Pomerleau, D.A. (in press) Knowledge-based Training of Artificial Neural Networks for Autonomous Robot Driving. To appear in Robot Learning, J. Connell and S. Mahadevan (eds.), Kluwer Academic Publishing.

Knowledge-based Training of Artificial Neural Networks for Autonomous Robot Driving

Dean A. Pomerleau
Carnegie Mellon University
School of Computer Science
Pittsburgh, PA 15213-3890

Abstract

Many real world problems quire a degree of flexibility that is difficult to achieve using hand programmed algorithms. One such domain is vision-based autonomous driving. In this task, the dual challenges of a constantly changing environment coupled with a real time processing constrain make the flexibility and efficiency of a machine learning system essential. This chapter describes just such a learning system, called ALVINN (Autonomous Land Vehicle In a Neural Network). It presents the neural network architecture and training techniques that allow ALVINN to drive in a variety of circumstances including singlelane paved and unpaved roads, multilane lined and unlined roads, and obstacle-ridden on- and off-road environments, at speeds of up to 55 miles per hour.

1 Introduction

Autonomous navigation is a difficult problem for traditional vision and robotic techniques, primarily because of the noise and variability associated with real world scenes. Autonomous navigation systems based on traditional image processing and pattern recognition techniques often perform well under certain conditions but have problems with others. Part of the difficulty stems from the fact that the processing performed by these systems remains fixed across various environments.

Artificial neural networks have displayed promising performance and flexibility in other domains characterized by high degrees of noise and vari-

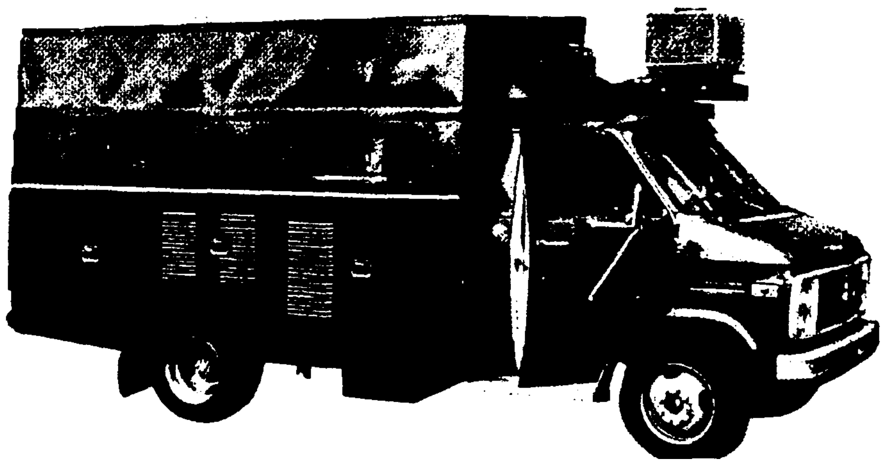

Figure 1: The CMU Navlab Autonomous Navigation Testbed

ability, such as handwritten character recognition [6] and speech recognition [15] and face recognition [2]. ALVINN (Autonomous Land Vehicle In a Neural Network) is a system that brings the flexibility of connectionist learning techniques to the task of autonomous robot navigation. Specifically, ALVINN is an artificial neural network designed to control the Navlab, the Carnegie Mellon autonomous driving test vehicle (See Figure 1).

This chapter 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 chapter presents training techniques that allow ALVINN to learn in under 5 minutes to autonomously control the Navlab by watching a human driver's response to new situations. Using 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 of up to 55 miles per hour.

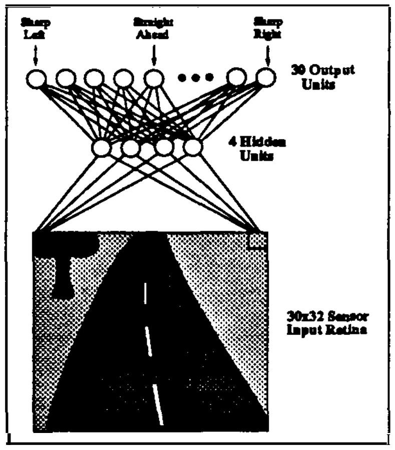

Figure 2: Neural network architecture for autonomous driving.

2 Network Architecture

The basic network architecture employed in the ALVINN system is a single hidden layer feedforward neural network · See Figure 2 'he input layer now consists of a single 30x32 unit "retina" onto which a :or image from either a video camera or a scanning laser rangefinder is projected. Each of the 960 input units is fully connected to the hidden layer of 4 units, which is in turn fully connected to the output layer. The 30 unit output layer is a linear representation of the currently appropriate steering direction which may serve to keep the vehicle on the mad or to prevent it from colliding with nearby obstacles'. The centexmost output unit represents the "travel straight ahead" condition, while units to the left and right of center represent successively sharper left and right turns. The units on the extreme left and right of the output vector represent turns with a 20m radius to the left and right respectively, and the units in between represent turns which 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 30x32 pixels and projected onto the input layer. After propagating activa-

'The task a particular driving network perfoms depends on the type of input sensor image and the driving situation it has been trained to handle.

tion 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 mass of activation instead of the most active output unit when determining the direction to steer permits finer steering corrections, thus improving ALVINN's driving accuracy.

