehicle trajectory is still moving somewhat outward. A small amount of overshoot did occur and can be attributed to the existing lateral movement of the vehicle coupled with the moderately long response time that results from the large look-ahead distance required for highway driving. Empirically, the **DVLT** method yielded smooth and realistic lane changes.

Figure **15** shows the average IRRE confidence values for the right- and left-lane ALVINN networks during right-to-left and left-to-right DVLTs. The IRRE confidence values produced by an ALVINN network increase as the vehicle moves into the lane for which it was trained and decrease as the vehicle moves out of it. In both graphs, though, the IRRE values remain well above the low-confidence threshold. One cause of the increasing or decreasing confidence values is the extreme location of the SLV and the DLV. Near the start and the end of the transition, because each view remains centered over the proper lane, the actual camera viewing field might not overlay with the virtual camera viewing field. Although the missing virtual camera pixels are filled with the best actual camera pixel, the image is not quite consistent with what the network was trained with.

Exit-Ramp Detection and Traversal

A scenario that requires a vehicle-control scheme similar to that of lane changing is exit-ramp detection and traversal. **A** key difference is that the exact location of the exit ramp is not usually known with enough precision to blindly move the vehicle onto it; therefore, it must first be located. To accomplish this task using the ALVINN sys-

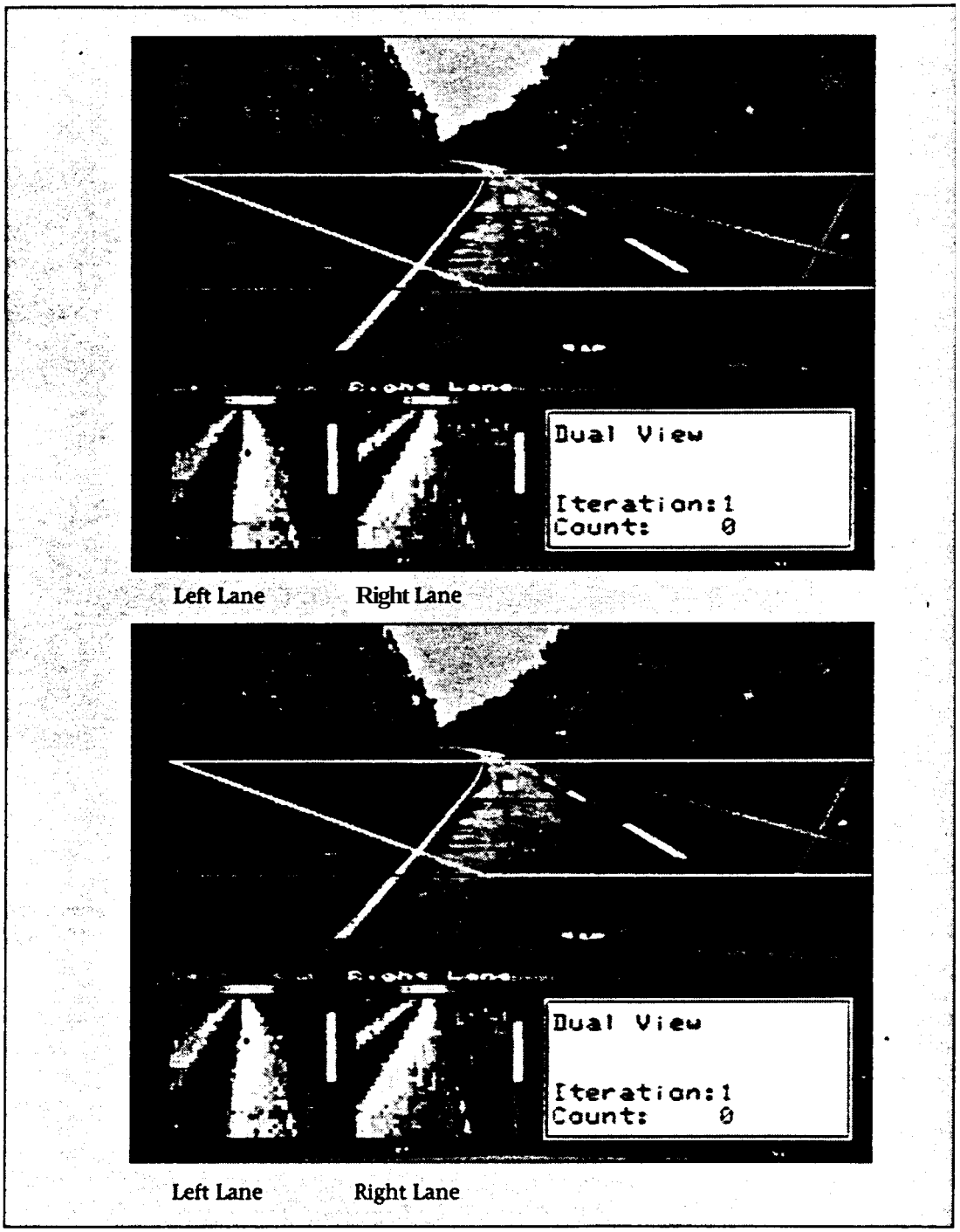

Figure 13. Dual-View Lane-Transition Processing.

tem, the following algorithm was developed.

Some distance before the exit lane begins, ALVINN receives a signal from another knowledge source, such as a map, informing it that the exit is approaching. Information about the exit lane's location relative to the vehicle and the lane type is passed to ALVINN. The information is used to create an appropriate virtual camera view, the ***exit*** view, and associate the correct network with it. In addition to being laterally offset from the current view being used to drive, the exit view is also shifted forward by 10 meters. This shift is made so that the exit lane can be detected before the vehicle is adjacent to it; this shift is important because in some situations, the driving-

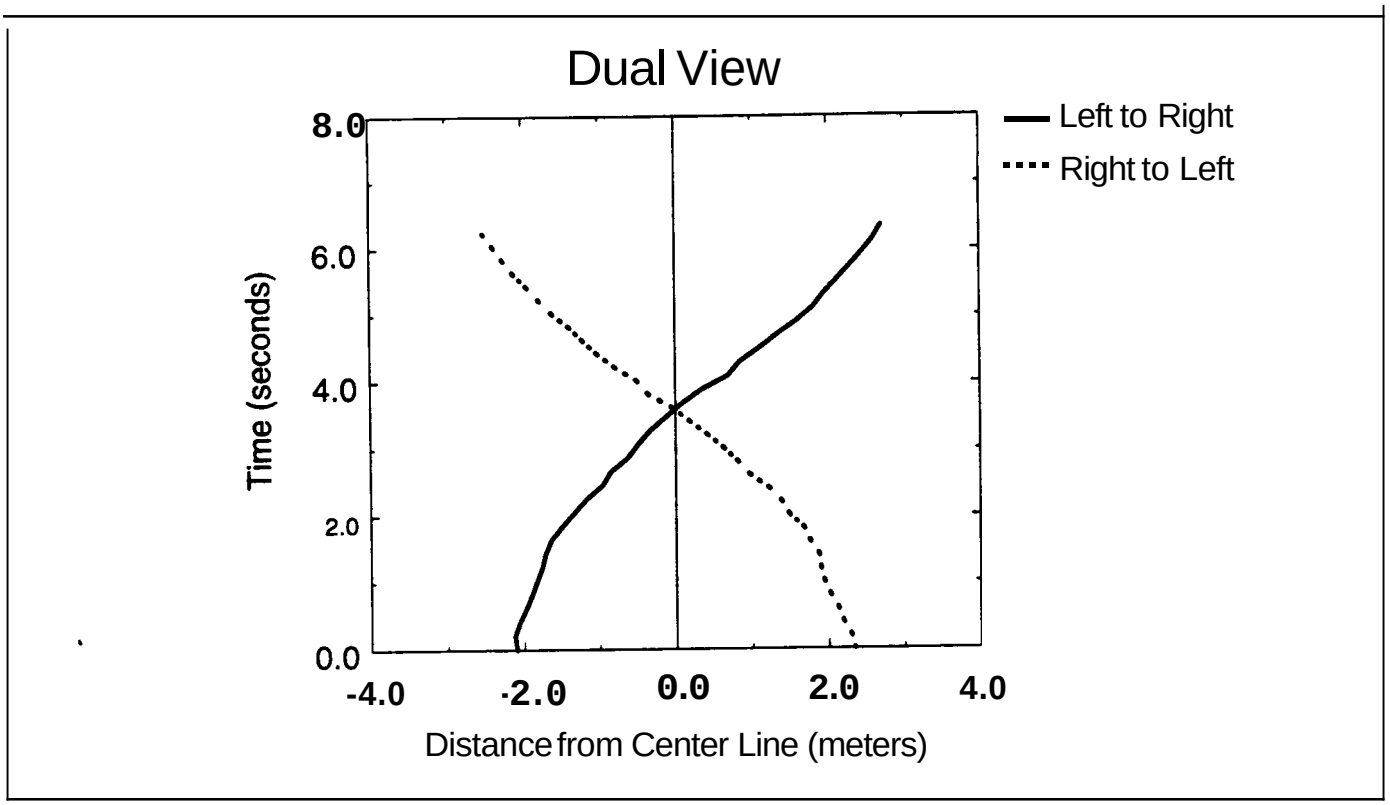

Figure 14. Average Vehicle Path during Dual-Vim Lane Transitions.

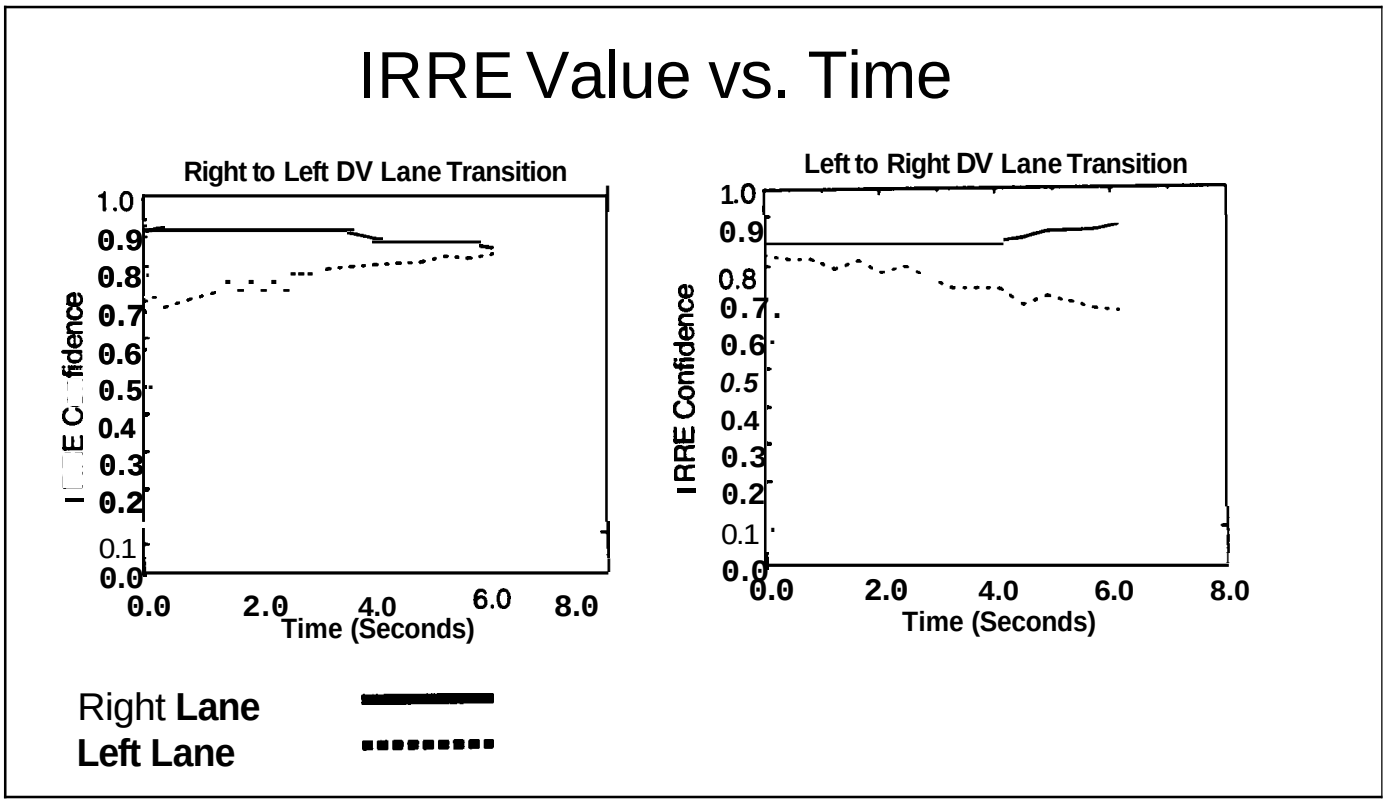

Figure 15. Input Reconstruction Reliability Estimation Values during Dual-Vim Lane Transitions.

