Expectation-Based Selective Attention for Visual Monitoring and Control of a Robot Vehicle

Shumeet Baluja & Dean A. Pomerleau
baluja@cs.cmu.edu, pomerleau@ri.cmu.edu
School of Computer Science and The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA. 15213
(412) 268-2339

ABSTRACT

Reliable vision-based control of an autonomous vehicle requires the ability to focus attention on the important features in an input scene. Previous work with an autonomous lane following system, ALVINN [Pomerleau, 1993], has yielded good results in uncluttered conditions. This paper presents an artificial neural network based learning approach for handling difficult scenes which will confuse the ALVINN system. This work presents a mechanism for achieving task-specific focus of attention by exploiting temporal coherence. A saliency map, which is based upon a computed expectation of the contents of the inputs in the next time step, indicates which regions of the input retina are important for performing the task. The saliency map can be used to accentuate the features which are important for the task, and de-emphasize those which are not.

1. INTRODUCTION

For the task of vision-based autonomous road following, the goal is to control a robot vehicle by analyzing an image of the road ahead. The direction of travel should be chosen based on the location of important features like lane markings and road edges. This is a difficult task since the scene ahead is often cluttered with extraneous features such as other vehicles, pedestrians, trees, crosswalks, road signs and other objects that can appear on or around a roadway. For the general task of autonomous navigation, these extra features are extremely important. However, for the restricted tasks of staying in the current lane or lane-marker detection, many of these features can be distracting. While we have had significant success on the road following task using simple feed-forward artificial neural networks (ANNs) to map images of the road into steering commands [Pomerleau, 1993], these simple methods fail when presented with cluttered environments like those encountered when driving in heavy traffic or on city streets (see Figure 1).

Figure 1: Typical scene to be processed in vision-based autonomous navigation tasks.

One of the methods by which humans function in the presence of many distracting features is to selectively attend to only portions of the input signal. A commonly accepted theory is that the limited processing capacity of humans is a fundamental constraint which imposes the need for selectivity in processing [Allport, 1989][Broadbent, 1982]. Although a machine may not be subject to the same limitations as a human, distracting features remain harmful. Most machine learning, artificial intelligence, and vision-based algorithms are not robust to unanticipated forms of noise, and improve performance with fewer distracting features in the input.

To discriminate between relevant and irrelevant inputs, it is first necessary to understand the requirements of the task. Importance of particular features, or "saliency" as defined in this work, is task-specific. For different tasks, different subsets of the same inputs may be important.

In this study, the attention of the vision-based processing system for lane tracking is dynamically focused only on the relevant inputs by masking out noise or distracting features. Because of the potentially high degree of similarity between the relevant features and the noise, this filtering is often extremely difficult. A means by which humans perform well on this task, even in the presence of large amounts of noise, is through the use of *expectations* to guide where to focus attention. Once the important features in the current input are found, an expectation of what the important features in the next inputs will be, as well as where they will be, can be formed. Similar approaches, which did not use learning, have been explored by Dickmanns and his colleagues [Dickmanns, 1992]; they will be discussed later.

Since different tasks require the processing of different features, ***expectations do not need to be formed for all of the features***. Rather, expectations are only required for the relevant inputs. Assuming an image of the road ahead with a fast camera sampling rate, the locations of the important features for staying on the road can be reliably predicted for the next time-step. Once the expectations of the next road-image are created, these expectations can be used in two manners. In the first use of expectation, inputs which are inconsistent with these expectations can be de-emphasized from processing. This will de-emphasize unexpected inputs. By attenuating unexpected inputs, the effects of distractions in the periphery and on the road ahead are reduced, while the road's features become more prominent. Therefore, the task of staying on the road becomes easier.

The potential problem in using expectation to remove unexpected features is that many surprise features, such as other cars or objects on the road, should not be removed from the input. Although these objects may be only distractions for the lane-tracking task, they are important when considering the full task of navigation. These objects will need to be processed in order to avoid collisions. To address this, expectation can also be used, in parallel, in an opposite manner. The second use of expectation is to emphasize anything which does *not match* expectation. This emphasizes the unexpected features. A high level schematic of the use of expectation is shown in Figure 2. Note that if memory or state information is allowed, the expectations can be formed based upon more than the single previous input.