3 Network Training

The network is trained to produce the c o m t steering direction using the backpropagation learning algorithm [13]. In backpropagation, 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.

Autonomous driving has the potential to be an ideal domain for a supervised learning algorithm like backpropagation since there is a readily available teaching signal or "correct response" in the form of the human driver's current steering direction. In theory it should be possible to teach a network to imitate a person as they drive 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 3.

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.

3.1 Potential Problems

There are two potential problems associated with training a network using live sensor images as a person drives. First, since 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 may occasionally stray from

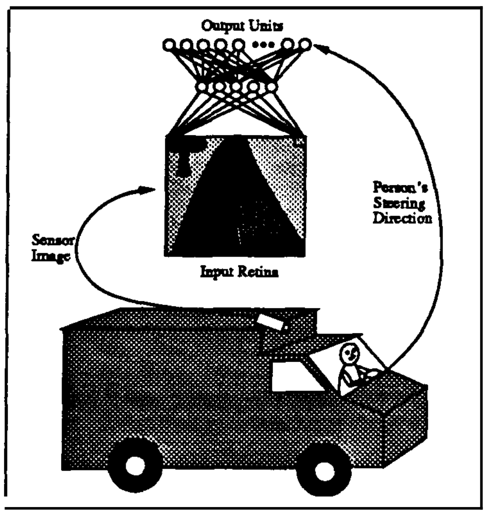

Figure 3: Schematic representation of training "on-the-fly". The network is shown images from the onboard sensor and trained to steer in the same direction as the human driver.

the mad center, so it must be prepared to recover by steering the vehicle back to the middle of the road. The second problem is that naively training the network with only the current video image and steering direction may cause it to overlearn recent inputs. 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.

Both problems associated with training on-the-fly stem from the fact that back-propagation requires training data which is 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 tuming learning off while the driver steers the vehicle off the road, and then back on when he 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. While possible, turning leaming 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 leam a general representation. This would be time consuming, and also dangerous when training in traffic.

3.2 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 have 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 (See Figure 4). 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 which 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 5. To determine the appropriate value for a pixel in the transformed image, that 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 which implements a particular transformation is constant. In other words, assuming a planar world, the pixels which need to be sampled in the original image in order 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 fact by precomputing the pixels that need to be sampled in order 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

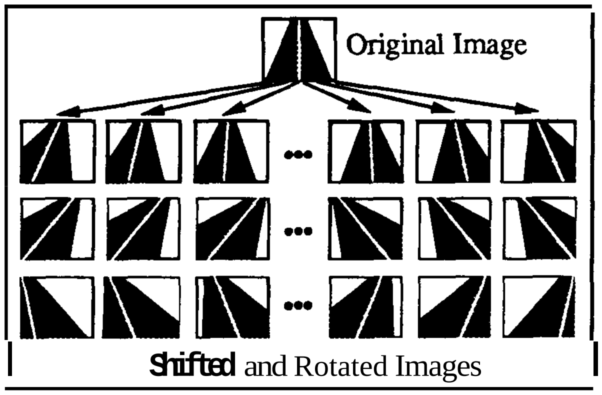

Figure 4: The single original video image is shifted and rotated to create multiple training exemplars in which the vehicle appears to be at different locations relative to the road.

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. But the elevation changes due to hills or dips in the road are small enough so as not to significantly violate the planar world assumption.

3.2.1 Extrapolating Missing Pixels

The less than complete overlap between the trapezoids of Figure 5 illustrates the need for one additional step in the image transformation scheme. The extra step involves determining values for pixels which have no corresponding pixel in the original image. Consider the transformation illustrated in Figure 6. To make it appear that the vehicle is situated one meter to the right of its position 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. Notice the number of pixels per row whose value needs to be extrapolated is greater near the bottom of the image than at the top. This is because the one meter of unknown ground plane to the right of the visible