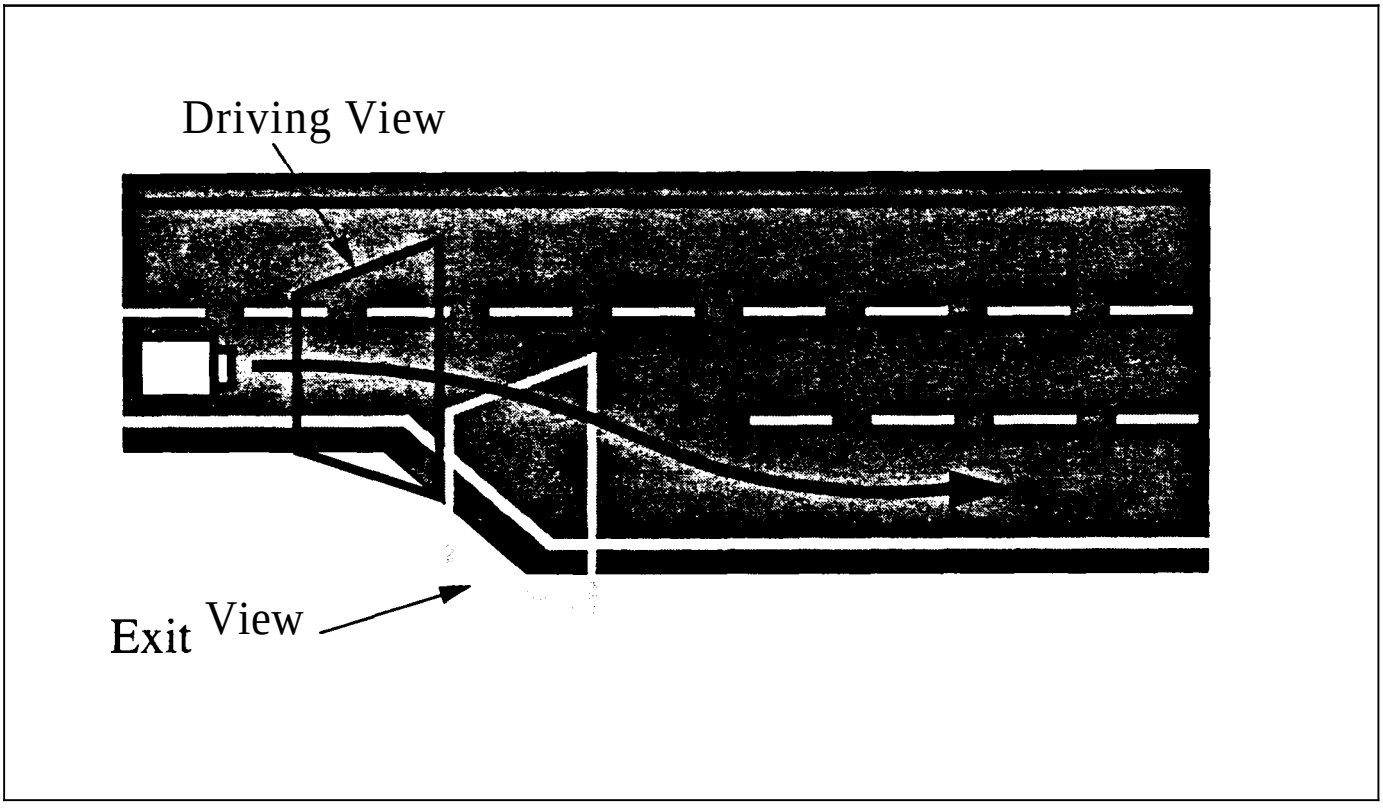

Figure 16. Exit-Ramp Detection Diagram.

lane network learns to key off features, such as the shoulder-to-road boundary, that change significantly when the exit lane appears. To maintain proper control in these situations it is important to detect the exit lane before it is imaged by the driving view. A diagram of the roadway near the exit and virtual camera view locations is depicted in figure 16.

While the system uses the driving view to keep the vehicle in its lane, images are created from the exit view and passed to the associated ALVINN network, which creates an output steering displacement and IRRE confidence measure. By monitoring these values, the system is able to determine when the exit lane begins. The IRRE confidence in the exit-lane network will become high, and its output displacement will match that of the driving view when the exit ramp is present. (The displacements will match because the driving and exit lanes are parallel, and the exit view is not rotated with respect to the road.

After the exit lane has been detected, the driving-lane network is no longer used, and the vehicle is controlled using only the output of the exit-lane network. The exit view is incrementally shifted both laterally and longitudinally toward its standard location in front of the vehicle. The lateral component of each shift results in an image in which it appears that the vehicle is offset to the left of its proper driving position in the exit lane. To recenter the vehicle, the network produces an output indicating that the vehicle should steer to the right, recentering it with respect to the image and moving it further into the exit lane. The network confidence for each new virtual view is required to be above a threshold for two iterations before the view is shifted again. This process continues until the exit view has reached its standard position, and the vehicle has completely transitioned into the exit lane.

Typical road scenes, both before and after an exit lane is present, are shown in figure **17.** In each image, the driving view is outlined in black, and the preprocessed image created from it, along with its associated output-displacement vector and bar graph representation of its IRRE measure, is shown in the lower left corner of each image. The exit view is outlined in white, and the items associated with it are to the right of those for the driving view. In the top image, the IRRE confidence value for the exit view is low, and the output-displacement vector does not match that of the driving view. However, in

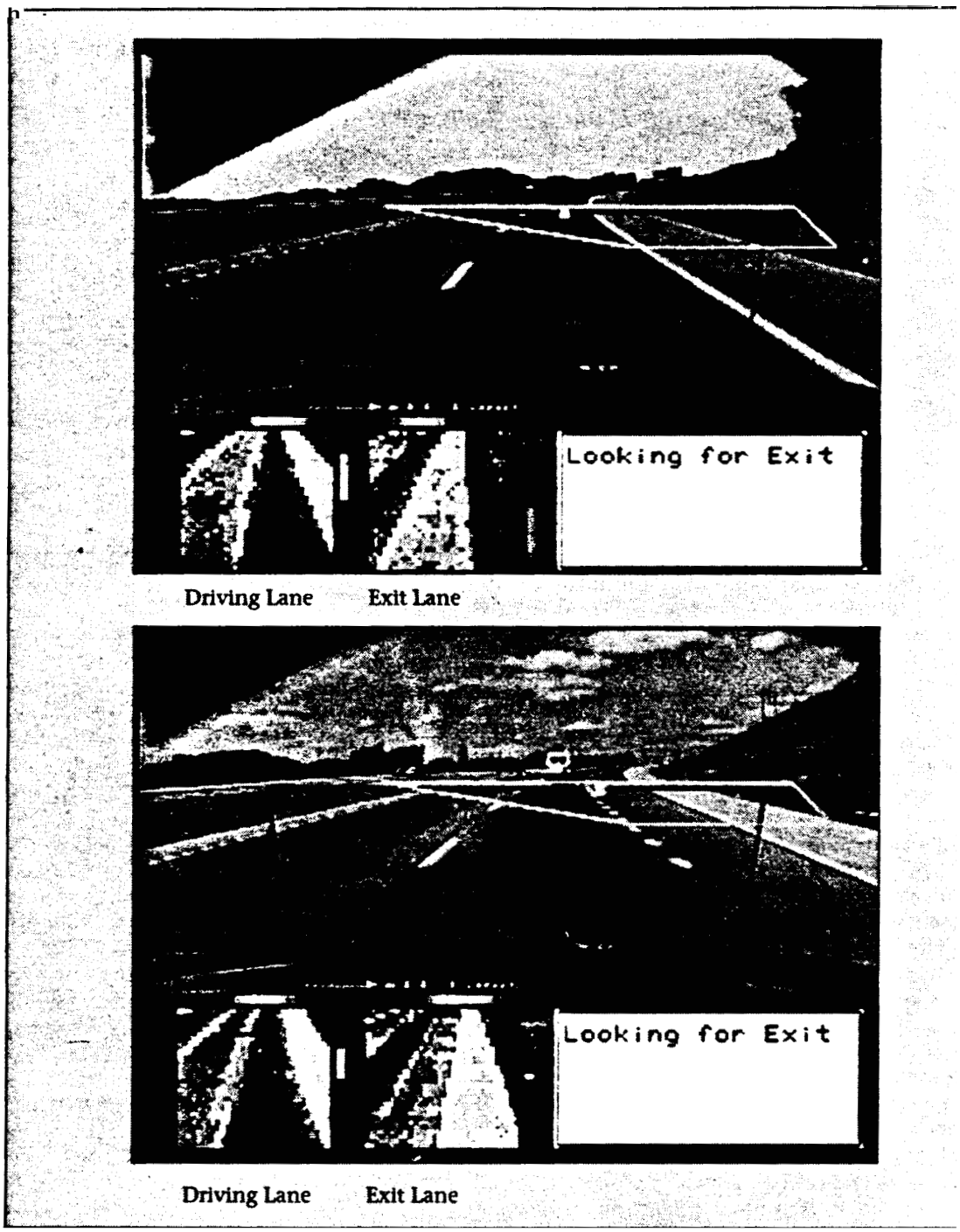

Figure 17. Exit-Ramp Detection Images.

the lower image where the exit lane is present, the IRRE response of both networks is high, and their output displacements are nearly identical.

Of the 20 logged attempts to detect and traverse exit lanes on a rural interstate highway, **19** were successful. The attempts were approximately evenly spread across three different ramps. The same driving- and exit-lane networks were used for every attempt except for the single failure, where a different exit-lane network was used (thus causing the failure). hit-ramp detection occurred at approximately **55** mph, and traversal was done at speeds between 35 and 50 mph.

The system was instructed to begin looking

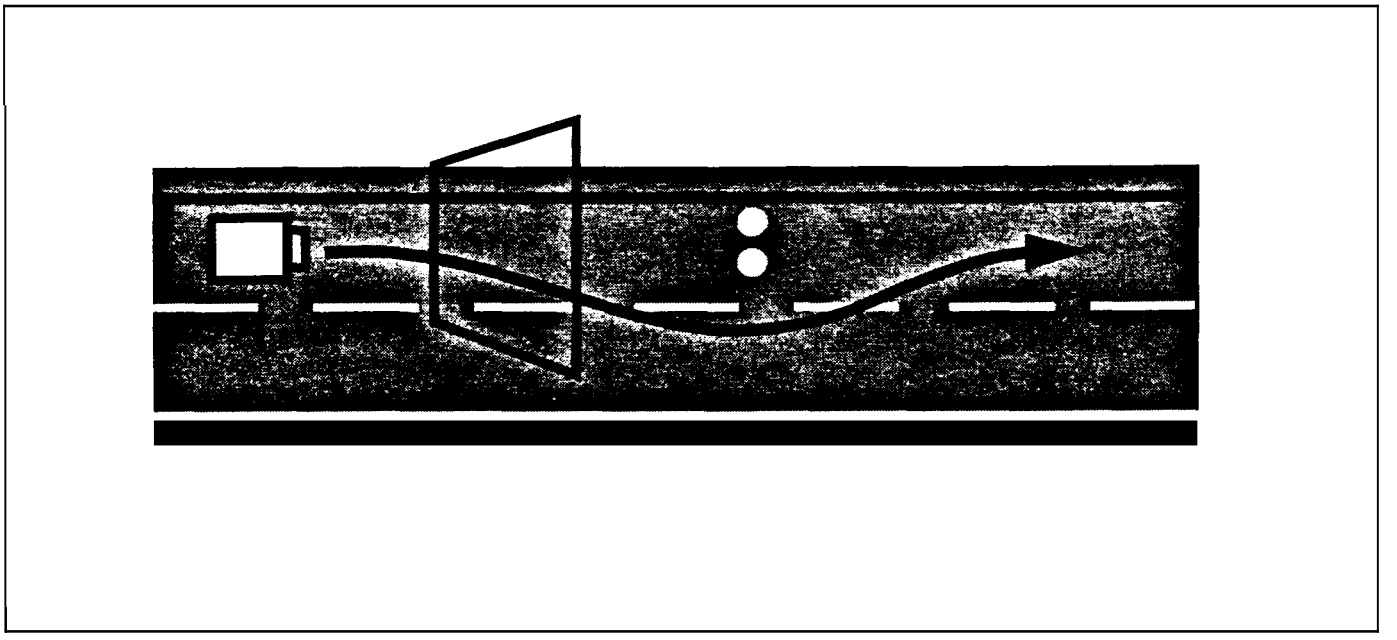

Figure 18. Vehicle Movement While Avoiding an Obstacle.

for the exit lane between 100 and 800 meters before it actually occurred and begin transitioning onto it as soon as it was found. Detection was robust and consistent, occurring immediately after the lane appeared centered in the exit view. Finally, there were no false positives, where the system incorrectly believed that the exit lane was present.