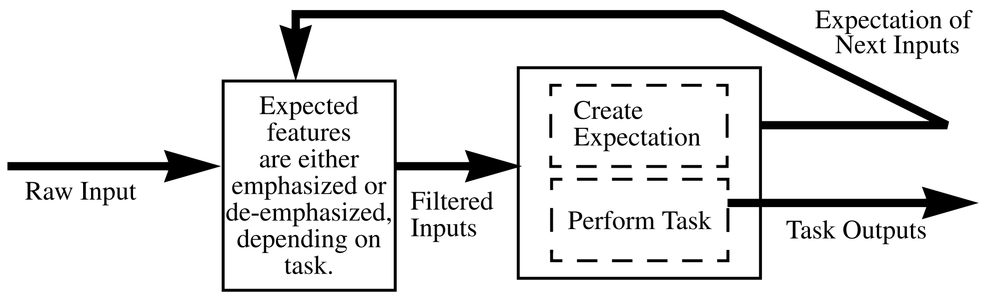

Figure 2: High-level schematic of the use of expectation. Note that the process which creates the next time step's expectation and the process which solves the task are closely related since the expectation which is created is task-specific. The module to create expectations and perform the task may contain memory or state information.

The motivation for employing this bipolar usage of expectation is largely pragmatic; it maps well to the structure of many real-world machine learning and adaptive systems. For example, consider Carnegie Mellon's Navlab autonomous navigation system. The road-following module (ALVINN, [Pomerleau, 1993]) is separate from the obstacle avoidance modules [Thorpe, 1991]. Each module can use different inputs, or different input filtering algorithms, to match its specific requirements. The first role of expectation, in which unexpected features are de-emphasized, is appropriate for the road-following module. The second role of expectation, to emphasize unexpected features, is appropriate for the obstacle detection modules. A higher level system arbitrates between these and other modules (see Figure 3).

The above model is simplified, because the appropriate use of expectations may depend not only on which module is using the filtered image, but also the state of the module. For example, consider designing an obstacle tracking module. To initially find obstacles, the expected features may be de-emphasized. How-

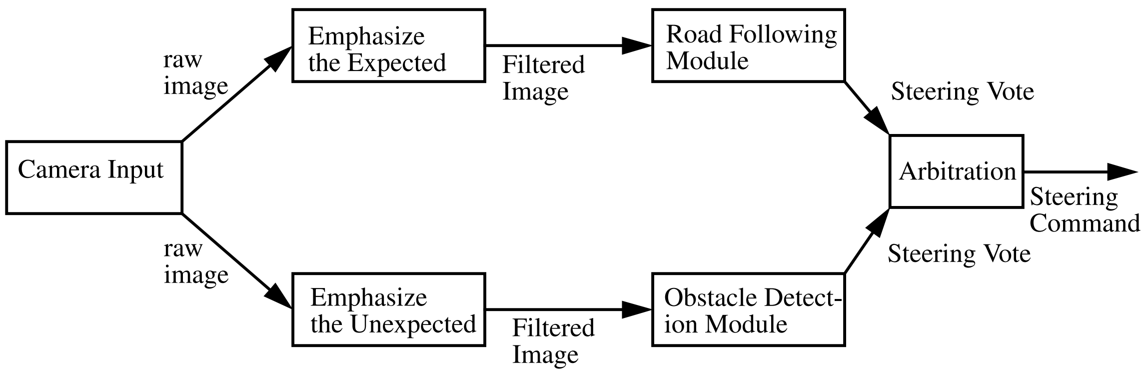

Figure 3: A simplified model for using both forms of expectation in the driving domain.

ever, to monitor other moving vehicles, once they are found, the first use of expectation may again be appropriate. Here, it is vital to note the role of task-specificity in forming expectations. The expectations which are required for road-following will not be the same as those which are required for tracking a potential obstacle. Rather, the expectations must be specific for the task being performed. In the next section, we describe how task-specific expectations can be created. These are used as a basis for filtering input images for the road following module. Extensions to the system for the obstacle avoidance modules are briefly described.