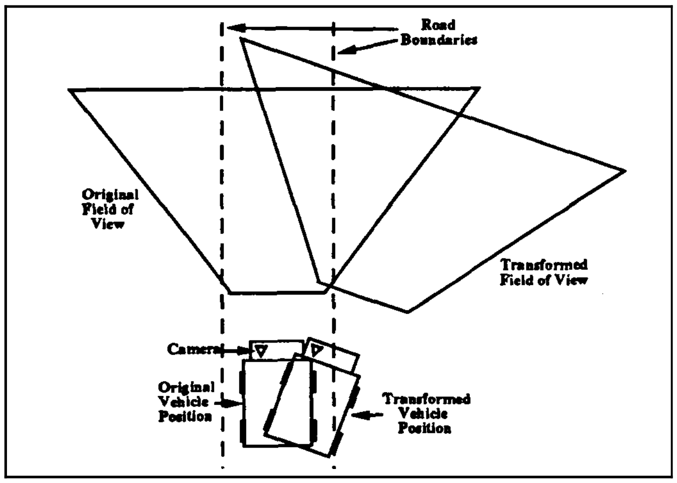
Road Boundaries
Original Field of View
Transformed Field of View
Camera
Original Vehicle Position
Transformed Vehicle Position

Figure 5: 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 used to direct resampling of the original image.

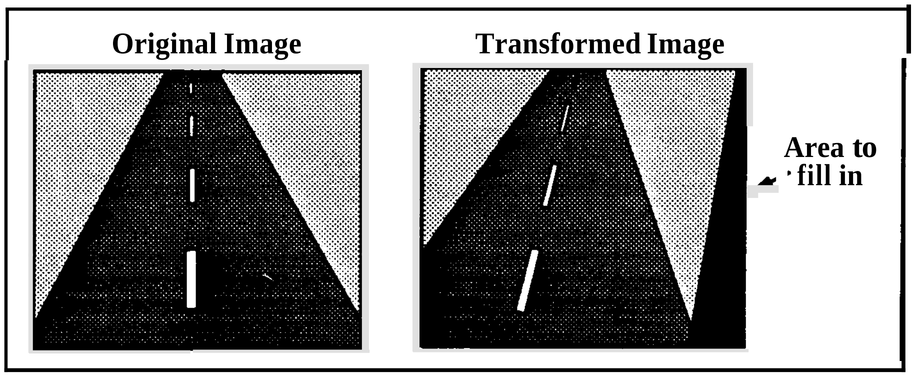

Figure 6: A schematic example of an original image, and a transformed image in which the vehicle appears one meter to the right of its initial position. The black region on the right of the transformed image corresponds to an unseen area in the original image. These pixels must be extrapolated from the information in the original image.

boundary in the original image covers more pixels at the bottom than at the top. We have experimented with two techniques for extrapolating values for these unknown pixels (See Figure 7).

In the first technique, to determine the value for a pixel that projects to the ground plane at point A in the transformed image, the closest ground plane point in the original viewing trapezoid (point B) is found. This point is then back-projected into the original image to find the appropriate pixel to sample. The image in the top right shows the sampling performed to fill in the missing pixel using this extrapolation scheme. The problem with this technique is that it results in the "smearing" of the image approximately along rows of the image, as illustrated in the middle image of Figure 8. In this figure, the leftmost image represents an actual reduced resolution image of a two-lane road coming from the camera. Notice the painted lines delineating the center and right boundaries of the lane. The middle image shows the original image transformed to make it appear that the vehicle is one meter to the right of its original position using the extrapolation technique described above. The line down the right side of the road can be seen smearing to the right where it intersects the border of the original image. Because the length of this smear is highly correlated with the correct

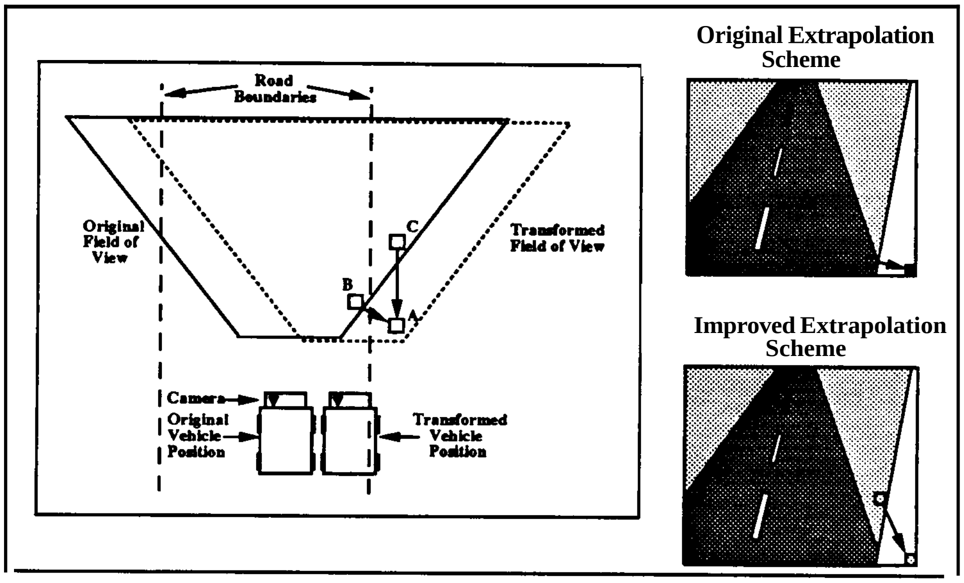
Road Boundaries
Original Field of View
Transformed Field of View
A
B
C
Camera
Original Vehicle Position
Transformed Vehicle Position
Original Extrapolation Scheme
Improved Extrapolation Scheme

Figure 7: An aerial view (left) and image based view (right) of the two techniques used to extrapolate the values for unknown pixels. See text for explanation.