The system performance differed the most in the traversal part of the task because of the exit-lane network and the view that was used. This view was positioned so that the network would learn the most visible feature in the scene—the shoulder-to-off road lane boundary. (The white lines on the concrete roadway were not a large enough feature, and the exit lane was too lightly traveled for noticeable oil spots to be created.) Choosing this view meant that when the shoulder width changed, navigation performance would suffer. In practice, this led to consistent overshoot during navigation of the particular exit, which had a wider shoulder than that on which the network trained.

Obstacle-Avoidance Maneuvers

Although most systems are designed to keep the vehicle in the center of the driving lane, there are times when this behavior is not desirable. Obstacle avoidance is one such scenario. At least three reasons, besides avoiding a collision, can be given why moving the vehicle from the center of the driving lane when an obstacle is detected is an important capability of a lane-keeping system.

First, the current state of automotive obstacle-detection technology precludes the detection of typical highway obstacles at distances significantly greater than the minimum stopping distance for vehicles traveling at the legal highway speed. If the systems do not detect the obstacle immediately and begin braking, the vehicle will not be able to avoid hitting the obstacle by stopping.

Second, many obstacles will enter the driving lane at a distance closer than the minimum stopping distance of the vehicle. In this case, even immediate detection will not allow the vehicle to avoid a collision by stopping. The only way to miss the obstacle is to swerve from the current driving lane.

The final reason relates not to the capabilities of the vehicle equipped with the obstacle-detection system but rather to following vehicles that are not equipped with it. In these situations, even if the detection system finds an upcoming obstacle in time to slow the vehicle to a stop, it might not be the safest maneuver because following vehicles might not have similar stopping ability. Thus, to avoid being rear ended by following vehicles, it might be necessary to swerve from the driving lane to avoid hitting an obstacle.

A separate swerve system is not necessary to implement an obstacle-avoidance maneuver. Active control of virtual camera views, similar to the techniques used for lane changing, can be used with ALVINN to accomplish

this maneuver.

To execute this maneuver, the lane-changing control algorithm was modified into what is called the ***swerve and offset driving algorithm*** (SODA). Instead of moving the vehicle completely into the destination lane, SODA only offsets the vehicle from the center of the driving lane by a prespecified amount and then returns it to the normal driving position. This maneuver effectively allows the vehicle to swerve to miss an obstacle in the driving lane. This maneuver is shown in figure **18.**

Key points of the SODA maneuver are as follows: First, **IRRE** confidence and road-model constraints are not enforced so that the maneuver takes place as quickly as possible. Although there is some lag between the virtual camera view location and the expected vehicle location because of the decoupling from the lack of constraints, the vehicle executes a smooth, stable maneuver. Second, after the vehicle has reached the apex of the maneuver, it does not begin to return to the destination lane immediately but, rather, drives with the specified offset for a predetermined distance. This ***hold time*** allows for some margin of error with respect to the actual location of the obstacle.

Approximately 10 left-to-right and **10** right-to-left swerves were logged. All were successfully completed. The obstacle was detected at distances between 40 and 60 meters by the test driver. After the test driver indicated an obstacle was present, the system moved the vehicle to an offset of **2.25** meters from the driving-lane position using **8** virtual camera views. The system held the vehicle at this position for **0.5** seconds and then returned it to its normal position in the driving lane.

Intersection Navigation

Another step in the evolution of autonomous driving systems is the intelligent handling of road junctions and intersections (Muller and Baten **1995;** Kluge and Thorpe **1993;** Pomerleau **1993;** Rossle, Kruger, and Gengenbach **1993;** Struck et al. **1993;** Siegle et al. **1992;** Ulmer **1992;** Crisman **1990;** Kushner and Puri **1987).** The techniques presented in this section are based on a data-driven, active philosophy of vision-based intersection detection and traversal Uochem **1996).** This section describes the application of virtual active vision tools to this area and presents the algorithms that make autonomous detection and traversal of intersections possible. The capability is based on geometrically modeling the world: This model is utilized to accurately image interesting and relevant parts of the intersection using virtual cameras and active camera control and by monitoring ALVINN's response to the created images.

ALVINN, enhanced with virtual active vision tools, can be used to detect and traverse road junctions and intersections in two different scenarios: In the first scenario, the system only has knowledge that an intersection is present in front of the vehicle. The system does not know the orientation of road branches that are intersecting at this road junction. In this scenario, the goal is to locate each intersection branch.

In the second scenario, ALVINN has a priori knowledge about an upcoming intersection. The information does not specify where the intersection is actually located, only that it is approaching. With this information, appropriate virtual camera views can be created, and correct networks can be associated with each. The location and orientation of each virtual camera, and the type of network used with each, is dependent on the type of road that is expected to be encountered. When the road or intersection to be detected is present, the virtual cameras will image it in a way that is meaningful to ALVINN's neural network. By continually monitoring the network's confidence for each virtual camera image, the system can determine when the intersection is present.

Two sets of experiments were conducted to assess the usefulness of virtual cameras for autonomously detecting roads and intersections. The goal of the first set of experiments, which were performed on the Navlab **2,** was to test the basic ability of virtual cameras to create images that were usable by ALVINN for intersection branch detection. The second set of experiments used the Navlab 5 vehicle, which was equipped with a roof-mounted pan-tilt platform instead of a fixed camera. In this set of experiments, the goal was to use active camera control to enhance the performance of the detection and traversal algorithms.

Experiment 1: Intersection Branch Detection

For this experimental set, the vehicle was positioned approximately **35** meters off the road that was to be detected and aligned perpendicularly to it. **A** virtual view rotated **90** degrees to the right of straight ahead was created. This view was placed 20 meters in front of the vehicle (figure **19).** The vehicle was instructed to move along its current heading until the system detected the road.

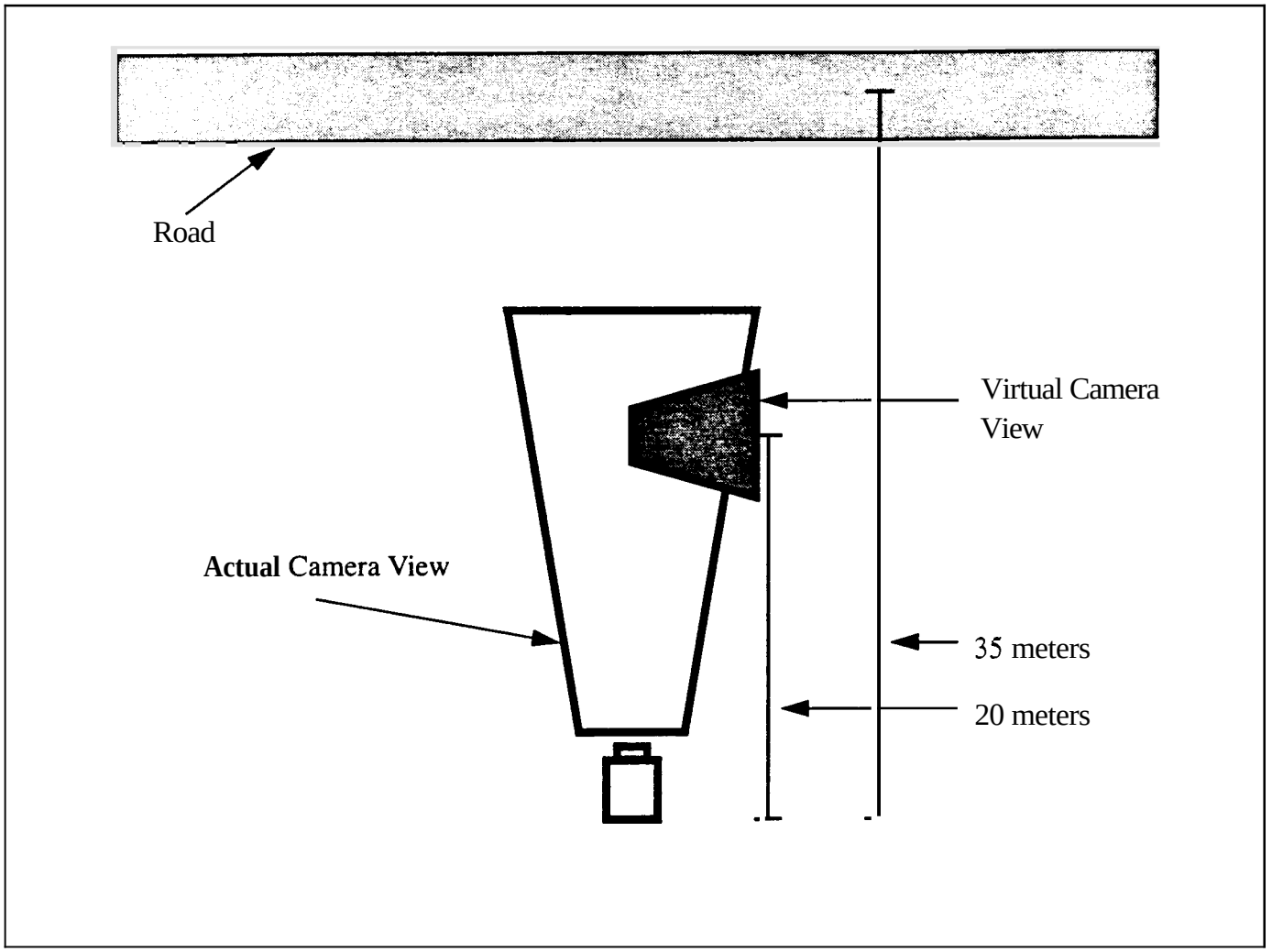

Figure 19. Road-Detection Scenario.

To accomplish detection, every 0.3 seconds as the vehicle approached the road at a speed of about 5 mph, a virtual image was created and passed to the system's neural network. The network produced an output-displacement vector and an IRRE confidence value. To determine when the system had actually located the road, the IRRE metric was monitored. When this metric increased above a user-defined threshold value, which was typically 0.8 out of 1.0, ALVINN reported that it had located the road.

Application of IRRE to Road Detection

Because the IRRE metric is the key to the intersection branch-detection process, emphasis was placed on evaluating the metric in typical branch-detection scenarios. Using the IRRE metric to indicate when roads are present in the input virtual image assumes moved so that it imaged areas between the vehicle and the road, on the road, and beyond the road. Specifically, actual images were taken when the vehicle was at distances of 25, 20, 15, and 10 meters from the center

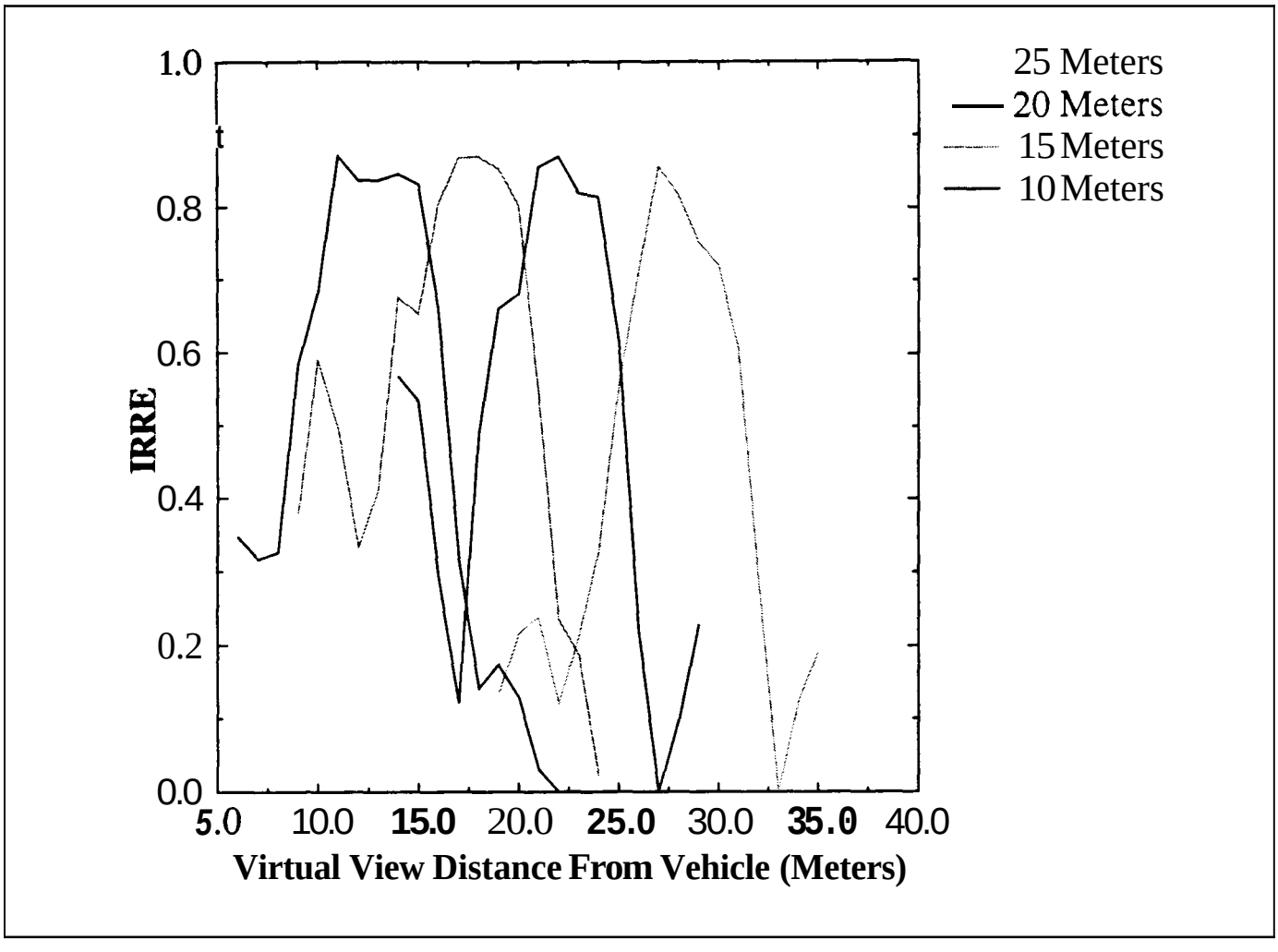

Figure 20. Input Reconstruction Reliability Estimation Response to Different Camera Images.

of the road. Virtual camera images were created at one-meter intervals on either side of the expected road location. For example, using the actual image taken 20 meters from the road center, virtual views were created every meter between the distances of 14 meters and 29 meters.