2. TASK-SPECIFIC FOCUS OF ATTENTION

In order to direct processing to only the relevant inputs, the relevant inputs must first be determined. Saliency maps are tools used to indicate the relative importance of different regions of the input scene. In many studies, saliency maps have been constructed in a bottom-up manner [Clark & Ferrier, 1992][Koch & Ullman, 1985]. In one of its simplest forms, the saliency map may contain only a binary value for each input, which indicates whether the particular input is relevant. More commonly, a real valued weight, signifying relevance, is associated with each input. A very cursory summary of one bottom-up approach to creating saliency maps is that multiple different task-specific feature detectors are placed around the input image. Each type of feature detector may contain a weight associated with it, to signify the relative importance of the particular feature. The regions of the image which contain high weighted sums of the detected features are the portion of the scene which are focused upon. Top-down knowledge can be incorporated in deciding which features are used, the weightings of the features, and how many regions are focused upon, see for example [Ahmad, 1991].

Although saliency maps are integral to the techniques presented in this study, the approach used to create and use them is very different than the one described above. The features and their weightings are developed as a neural network learns to use the features. No *a priori* top-down information is assumed. In the method proposed here, the expectation of what the features will be in the *next* frame plays a key role in determining which portions of the visual scene *will be* focused upon.

Simultaneously learning to perform a vision-based task and filtering out clutter in the input scene is doubly difficult because of a “chicken-and-egg problem.” It is hard to learn which features to attend to before knowing how to perform the task, and it is hard to learn how to perform the task before knowing on which features to attend. In this paper, we address this problem by deriving a “saliency map” of the image from a neural network’s internal representation, as the network is being trained, which highlights regions of the scene considered to be important. The saliency map, which is a function of the expected input image, the current input image and the previous inputs, is used as feedback to focus the attention of the network’s processing on subsequent images. The proposed technique overcomes the chicken-and-egg problem by simultaneously learning how to identify which aspects of the scene are important, and how to use them to perform a task.

3. USING THE NETWORK’S HIDDEN LAYER TO DETERMINE TASK-SPECIFIC IMPORTANCE

The first step in creating a conceptually based saliency map (one which is driven by the task to be performed) is determining which inputs are important. The method described in this section is based on *Input Reconstruction Reliability Estimation (IRRE)*, proposed by [Pomerleau, 1993]. IRRE is used to estimate how reliable a network's outputs are. The reliability estimation in IRRE is made by using the hidden units to both reconstruct the input image and complete the main task. The heuristic used to relate the reconstructed inputs to the task output is the greater the similarity between the actual input image and the reconstructed input image, the more the internal representation has captured the input features, and therefore the more reliable the network's response. This has been shown to work well empirically [Pomerleau, 1993]. Figure 4 provides a schematic of IRRE. Note that the weights between the input and hidden layers are trained to reduce *both task and reconstruction error.*

IRRE has successfully been used as a method of arbitrating between multiple "expert" networks. For example, consider the case in which one network (Network-1) was trained to steer an autonomous vehicle on a single lane road, and another on a double lane road (Network-2). The reconstruction of the inputs on a two lane road will be very poor for Network-1, while Network-2 will be able to accurately reconstruct the inputs. The performance will switch on single lane roads. The reconstruction error provides a method to determine which network to trust when road-types change. In contrast to other arbitration methods [Jordan & Jacobs, 1994], task allocation to experts in not done automatically, and only one expert is used at a time. As pointed out by Pomerleau, a potentially large drawback of IRRE is the use of the hidden layer to encode all of the features in the image, rather than only the ones required for solving the particular task [Pomer-

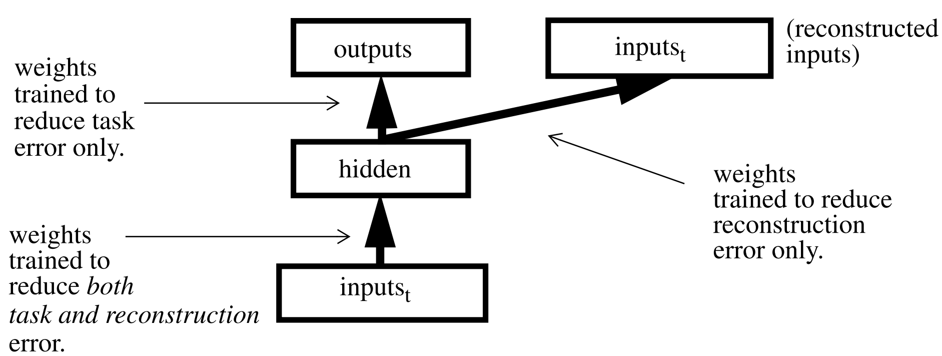

Figure 4: IRRE: using the hidden layer to encode information for reconstruction and task completion.

leau, 1993]. This problem is addressed below.