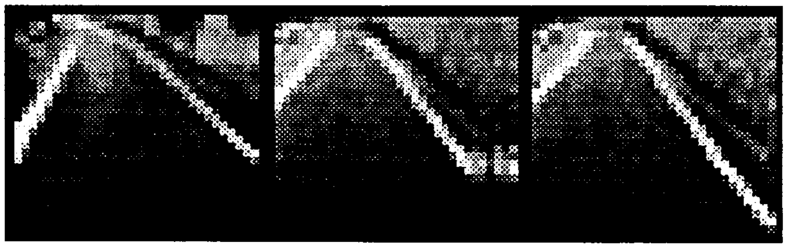

Figure 8: Three reduced resolution images of a two-lane mad with lines painted down the middle and right side. The left image is the original coming directly from the camera. The middle image was created by shifting the original image to make it appear the vehicle was situated one meter to the right of its original position using the first extrapolation technique described in the text. The right image shows the same shift of the original image, but using the more realistic extrapolation technique.

steering direction, the network learns to depend on the size of this smear to predict the correct steering direction. When driving on its own however, this lateral smearing of features is not present, so the network performs poorly.

To eliminate this artifact of the transformation process, we implemented a more realistic extrapolation technique which relies on the fact that interesting features (like road edges and painted lane markers) normally run parallel to the road, and hence parallel to the vehicle's current direction. With this assumption, to extrapolate a value for the unknown pixel A in Figure 7, the appropriate ground plane point to sample from the original image's viewing trapezoid is not the closest point (point B), but 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).

The effect this improved extrapolation technique has on the transformed image can be seen schematically in the bottom image on the right of Figure 7. This technique results in extrapolation along the line connecting a missing pixel to the vanishing point, as illustrated in the lower right image. The realism advantage this extrapolation technique has over the previous scheme can be seen by comparing the image on the right of Figure 8 with the middle image. The line delineating the right side of the lane, which was unrealistically smeared using the previous method, is smoothly extended in the image on the right, which was created by shifting the original image by the same amount as in the middle image, but using the improved extrapolation method.

The improved transformation scheme certainly makes the transformed images look more realistic, but to test whether it improves the network's driving performance, we did the following experiment. We first collected actual two-lane road images like the one shown on the left side of Figure 8 along with the direction the driver was steering when the images were taken. We then trained two networks on this set of images. The first network was trained using the naive transformation scheme and the second using the improved transformation scheme. The magnitude of the shifts and rotations, along with the buffering scheme used in the training process are described in detail below. The networks were then tested on a disjoint set of real two-lane road images, and the steering direction dictated by the networks was compared with the person's steering direction on those images. The network trained using the more realistic transformation scheme exhibited 37% less steering error on the 100 test images than the network trained using the naive transformation scheme. In more detail, the amount of steering error a network produces is measured as the distance, in number of units (i.e. neurons), between the peak of the network's "hill" of activation in the output vector and the "correct" position, in this case the direction the person was actually steering in. This steering error measurement is illustrated in Figure 9. In this case, the network trained with the naive transformation technique had an average steering error across the 100 test images of 3.5 units, while the network trained with the realistic transformation technique had an average steering error of only 2.2 units.

3.3 Transforming the Steering Direction

As important as the technique for transforming the input images is the method used to determine the correct steering direction for each of the transformed images. The c o m t 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 [16]. 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.