For each of the actual images, virtual camera images were created at the interval specified earlier and shown to a network previously trained to drive on the road. The output road location and the IRRE confidence metric were computed. The results of this experiment are shown in figure 20. This figure shows the IRRE response as a function of the virtual camera distance in front of the vehicle for several actual images taken at different distances from the road. (Data from the different actual images are represented by different curves in the graph.) For each actual image, the network's IRRE response clearly peaks near the expected road distance. As the virtual view moves closer to the road, the IRRE response increases, peaking when the virtual view is directly over the road. Response quickly falls again after the view passes over the road. The peaks in all the curves have IRRE values greater than 0.80. For comparison, when the system is driving on a familiar road, the IRRE response is typically between 0.65 and 0.95. The peaks in each IRRE curve actually occur about two meters past the actual road center. This error results from three causes: (1) a violation of the flat-world assumption, (2) errors in camera calibration, and (3) improper initial alignment to the road.

This graph shows that both assumptions stated previously are basically correct: The IRRE response is low when the network is not being presented road images, and the IRRE response is high when the network is being presented accurately imaged virtual views.

The relationship between the input virtual

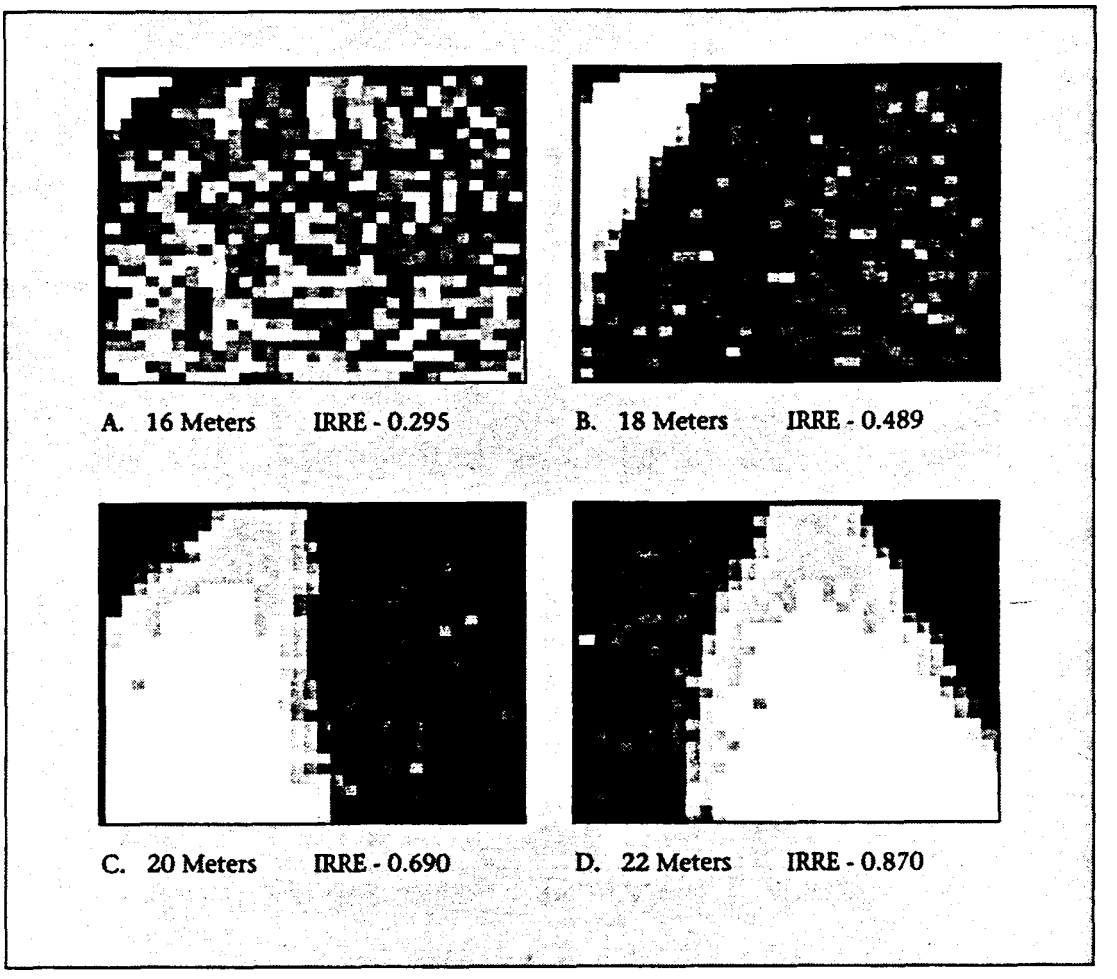

Figure 21. Input Reconstruction Reliability Estimation Values for Typical Detection Images.

image and the IRRE value associated with the image can be better seen in figure 21. It shows virtual images created at different distances in front of the vehicle along with the IRRE response they solicit. In figure 21a, the road is barely visible in the top left corner, and as expected, the IRRE response is very low. As the virtual view is moved forward, it begins to image more of the road, as shown in figure 21b. The IRRE value increases correspondingly. The trend continues until the virtual view is centered over the road, as shown in figure 21d. At this location, the IRRE value is at its peak.

Experiment 2: Intersection-Traversal Experiments on the Navlab 5

After the intersection branches have been detected, traversal can begin. A robust traversal algorithm must overcome two potential problems: (1) a fixed-camera location and (2) violations of the assumptions about the geometry of the road branch. The fixed-camera problem, which limits the effective field of view of the system, was overcome by simply placing the camera on a pan-tilt mount located on the roof of the vehicle. The geometric constraint-violation problem, which caused poor traversal performance in prior experiments, was resolved by using image-derived information to continually update the estimate of the branch location and orientation. Adding these capabilities allowed the system to reliably detect and navigate two test intersections. The first was a Y intersection in a park near campus, and the other was a T junction from a driveway onto a rural road outside Pittsburgh. Although not exhaustive, these two locations are typical of intersection geometries encountered in everyday driving.

Detection with Known Geometry and Unknown Location

In this experiment, the goal was to move along a single-lane road, search for and detect a branch of Y intersection, and drive onto it.

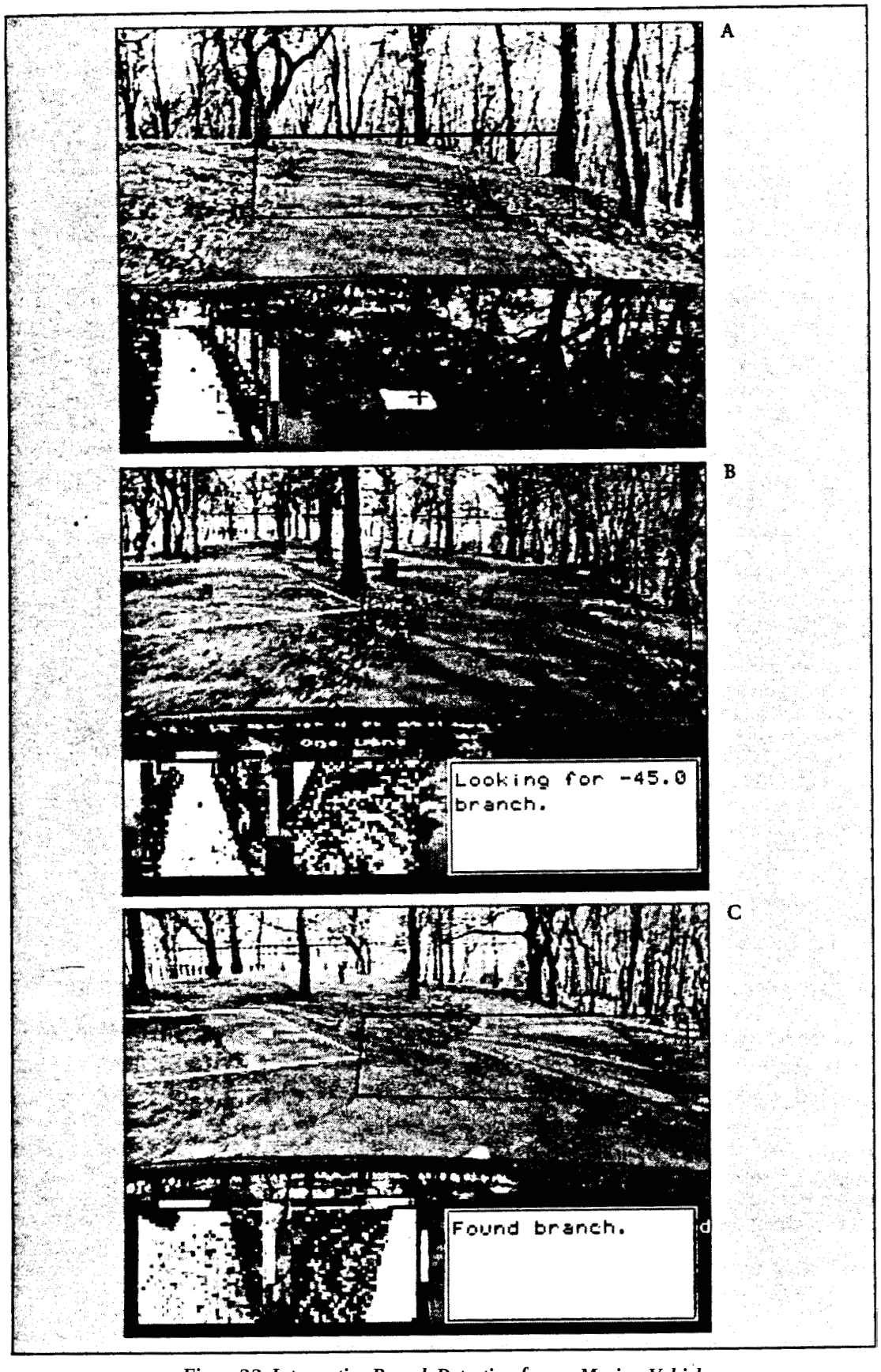

Figure 22. Intersection Branch Detection from a Moving Vehicle.

RALPH

Although ALVINN was very successful, it was not without drawbacks. The biggest was its relatively long training time because of the quick changes in road appearance. A better system would be able to adapt to different road appearances very quickly and do so before the vehicle actually reached the new road type. To resolve this problem, Pomerleau developed RALPH (rapidly adapting lateral position handler) in early 1995. Like ALVINN, RALPH is a vision-based adaptive system that can learn the current road features. Instead of using a neural network to learn, it uses a weak model of road geometry and image reprojection to extract and adapt to the relevant features for driving.

To locate the road ahead, RALPH first resamples a trapezoid-shaped area in the video image, much like a bird's-eye virtual camera, to eliminate the effect of perspective. RALPH then uses a template-based matching technique to find parallel image features in this perspective-free image. These features can be as distinct as lane markings or as subtle as the diffuse oil spots from previous vehicles down the center of the lane. RALPH rapidly adapts to varying road appearance and changing environmental conditions by altering the features it uses to find the road. This rapid adaptation is accomplished in under one second without human intervention.