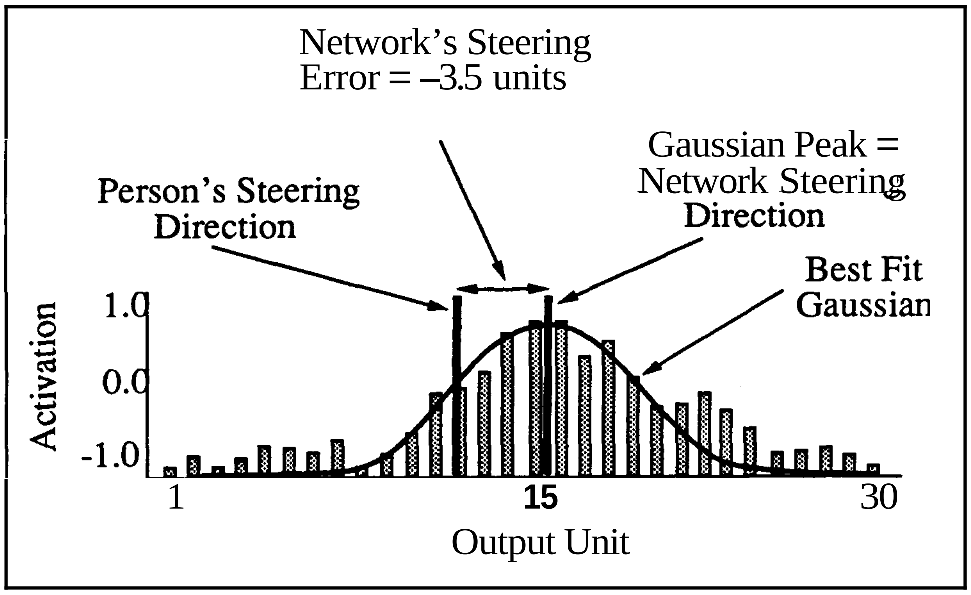

Figure 9: To calculate a network's steering error the best fit gaussian is found to the network's output activation profile. The distance between the peak of the best fit gaussian and the position in the output vector representing the reference steering direction (in this case the person's steering direction) is calculated. This distance, measured in units or neurons between the two positions, is defined to be the network's steering error.

The idea underlying pure pursuit steering is illustrated in Figure 10. 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 the 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

$$r = \frac{l^2 + d^2}{2d}$$

where r is the steering radius *1* is the lookahead distance and d is the distance from point T the vehicle would end up at if driven straight ahead from point B for distance I. The displacement d can be determined using the following formula:

$$d = \cos\theta \cdot (d_p + s + l \tan 6)$$

where d_p is the distance from point T the vehicle would end up if it drove straight ahead from point **A** for the lookahead distance I, s is the horizontal distance from point **A** to **B,** and 8 is the vehicle rotation from point **A** to B. The quantity d_p can be calculated using the following equation:

$$d_p = r_p - \sqrt{r_p^2 - l^2}$$

where r_p 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 towards. Empirically, I 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 the vehicle will travel in 2-3 seconds.

Interestingly, with this empirically determined rule for choosing the lookahead distance, the pure pursuit model of steering is a fairly good approximationto how people actually steer. Reid, Solowka and Billing [12] found that at 50km/h, human subjects responded to a 1m lateral vehicle displacement with a steering radius ranging from 511m to 1194m. With

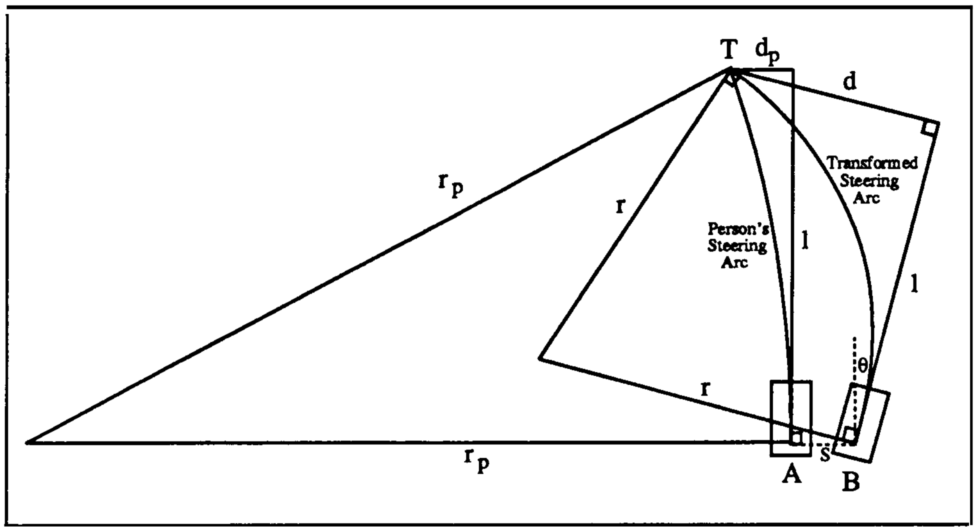

Figure 10: Illustration of the "pure pursuit" model of steering. See text for explanation.

a lookahead equal to the distance the vehicle will travel in 2.3 seconds, the pure pursuit model dictates a steering radius of 594m, within the range of human responses. Similarly, human subjects reacted to a 1 degree heading error relative to the current mad direction with a steering radius ranging from 719m to 970m. Again using the 2.3 second travel distance for lookahead, the pure pursuit steering model's dictated radius of 945m 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 and/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 and/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 mad or a multi lane 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 "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 may see them in". Similarly, the technique for transforming the steering direction for each of these new training images is equivalent to telling the network "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, but instead acquires this knowledge through training, the system is able to drive in a wide variety of circumstances, as will be seen later in the chapter.

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 of a problem, since the transformed training exemplars maintain variety in the training set.