Because RALPH can exploit any visible features running parallel to the lane, instead of relying exclusively on the presence of distinct lane markings, it can operate in a wider variety of situations than previous road-following systems. In one experiment, called No Hands across America, RALPH drove the Carnegie Mellon University Navlab 5 test-bed vehicle 98 percent of the 2850-mile journey from Washington, D.C., to San Diego, California (Pomerleau and Jochem 1996). During the trip, RALPH drove at an average speed of 63 rnph in conditions including bright sunlight, dusk, rain, and nighttime. During one stretch in Kansas, RALPH drove continuously for 69 miles without the safety driver touching the steering wheel.

The 2 percent of the trip during which manual intervention was required included a nighttime encounter with a 10-mile stretch of freshly paved highway that had not yet been painted with lane markings. Other difficult situations included driving directly into the setting sun and driving through cities.

Near-term applications for the RALPH system include using it as a warning system to alert drowsy or inattentive drivers when they begin to drift off the road. Every year, this type of accident results in nearly 15,000 deaths on U.S. highways. RALPH might also play a role in the automated highway system, where it could automatically steer to keep the vehicle in its lane.

For more information about No Hands across America and RALPH, see the web page http://www.cs.cmu.edu/-pomerlea/nhaa.html.

To accomplish this set of tasks, the moveable camera was required. This camera could be positioned so that the road, as well as a substantial portion of the anticipated road-branch location, could be imaged. If a fixed camera had been used, the number of branch locations and the amount of actual and virtual camera-view overlap that was possible would have been limited.

Initially in this scenario, ALVINN controlled the vehicle in normal lane-keeping mode. See figure 22a. While driving, the system received a message that an intersection was approaching. Although the exact location of the branch was not known, its orientation with respect to the current road segment was given. Using this information, the system created the appropriate virtual camera view, called the detection *view,* which would properly image the branch when it appeared. For this experiment, the target road branch was oriented approximately 40 degrees left of straight ahead. In addition to being angled 40 degrees, the detection view was typically located 7 meters in front of the vehicle. This distance was selected so that the branch would be detected enough in advance to perform the traversal maneuver but close enough so that violations in the flat-world assumption would not become significant.

After creating the detection view, the system determined if the current pan location of the actual camera was sufficient to image both views completely. If not, which was typically the case, the system automatically panned the camera so that the largest portion of the detection view was in the field of view of the actual camera while it maintained the entire driving view in the actual camera's field of view. See figure 22b. Note that after the actual camera has been panned, it is no longer in the same orientation as when the ALVINN network was trained. However, because the virtual camera is at a fixed location with respect to the vehicle and is independent of the actual camera location, the images it created allowed ALVINN to continue driving reliably.

New images from the detection view were created approximately four times a second and passed to ALVINN's neural network for processing. The intersection branch was considered detected when the IRRE confidence value of the network, in response to a detection-view image, became greater than a predetermined threshold value. See figure 22c. The threshold value was typically set to 0.75. When detection occurred, the system indicated this to the safety driver who

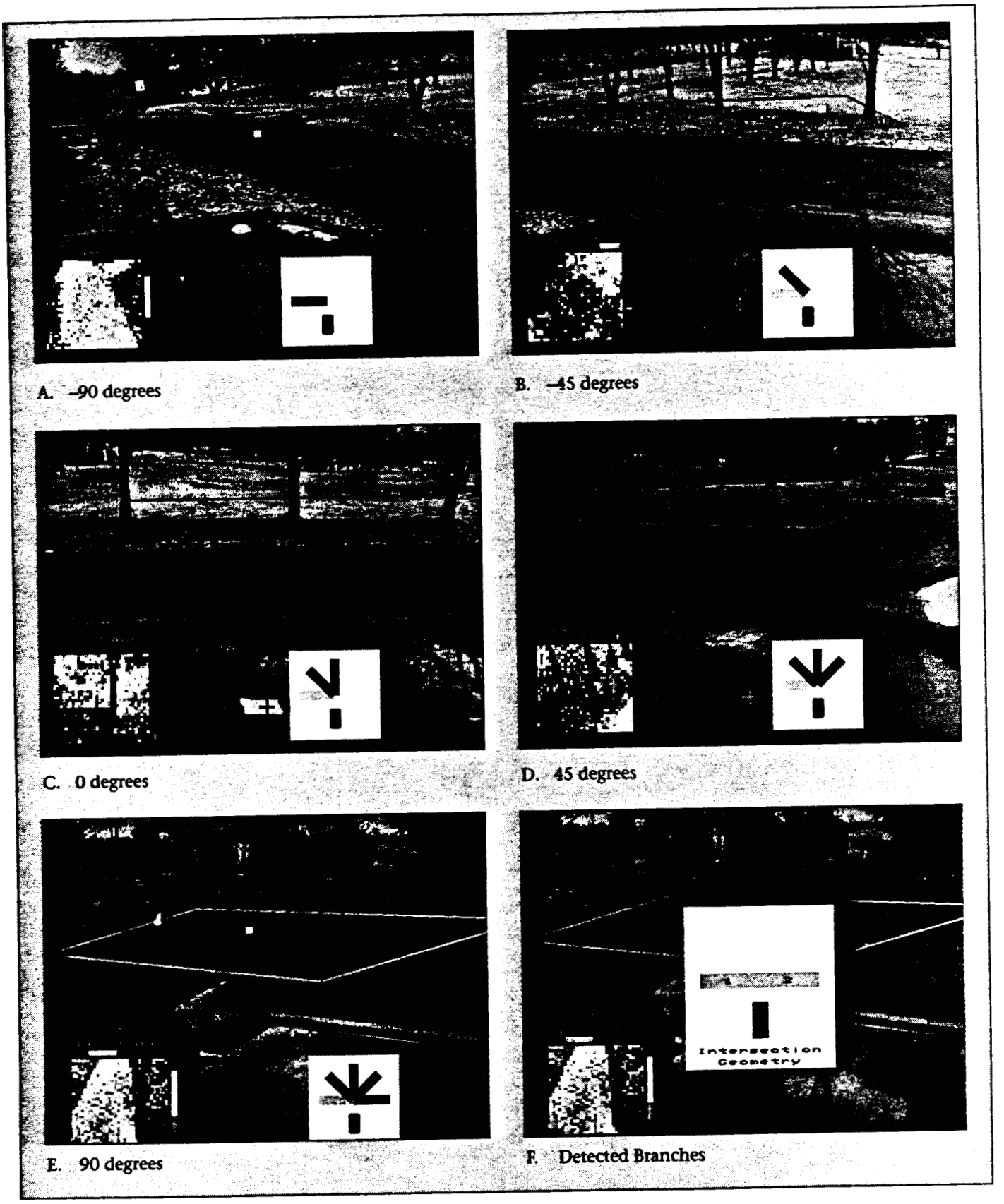

Figure 23. Searching for T-Intersection Branches.

stopped the vehicle. At this point, the system began to localize the intersection branch and navigate through the intersection. This process is described in detail in later sections.

Detection with Unknown Geometry and Known Location

This intersection-detection scenario is the opposite of the previous. In this case, the location of the intersection was known, but the geometry of the intersection was not. Specifically, ALVINN had knowledge about where the center of the intersection was located with respect to the vehicle but did not know the orientation of any of the intersection's branches. The goal was to find each branch and pass its location to a higher-level knowledge source that could request that one of the branches be taken or store the information for later use.

The branch-detection process began with the vehicle located a known distance from the intersection center. The detection algorithm uses a radial search technique to create virtual camera views that image different hypothesis branch locations. Virtual views are created a fixed distance from the intersection center at varying orientations. For this experiment, the angular change between hypothesis views was 45 degrees, and the vehicle's distance from the intersection center ranged between **7** and **10** meters. Images taken at each of these hypothesis branch locations are shown for the T intersection in figure 23. Figure 23f shows which hypothesis intersection branches the system believes are likely to be actual branches, as determined by the simple detection method described in the next paragraph.

As before, the basis for signaling detection is a high IRRE value. If the hypothesis view images an actual road branch, the corresponding IRRE confidence metric will be high, and the orientation of the branch being examined can be saved for further processing. Figure 24 shows enlarged versions of the preprocessed ALVINN image from each of the first five images of figure **23.** Along with the preprocessed image is the IRRE confidence value that ALVINN's network produced when shown the image. For preprocessed images created from virtual cameras that did not image actual road branches, the IRRE value is very low, but for the images that were of actual road branches, the confidence value is significantly higher. From this examination, it is evident that by thresholding based on the IRRE value, hypothesis views that image actual road branches can be discriminated from those that do not.

After the branch orientations had been established, the system waited for the safety driver (or other knowledge source) to indicate which branch should be taken.

Intersection Localization and Traversal Using Active Camera Control

Before traversal takes place, the road branch must be localized to a greater degree of accuracy than was done for detection because of ALVINN's ability to correctly respond to images in which the vehicle appears misaligned with the road. Thus, the exact location and orientation of the road branch with respect to the vehicle is not precisely known. Because lateral translation and orientation errors in virtual-view alignment to the road cannot be determined from a single road image and its associated output, the following two-step branch-localization process must take place.

The first step in localizing the branch further is to use the output displacement of the network to update the position of the virtual camera imaging the road branch. Localization is done by moving the view laterally, perpendicular to the hypothesis branch direction, for a distance equal to the output displacement of the network. After the view has been moved, an image is created from this new location and passed through ALVINN's network, producing another output displacement that is again used to adjust the view. This process is continued until the output displacement of the network changes sign, meaning that the current and last view have "bracketed" the view location that will produce zero output displacement. In this last step, the final displacement from straight ahead is very small.

Figure 2Sa to 25d shows the progression of view locations during this portion of the localization algorithm. The input image and associated output are shown in the lower left portion of the image. Note how the output displacement from straight ahead, shown above the preprocessed image, decreases as the virtual view becomes better aligned with the road branch.

Although the first phase of road-branch localization causes the output displacement of the network to become nearly zero, the orientation of the view with respect to the road branch cannot be assumed correct. Figure 26 illustrates this concept. In this figure, the preprocessed image, along with ALVINN's output displacement created using each image (when the vehicle is in both the left and the right configurations with respect to the road), is

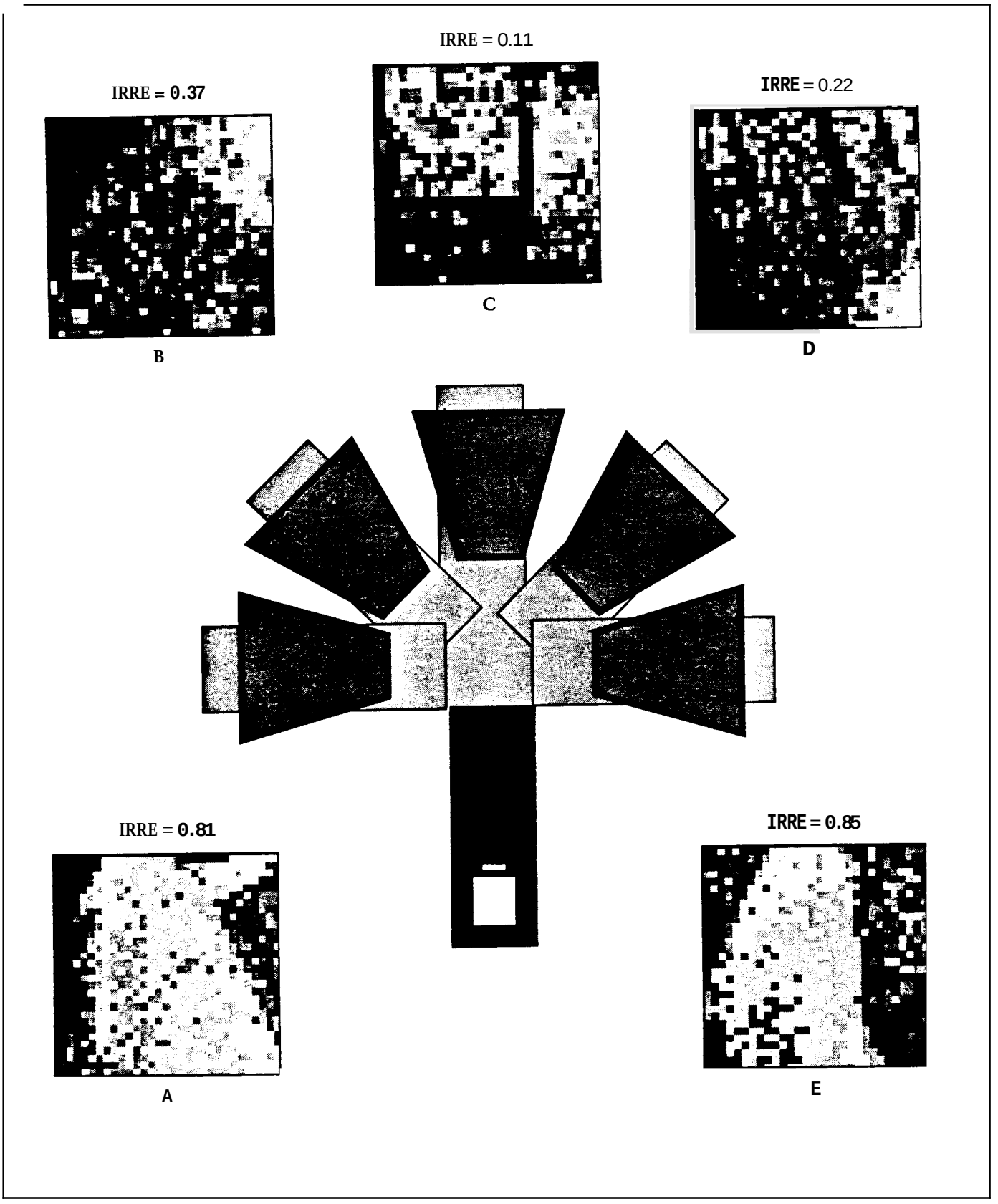

Figure 24. Images and Input Reconstruction Reliability Estimation Values for the T Intersection.