3.4 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, while older patterns are removed. We have experimented with four techniques for determining which patterns to replace. The first is to replace oldest patterns first. Using this scheme, the training pattern buffer represents a history of the driving situations encountered recently. But if the driving situation remains unchanged for a period of time, such as during an extended right turn, the buffer will loose its diversity and become filled with right turn patterns. The second technique is to randomly choose old patterns to be replaced by new ones. Using 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 was to replace those 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 the patterns the network was performing best on, and leave in the training set those images the network was still having trouble with. The problem with this technique results from the fact that the human driver doesn't always steer in the correct direction. Occasionally he may have a lapse of attention for a moment 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 there in the training buffer for a long time, since the network will have trouble outputting a steering response to match the person's incorrect steering command. In fact, using this replacement technique, the only way the pattern would be removed from the training set would be if the network leamed to duplicate the incorrect steering response, obviously not a desired outcome. I considered replacing both the patterns with the lowest error and the patterns with the highest error, but decided against it since high network error on a pattern might also result on 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 ensured that even if the network never leams to produce the same steering response as the person on an image, that image will eventually be eliminated from the training set.

While this augmented lowest-error-replacement technique did a reasonable job of maintaining diversity in the training set, we found a more straightforward way of accomplishing the same result. To make sure the buffer of training patterns does not become biased towards one steering direction, we add a constraint to ensure that the mean steering direction of all the patterns in the buffer is as close to straight ahead as possible. When choosing the pattern to replace, I select the pattern whose replacement will bring the average steering direction closest to straight. For instance, 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 towards straight ahead. If the buffer already had a straight ahead average steering direction, then an old pattern requiring a similar steering direction the new one would be replaced in order 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,

3.5 Training Details

The final details required to specify the trainiig on-the-fly process are the number and magnitude of transformations to use for training the network. The following quantities have been 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 describe above 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 0.6m results in the road shifting approximately 1/3 of the way across the input image at the bottom.

Before the randomly selected shift and rotation is performed on the original image, the steering direction that would be appropriate for the resulting transformed image is computed using the formulas given above. If the resulting steering direction is sharper than the sharpest turn representable by the network's output (usually a turn with a 20m 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 vehicle is heading sharply to the left, the network is able to devote more of its representation capability to handling plausible scenarios.

The 14 transformed training pattems, 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 above. After this replacement process, one forward and one backward pass of the back-propagation algorithm is performed on the 200 exemplars to update the network's weights, using a leaming rate of 0.01 and a momentum of 0.8. The entire process is then repeated. Each cycle

requires approximately 2.5 seconds on the three Sun Sparcstations onboard the vehicle. One of the Sparcstation performs the sensor image acquisition and preprocessing, the second implements the neural network simulation, and the third takes care of communicating with the vehicle controller and displaying system parameters for the human observer. 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 per cycle, training takes approximately four 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 miles per hour.

4 Performance Improvement Using Transformations

The performance advantage this technique of transforming and buffering training patterns offers over the more naive methods of training on real sensor data is illustrated in Figure 11. This graph shows the vehicle's displacement from the road center measured as three different networks drove at 4 mph over a 100 meter section of a single lane paved bike path which included a straight stretch and turns to the left and right. The three networks were trained over a 150 meter stretch of the path which was disjoint from the test section and which 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 forward and backward pass of back-propagation was performed on that training exemplar, and then the process was repeated. The second network, labeled "+trans -buff', was trained using the following technique. An image was digitized from the camera and then transformed 14 times to create 15 new training patterns as described above. A forward and backwards pass of back-propagation was then 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 above 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

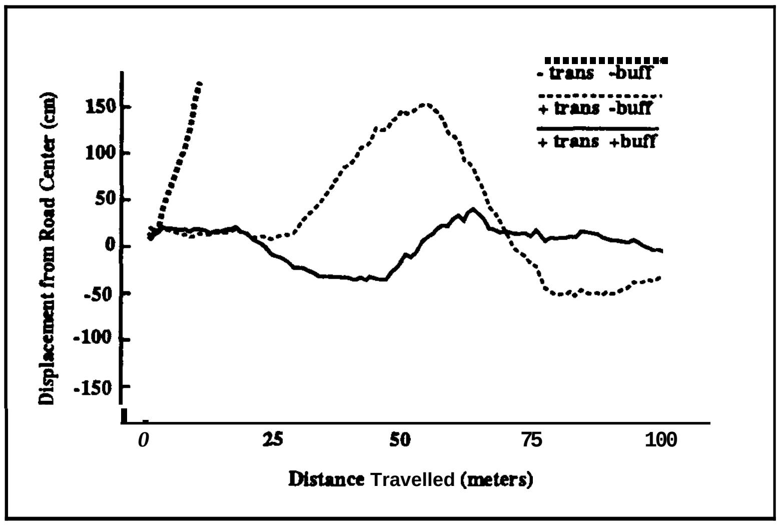

Figure 11: Vehicle displacement from the road center as the Navlab was driven by networks trained using three different techniques.

quantity of data the networks were trained on, only its 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, since it performed a forward and backward pass 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 towards turning right. Because of the increased diversity provided by the image transformation scheme, the second network performed much better than the first. It was able to follow the entire test stretch of mad. However it still had a tendency to steer too much to the right, as illustrated in the graph by the vehicle's positive displacement over most of the test run. In fact, the mean position of the vehicle was 28.9cm 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. Quantitatively, the standard deviation of this network's displacement was 62.7cm.