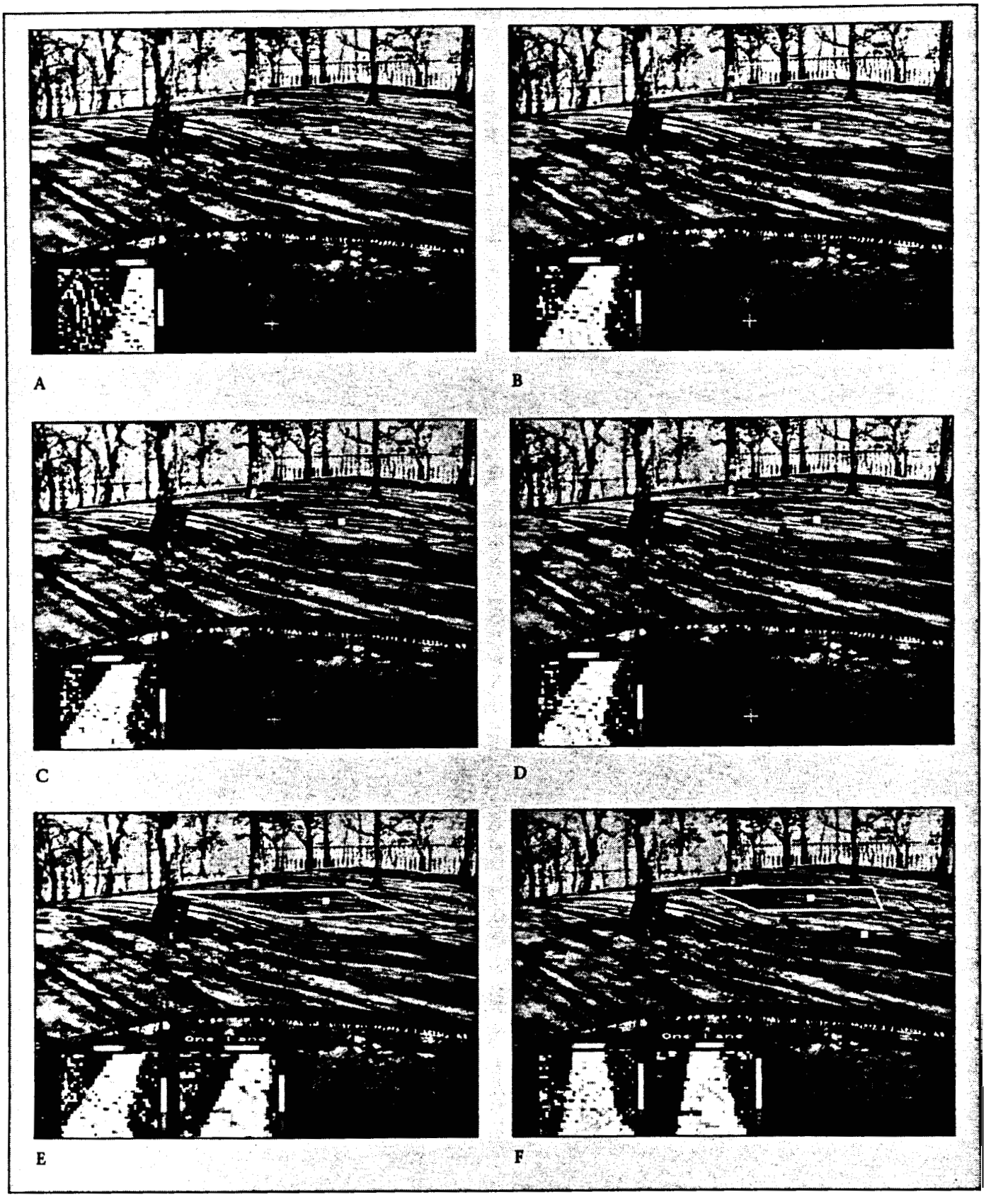

Figure 25. Lateral and Angular Branch Localization.

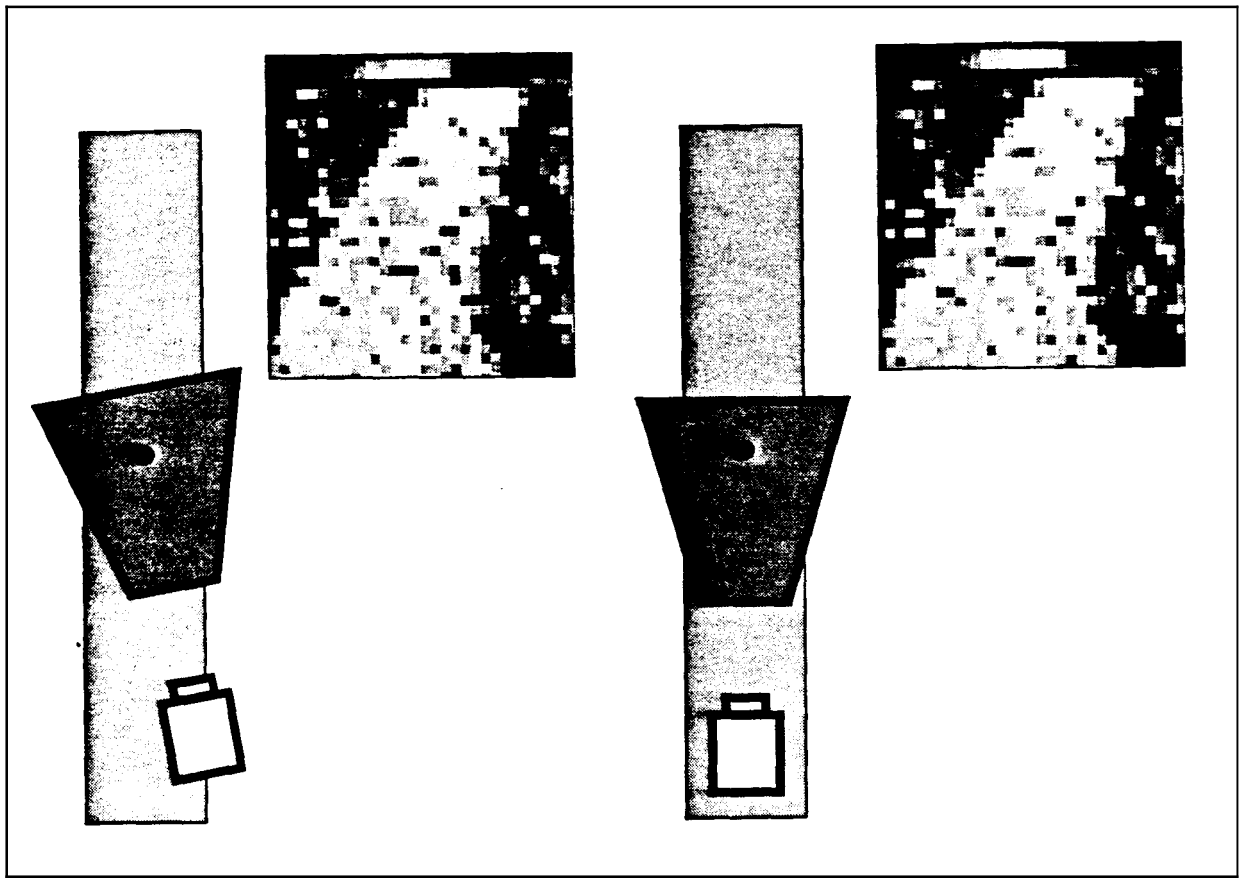

Figure 26. Possible Alignments with Zero Output Displacement.

shown. Note that in each case, the displacement is near zero although the vehicle is only properly aligned with the road in the right example. This discrepancy occurs because ALVINN is trained to produce the displacement required to return the vehicle to the center of the road the look-ahead distance in front of the vehicle. In both cases, the requisite displacement is near zero.

To accurately determine the intersection branch orientation, a second view is required. This view, called the ***projection view,*** is typically created between three and five meters in the direction of the current estimated road-branch orientation. Figure 27 shows the actual road scene (figure 25e) along with a diagram of the original and projection-view arrangement.

If the original view is properly aligned with the intersection branch and accurately reflects the branch's location and orientation, creating an image using the projection view and passing it through ALVINN's network should yield an output displacement close to zero. This is because the previous alignment step reduced the lateral offset of the original view to near zero, in effect centering the original view over the longitudinal axis of the road branch. If the original view is at the correct orientation, projecting 5 meters along the branch orientation should also create a view that is centered over the longitudinal axis of the intersection branch. As shown in figure 27, the projection view is not centered. Although the original view has zero displacement, indicated by the centered Gaussian hump of activation over the preprocessed image, it was not aligned correctly with the intersection branch. Because the original view was misaligned, the projection view is also misaligned, resulting in a nonzero output displacement. In figure **27,** the Gaussian hump indicating the network output displacement created from the projection-view image is shifted right to reflect this misalignment.

The output-displacement difference from zero that the projection-view image produces is a measure of the misalignment in orientation between the original view and the intersection branch. By using the output displacement from the projection view, the

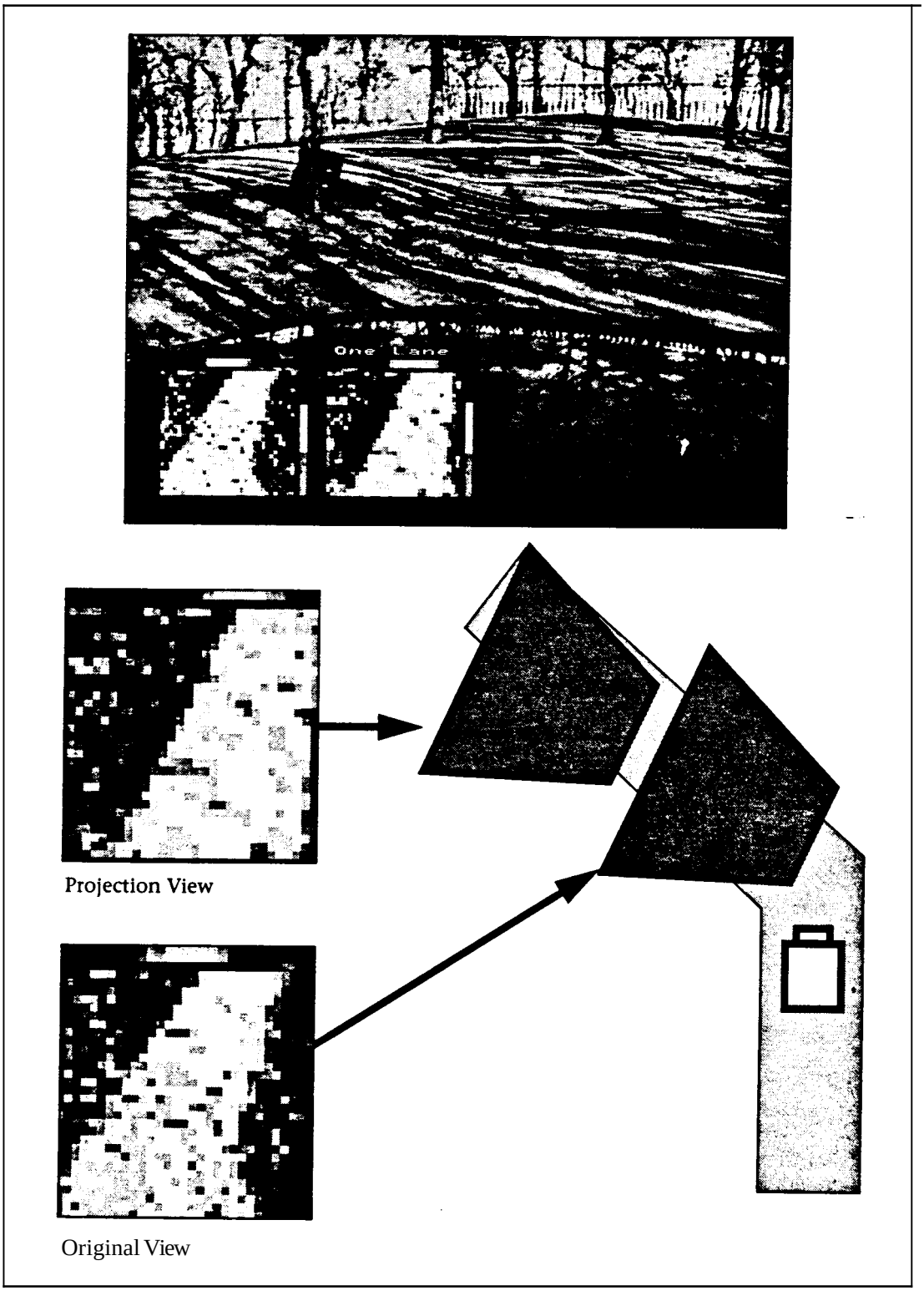

Figure 27. Orientation Localization.