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. While the third network drove, the average position of the vehicle was 2.7cm right of center, with a standard deviation of only 14.8cm. This represents a 423% 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 one, however 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 miles per hour along the 100meter test section of road, the average position of the vehicle was 1.6cm right of center, with a standard deviation of 7.2cm. Under human control, the average position of the vehicle was 4.0cm right of center, with a standard deviation of 5.47cm. The average distance the vehicle was from the road center while the person drove was 5.70cm. It appears that the human driver, while more consistent than the network, had an inaccurate estimate of the vehicle's centerline, and therefore drove slightly right of the road center. Studies of human driving performance have found similar steady state errors and variances in vehicle lateral position. Blaauw [Blaauw, 19821 found consistent displacements of up to 7cm were not uncommon when people drove on highways. Also for highway driving, Blaauw reports standard deviations in lateral error up to 16.6cm.

5 Results and Comparison

The competence of the ALVINN system is also demonstrate by the range of situations in which it has successfully driven.

The training on-the-fly scheme gives ALVI" a flexibility which is novel among autonomous navigation systems. It has allowed me to successfully train individual networks to drive in a variety of situations, in-

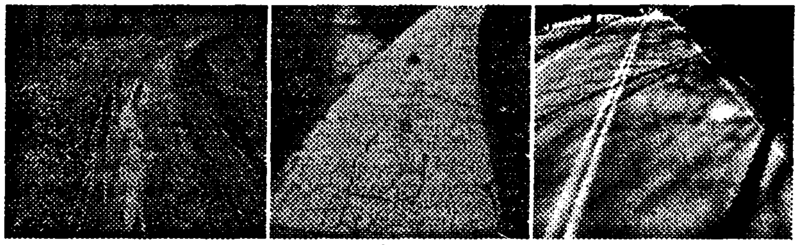
Figure 12: Video images taken on three of the road types ALVINN modules have been trained to handle. They are, from left to right, a single-lane dirt access road, a single-lane paved bicycle path, and a lined two-lane highway.

cluding a single-lane dirt access road, a single-lane paved bicycle path, a two-lane suburban neighborhood street, and a lined two-lane highway *(See* Figure 12). 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, to avoid collisions in obstacle rich environments, and to follow alongside railroad tracks. ***ALVINN*** networks have driven without intervention for distances of up to 22 miles. In addition, since determining the steering direction from the input image merely involves a forward sweep through the network, the system is able to process 15 images per second, allowing it to drive at up to *55* miles per hour. This is over four times faster than any other sensor-based autonomous system using the same processing hardware, has driven the Navlab [3] [8].

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, while hand programmed systems have been developed to drive in some of the individual domains **ALVINN** can handle [3] [8] [14] [4], 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 which correlate with the correct steering direction. When trained on multi-lane roads, the network develops hidden unit feature detectors for the lines painted on the road, while in single-lane driving situations, the detectors developed are sensitive to road edges and road-shaped regions of similar intensity in the image. For a more detailed analysis of ALVINN's internal representations see [9] [10].

This ability to utilize arbitrary image features can be problematic. This was the case 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 non-road, 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 so as to minimize the effects of transient image features.

On the other hand, experience has shown that it is more efficient to train several domain specific networks for circumstances like one-lane vs. two-lane driving, instead training a single network for all situations. To prevent this network specificity from reducing ALVINN's generality, we are currently implementing connectionist and non-connectionist techniques for combining networks trained for different driving situations. Using a simple rule-based priority system similar to the subsumption architecture [13, we have combined a road following network and an obstacle avoidance network. The road following network uses video camera input to follow a single-lane road. The obstacle avoidance network uses laser rangefinder images as input. It is trained to swerve appropriately to prevent a collision when confronted with obstacles and to drive straight when the terrain ahead is free of obstructions. The arbitration rule gives 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 is determined by the obstacle avoidance network. Together, the two networks and the arbitration rule comprise a system capable of staying on the road and swerving to prevent collisions.

To facilitate other rule-based arbitration techniques, we have adding to ***ALVI"*** a non-connectionist module which maintains the vehicle's PO-sition on a map [11]. Knowing its map position allows **ALVINN** to use arbitration rules such **as** "when on a stretch of two lane highway, rely primarily on the two lane highway network". This symbolic mapping module also allows ALVI" to make high level, goal-oriented decisions such **as** which way to turn at intersections and when to stop at a predetermined destination.