projection distance, and the location of the original and projection views, the amount of this angular misalignment can be computed (figure 28). After rotating the original view to its correct orientation, its lateral position must also be corrected; the rotation correction was made about the original view's location, not about a point on the longitudinal

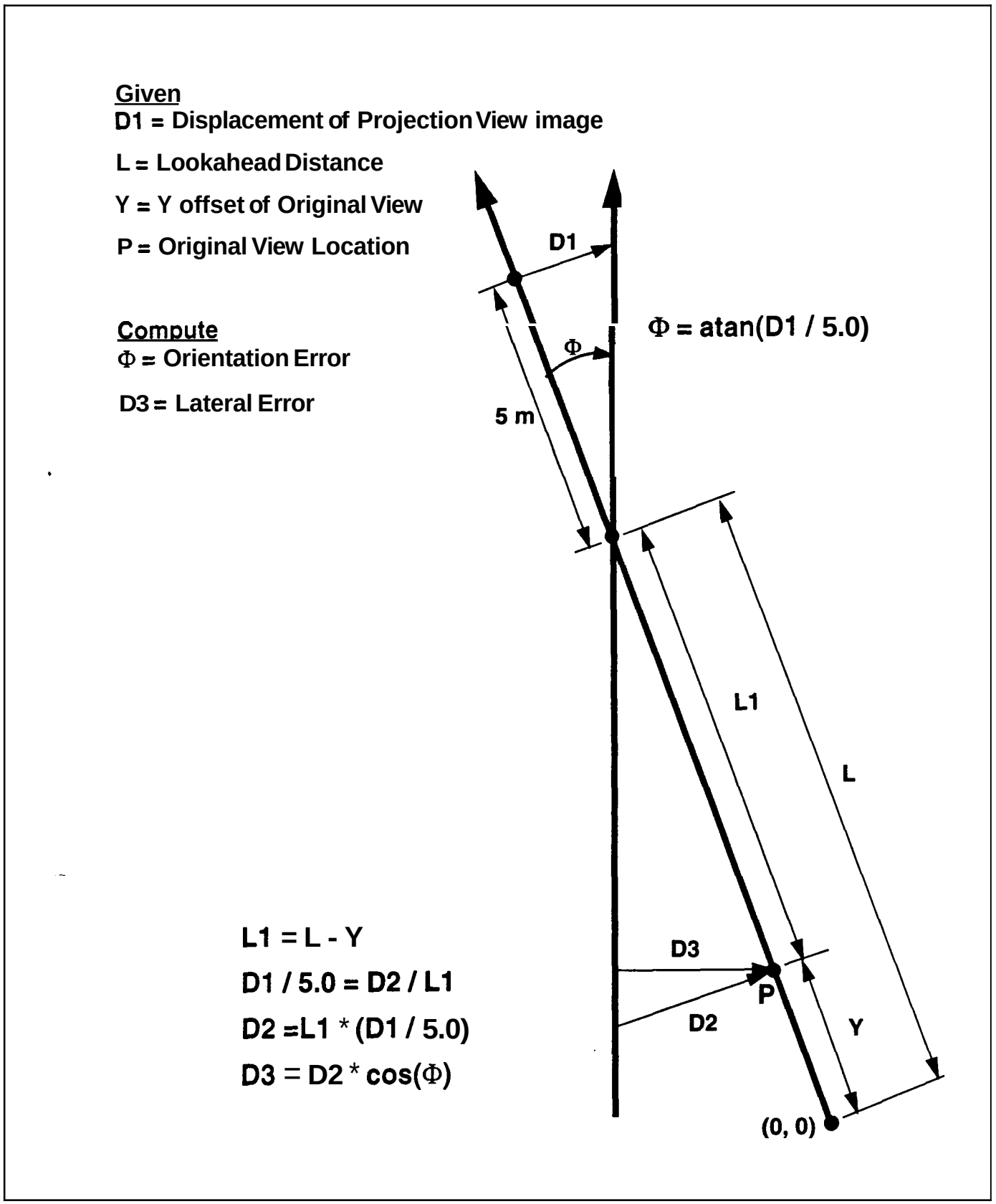

Figure 28. Orientation and Lateral Offset Error Correction.

axis of the intersection branch. From the same information used to compute the orientation error, the lateral offset error, which is a result of the orientation error correction, can also be computed (figure 28). After both error values have been computed, they can be used to update the original view location and orientation so that it more closely matches the intersection branch geometry. Figure 25f shows the original and projection views after the entire localization procedure has been completed. The original view and the projection view are both producing output displacements near zero, indicating that they are aligned with the longitudinal axis of the intersection branch. Once this localization step is finished, traversal of the intersection can begin.

Traversal

Two issues must be considered and resolved for intersection traversal to be successful: (1) tracking the branch as the vehicle moves through the intersection and (2) computing the correct steering arc to execute.

The branch-tracking problem was solved by adapting the branch-localization algorithm presented in the previous section. During traversal, the system continually updated the location and orientation of the original virtual camera view by repeating the alignment procedure presented in the previous section. In addition, when the original view was about to move out of the field of view of the actual camera, the system automatically panned the actual camera appropriately. The ability of the system to correctly orient and localize the intersection branch during traversal is shown for the T intersection in figure 29. Note that a camera pan occurs before figure 29b, 29c, 29e, and 29f.

Creating an acceptable vehicle-control algorithm for navigating intersections was one of the most difficult tasks in this work. The majority of the methods tried caused the vehicle to either severely cut corners, overshoot, or generally become misaligned with the road. A contributing factor to these problems was the lack of accurate geometric information about the intersection branch as the vehicle turned.

As shown in the traversal figures, this problem was alleviated by tracking the intersection branch using a combination of traditional and virtual active vision techniques. However, many of the vehicle-control algorithms still had difficulty matching the vehicle heading to the road Orientation. Based on this observation, the vehicle-control algorithm shown in figure 30 was developed. This algorithm takes into account the branch orientation as determined by the localization algorithm.

The vehicle control algorithm finds tangent points on two lines representing the vehicle's current heading and the intersection branch orientation. The first tangent point, P1, is defined to be the current vehicle position, and the second point, P2, is on the intersection-branch axis. The distance along the branch that P2 is located, measured from the intersection center point, C, is defined to be equal to the distance from P1 to C. After being computed, this distance is held fixed throughout the intersection traversal. C is computed before traversal begins by finding the intersection point of the branch axis and the line representing the vehicle heading. As mentioned earlier, the orientation of the branch axis is not fixed at the hypothesis view orientation but, rather, is the refined branch orientation derived during the branch localization phase.

This construction ensures that a circle can be found that will intersect *P1* and *P2* tangentially. The radius of the circle that intersects these tangent points is the arc that the vehicle uses to drive through the intersection. For each new image, the tangent point, *P2*, and the arc to drive are recalculated based on the new location of the intersection branch so that any errors in vehicle control, positioning, pan angle, or nonlinear branch geometries are considered.

Intersection Navigation Results and Discussion

Using simple thresholding as the discriminating technique, the system was able to successfully detect each intersection branch in 33 of the 35 cases on the Y and T intersections. These trials were distributed about evenly over the moving vehicle Y and the stationary Y and T detection scenarios. In no cases did the system detect a branch that was not present.

Both failure cases were in the T scenario. In one case, the system successfully detected one of the two branches. The other branch's confidence value fell just below the threshold but was still much higher than any of the other three hypothesis locations. In the other failure case, neither branch was detected. For this case, of the two real branches that should have been detected, one did have a noticeably higher IRRE value, but it was still below the detection threshold. The other branch's IRRE value was not significantly different

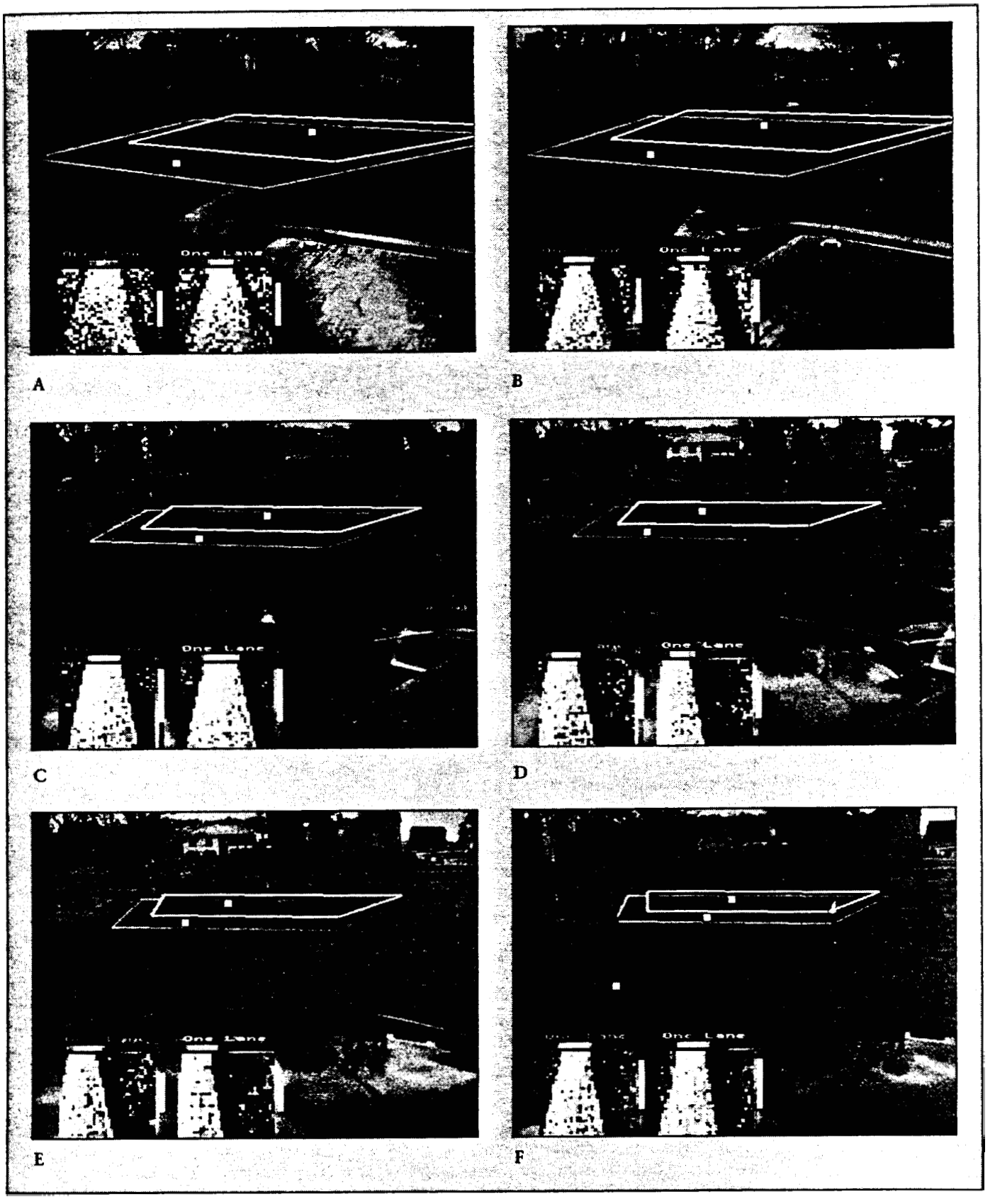

Figure 29. T-Intersection Traversal Images.