Finally, we **are** experimenting **with** connectionist techniques, such **as** the task decomposition architecture [7] and the meta-pi architecture [5], for combining networks more seamlessly than is possible with symbolic rules. These connectionist arbitration techniques will enable ***ALVI"*** to combine outputs from networks trained to perform the same task using different sensor modalities and to decide when a new expert must be trained to handle the current situation.

6 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. But teaching an artificial neural network to drive based on a person's driving behavior presents a number of challenges. Prominent among these 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 which make training set variety difficult to maintain are temporal correlations and the limited range of situations encountered. Extended intervals of nearly identical sensor input can bias **a** network's internal representation and reduce later 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 in this chapter 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.

Acknowledgements

The principle support for the Navlab has come from DARPA, under contracts DACA76-85-C-0019, DACA76-85-C-0003 and DACA76-85-C-0002. This research was also funded in part by a grant from Fujitsu Corporation.

References

[Blaauw, 19821 Blaauw, G.J. (1982) Driving experience and task demands in simulator and instrumented car: A validation study, ***Human Factors, Vol.24,*** pp. 473-486.

[1] Brooks, R.A. (1986) A robust layered control system for a mobile robot. IEEE ***Journal of Robotics and Automation,*** vol. RA-2, no. 1, pp. 14-23, April 1986.

[2] Cottrell, G.W. (1990) Extracting features from faces using compression networks: Face, identity, emotion and gender recognition using holons. In Connectionist Models: Proc. of the 1990 Summer School, David Touretzky (Ed.), Morgan Kaufmann, San Mateo, CA.

[3] Crisman, J.D. and Thorpe C.E. (1990) Color vision for road following. In ***Vision and Navigation: The CMU Navlab*** Charles Thorpe (Ed.), Kluwer Academic Publishers, Boston, MA.

[4] Dickmanns, E.D. and Zapp, A. (1987) Autonomous high speed road vehicle guidance by computer vision. ***Proceedings of the 10th World Congress on Automatic Control, Vol.4,*** Munich, West Germany.

[5] Hampshire, J.B., Waibel A.H. (1989) The meta-pi network: Building distributed knowledge representations for robust pattern recognition. Camegie Mellon Technical Report CMU-CS-89-166-R. August, 1989.

[6] LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., and Jackel, L.D. (1989) Backpropagation applied to handwritten zip code recognition. ***Neural Computation 1(4).***

[7] Jacobs, R.A., Jordan, M.I., Barto, A.G. (1990) Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. Univ. of Massachusetts Computer and Information Science Technical Report 90-27, March 1990.

[8] Kluge, K. and Thorpe C.E. (1990) Explicit models for robot road following. In ***Vision and Navigation: The CMU Navlab*** Charles Thorpe (Ed.), Kluwer Academic Publishers, Boston, MA.

[9] Pomerleau, D.A. (1989) ALVINN: *An* Autonomous Land Vehicle In a Neural Network, ***Advances in Neural Information Processing System, 1,*** D.S. Touretzky (Ed.), Morgan Kaufmann, San Mateo, CA.

[10] Pomerleau, D.A. (1990) Neural network based autonomous navigation. In ***Vision and Navigation: The CMU Navlab*** Charles Thorpe (Ed,), Kluwer Academic Publishers, Boston, MA.

[11] Pomerleau, D.A., Gowdy, J., Thorpe, C.E. (1991) Combining artificial neural networks and symbolic processing for autonomous robot guidance. ***Engineering Applications of Artificial Intelligence, 4:4*** pp. 279-285.

[12] Reid, L.D., Solowka, E.N., and Billing, A.M. (1981) A systematic study of driver steering behaviour. ***Ergonomics Vol.24,*** pp. 447462.

[13] Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986) Learning internal representations by error propagation. In D.E. Rumelhart and J.L. McClelland (Eds.) ***Parallel Distributed Processing :Explorations in the Microstructures of Cognition. Vol. I: Foundations.*** Bradford Books/MIT Press, Cambridge, MA.

[14] Turk, M. A., Morgenthaler, D. G., Gremban, K. D., Marra, M. (1988) VITS – A vision system for autonomous land vehicle navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 10.*

[15] Waibel, A., Hanazawa, T., Hinton, G., Shikano, K.. Lang, K. (1988) Phoneme recognition: Neural Networks vs. Hidden Markov Models. *Proceedings from Int. Conf. on Acoustics, Speech and Signal Processing,* New York, New York.

[16] Wallace, R., Stentz, A., Thorpe, C., Moravec, H., Whittaker, W., Kanade, T. (1985) First results in robot mad-following. *Proceedings of Int. Joint Con. on Artificial Intelligence.*