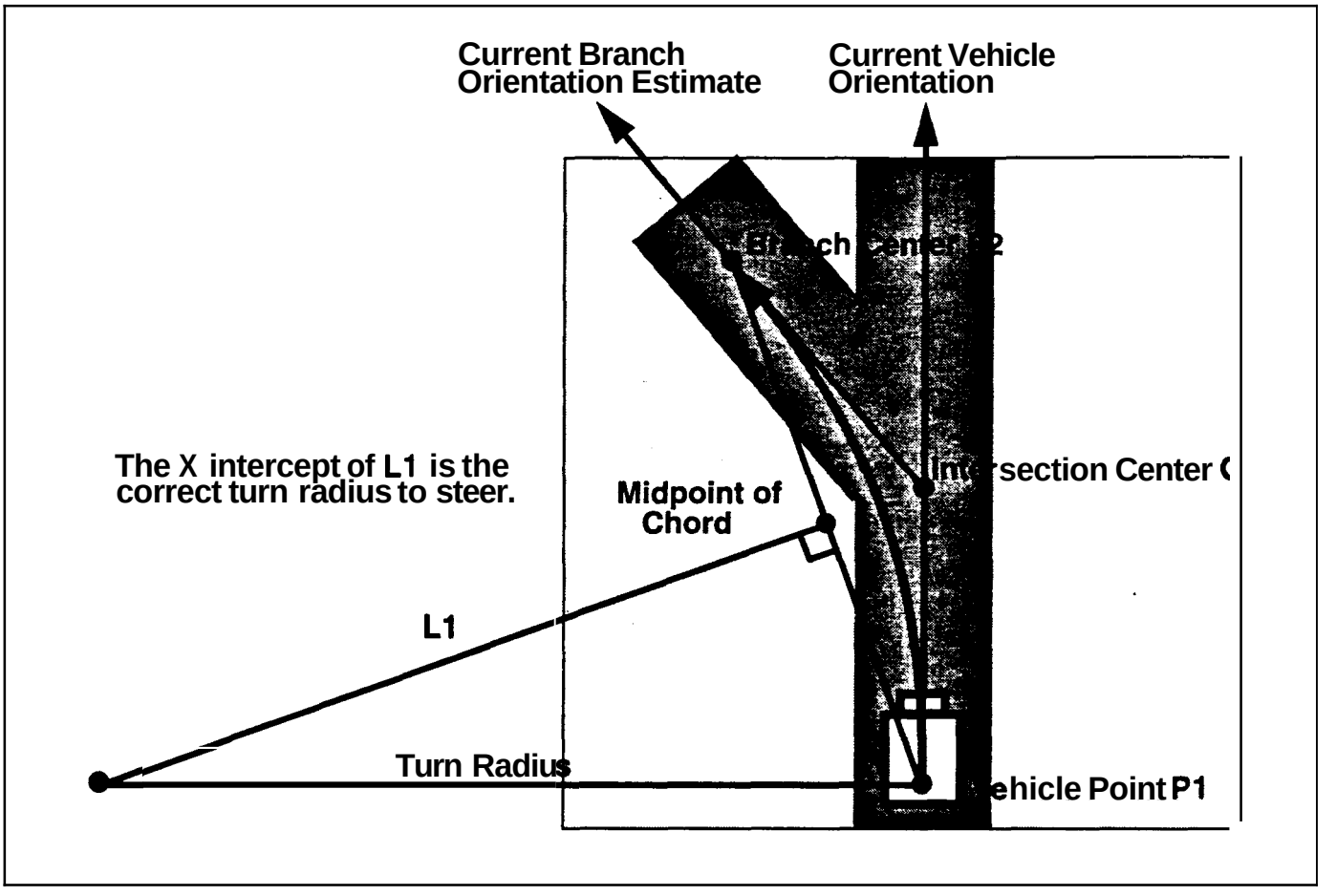

Figure 30. Traversal-Turn Radius Determination.

than any of the other branches. In this case, the problem can be attributed to a change in the ambient lighting, from overcast skies to sunshine, to which the camera was not able to properly adjust. In any case, although detection is robust, it is not foolproof, and redundant branch-verification procedures are necessary.

In all 34 scenarios in which the system detected at least 1 branch, it was able to properly move the vehicle onto the branch and continue driving. The system was able to drive the vehicle onto the left and right branches of the T intersection as well as navigate onto the 45-degree branch of the Y intersection. Although the control algorithm during traversal is simple, having a moving camera and tracking the road branch throughout the maneuver allowed it to work reliably over the experimental domain.

It is reasonable to assume that the detection method will work for any road branch type that ALVINN can learn to drive on. If this assumption is true, this system will have an advantage over other road- and intersection-detection systems that require the researcher to program in new detection methods when new road types are encountered.

Conclusions

The ALVINN system has changed dramatically over the past nine years. From its beginning as a fragile system tied to the resources of a supercomputer, it has evolved into a reliable system capable of driving vehicles at highway speeds for long distances using modest computing resources. Enhancements to ALVINN have produced advances in the areas of neural network training, autonomous on-road navigation, and computer vision. With the recent addition of a simple model of lane geometry, ALVINN now forms the core of a tactical driving system capable of changing lanes and navigating through intersections.

ALVINN'S success can be attributed to a single design principle—build in the well-understood aspects of the autonomous navigation

task, and learn the remainder. Our a priori knowledge of the geometry involved in driving forms the core of ALVINN's built-in functions, freeing the neural network to learn the more difficult aspects of image processing. We continue to search for new methods of incorporating geometric information into autonomous driving systems while maintaining the ability to adapt to the changing situations encountered in the real world. One such method, which we believe might have several advantages over the ALVINN neural network approach, is described in the sidebar on RALPH (Pomerleau and Jochem 1996).

Acknowledgments

This research was sponsored in part by the Defense Advanced Research Projects Agency (DARPA) under contracts DACA76-89-C-0014 and DAAE07-90-C-R059, respectively monitored by the U.S. Army Topographic Engineering Center and Tank-Automotive Command, and in part by a DARPA Research Assistantship in Parallel Processing administered by the Institute for Advanced Computer Studies, University of Maryland. Additional sponsorship was provided by the U.S. Department of Transportation under contracts DTNH22-93-C-07023 and DTFH61-94-X-00001. Finally, the authors would like to thank Delco Electronics for providing the test-bed vehicle on which much of this work was conducted.

References

Behringer, R.; Holt, V.; Dickmanns, D. **1992.** Road and Relative Ego-State Recognition. Presented at the 1992 IEEE Symposium on Intelligent Vehicles, 29 June–1 July, Detroit, Michigan.

Blaauw, G. J. **1982.** Driving Experience and Task Demands in Simulator and Instrumented Car: A Validation Study. *Human Factors* 24:473–486.

Brooks, R. A. **1986.** A Robust Layered Control System fora Mobile Robot. *IEEE Journal of Robotics and Automation* RA-2(1): **14–23.**

Crisman, J. D. **1990.** Color Vision for the Detection of Unstructured Roads and Intersections. Ph.D. diss., Department of Electrical Engineering, Carnegie Mellon University.

Dickmanns, E. D., and Zapp, A. **1987.** Autonomous High-Speed Road Vehicle Guidance by Computer Vision. In Proceedings of the Tenth World Congress on Automatic Control, Volume **4, 221–226.** New York: Pergamon.

Grupen, R.; Connolly, C.; Souccar, K.; and Burleson, W. **1995.** Toward a Path Co-Processor for Automated Vehicle Control. Presented at the **1995** IEEE Symposium on Intelligent Vehicles, **25–26** September, Detroit, Michigan.

Hatipoglu, C.; Ozguner, U.; and Unyelioglu, K. 1995. On Optimal Design of a Lane Change Controller. Presented at the **1995** IEEE Symposium on Intelligent Vehicles, **25–26** September, Detroit, Michigan.

Jochem, T. M. **1996.** Vision-Based Tactical Driving. Ph.D. diss., Technical Report CMU-RI-TR-96-14, The Robotics Institute, Carnegie Mellon University.

Jochem, T. M.; Pomerleau, D. A.; Kumar, B.; and Armstrong, J. **1995.** PANS: A Portable Navigation Platform. Presented at the **1995** IEEE Symposium **on** Intelligent Vehicles, **25–26** September, Detroit, Michigan.

Kluge, K. **1993.** YARF: An Open-Ended Framework for Robot Road Following. Ph.D. diss., School of Computer Science, Carnegie Mellon University.

Kluge, K., and Thorpe C. **1993.** Intersection Detection in the YARF Road-Following System. ***Intelligent Autonomous Systems, Volume3,* 145–155.**

Kushner, T., and Puri, S. **1987.** Progress in Road Intersection Detection for Autonomous Vehicle Navigation. ***The International Society for Optical Engineering, Mobile Robots II*** **852:19–24.**

Muller, N., and Baten, S. **1995.** Image Processing–Based Navigation with an Autonomous Car. ***Intelligent Autonomous Systems*** **4:591–598.**

Nelson, W. **1989.** Continuous-Curvature Paths for Autonomous Vehicles. Presented at the **1989** IEEE International Conference on Robotics and Automation, **14–19** May, Scottsdale, Arizona.

Pomerleau, D. A. **1993.** ***Neural Network Perception for Mobile Robot Guidance.*** Norwell, Mass.: Kluwer.

Pomerleau, D. A., and Jochem, T. M. **1996.** A Rapidly Adapting Machine Vision System for Automated Vehicle Steering. ***IEEE Expert*** **11(2): 19–27.**

Pomerleau, D. A.; Gowdy, J.; and Thorpe, C. E. **1992.** Combining Artificial Neural Networks and Symbolic Processing for Autonomous Robot Guidance. Presented at the **1992** IEEE Symposium on Intelligent Vehicles, **29** June–1 July, Detroit, Michigan.

Reid L. D.; Solowka E. N.; and Billing A. M. **1991.** A Systematic Study of Driver Steering Behaviour. ***Ergonomics*** 24:31–38.

Rosenblum, M. **1994.** The Use of Neural Networks for Machine Vision Applications. Master's thesis, Department of Computer Science, University of Maryland.

Rossle, S.; Kruger, V.; and Gengenbach, V. **1993.** Real-Time Vision-Based Intersection Detection for a Driver's Warning Assistant. Presented at the **1993** Symposium on Intelligent vehicles, **14–16,** Tokyo, Japan.

Rumelhart, D. E.; Hinton G. E.; and Williams R. J. **1986.** Learning Internal Representations by Error Propagation. In ***Parallel Distributed Processing: Explorations in the Microstructures of Cognition, Volume 1,*** **318–362.** Cambridge Mass.: MIT Press.

Siegle, G.; Geisler, J.; Laubenstein, F.; Hagel, H.; and Struck, G. **1992.** Autonomous Driving on a Road Network. Presented at the **1992** IEEE Symposium on Intelligent Vehicles, **29** June–July **1,** Detroit, Michigan.

Struck, G.; Geisler, F; Laubenstein, H.; Nagel, H.;

and Siegle, G. **1993.** Interaction between Digital Road Map System and Trinocular Autonomous Driving. Presented at the **1993** IEEE Symposium on Intelligent Vehicles, **14–16** July, Tokyo, Japan.

Turk, M.; Morgenthaler, D.; Gremban, K.; and Marra, M. **1988.** VITS: A Vision System for Autonomous Land Vehicle Navigation. IEEE *Transactions in Pattern Analysis and Machine Intelligence* 10(3): **342–361.**

Ulmer, B. **1992.** VITA—An Autonomous Road Vehicle (ARV) for Collision Avoidance in Traffic. Presented at the **1992** IEEE Symposium on Intelligent Vehicles, **29** June–1 July, Detroit, Michigan.

Wallace, R.; Stentz, A.; Thorpe, C.; Moravec, H.; Whittaker, W.; and Kanade, T. **1985.** First Results in Robot Road-Following. In Proceedings of the Ninth International Joint Conference on Artificial Intelligence, **1089–1095.** Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Todd Jochem is a postdoctoral research associate at Carnegie Mellon University's (CMU) Robotics Institute. He received his Ph.D. from CMU in January **1996.** His dissertation concentrated on using vision-based focus-of-attention techniques to allow an autonomous vehicle to execute driving tasks such as lane changing and intersection navigation. Jochem is also part of CMU's Automated Highway System (AHS) Team and is leading CMU's development effort for the **1997** AHS Technical Feasibility Demonstration. This work includes developing hardware and software systems that demonstrate the effectiveness and applicability of vehicle-based technologies to improve the safety and efficiency of highway travel.

Dean Pomerleau is a research scientist at Carnegie Mellon University (CMU), with a joint appointment in the Robotics Institute and the School of Computer Science. He received his Ph.D. from CMU in **1992.** His dissertation and current research focus on perception and learning for intelligent vehicles. Pomerleau is the director of the Single-Vehicle Roadway Departure Prevention Project, a U.S. Department of Transportation (USDOT)–sponsored partnership of universities and companies developing and testing warning systems to prevent highway accidents. He is also a principle investigator in the Automated Highway System (AHS) Program, a consortium of companies, universities, and the USDOT whose goal is to develop driving systems using a combination of smart-vehicle and smart-highway technology. The aim of the AHS program is to improve the safety, efficiency, and convenience of highway travel.