If a strictly layered (connections are only between adjacent layers) feed-forward neural network can solve a given task, the activations of the hidden layer contain, in some form, the important information for this task from the input layer. One method of finding out what information is contained within the hidden layer is to attempt to reconstruct the original input image, *based solely upon the representation developed in the hidden layer*. Like IRRE, the input image is reconstructed from the activations of the units in the hidden layer. *Unlike IRRE, the hidden units are not trained to reduce reconstruction error, they are only trained to solve the particular task.* The input reconstruction is done by changing the weights from the network's hidden layer to the reconstruction outputs only. The weights from the inputs to the hidden layer are ***not*** affected by the errors in reconstruction. Therefore, all of the capacity and representations developed in the hidden layer is devoted only to solving the task, see Figure 5.

The weight updates are based on the standard backpropagation algorithm. In the architecture shown in Figure 4, the task error is used to train the weights between the task-output and hidden layer, and the reconstruction error is used to train the weights between the reconstruction and hidden layer. Both of these errors

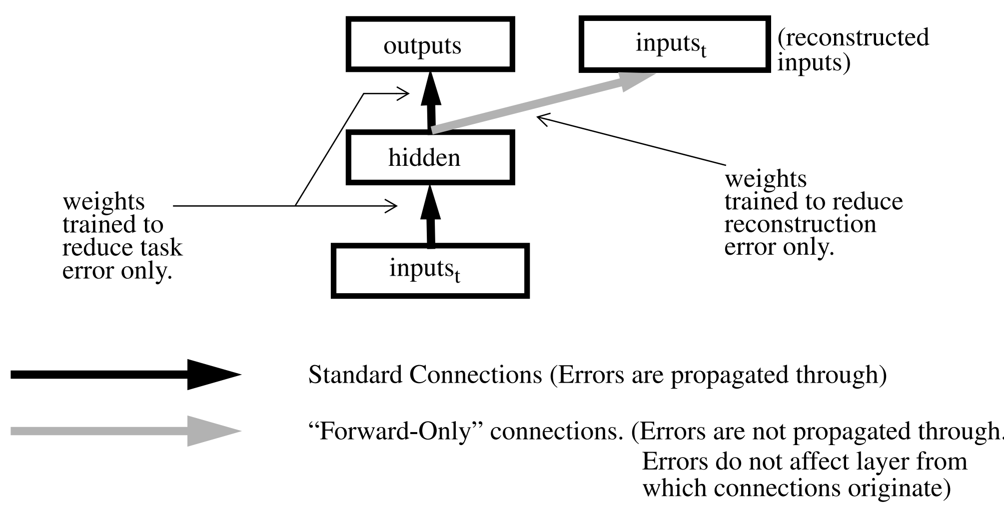

Figure 5: Using the activation of the hidden layer to reconstruct the input. The weights between the input and hidden layers are trained to only reduce task error, *not* reconstruction error. Extra hidden layers for reconstruction can be used.

are used to train the weights between the input and hidden layer. In the architecture shown in Figure 5, the errors from the reconstruction are not propagated through the hidden units; therefore, only the errors from the task-output are used to train the connections from the input to hidden layer.

The inputs will not be perfectly reconstructed in the output layer because the weights of the network between the input and hidden layers are not trained to perform reconstruction; they are trained to perform the main task. The information which is encoded in the activations of the hidden units corresponds to the information which the network uses to solve the task. *Information which is not relevant to the task will not be encoded in the hidden units*. Since the reconstruction of the inputs is based solely on the hidden units' activations, and the irrelevant inputs are not encoded in the hidden units' activations, the irrelevant inputs *cannot* be reconstructed. [1]

Both IRRE, and this modified IRRE, are related to auto-encoding networks and principal components analysis (PCA). In auto-encoding networks, the output is trained to reproduce the input layer [Cottrell, 1990]. The hidden layer, which is usually used as a bottleneck, captures important features for reconstructing the input. It has also been shown that under certain conditions, auto-encoding networks will capture the principal components of the inputs [Baldi & Hornik, 1989][Kramer, 1991]. The difference between auto-encoding networks and the networks employed in this study is that the networks used here were not trained to reproduce the input layer accurately; they were trained to perform well on a specific task. The inputs which can be reconstructed accurately are those which the hidden units have encoded to solve the task.

Although IRRE provides a method to determine which inputs the network has encoded, a notion of time is necessary in order to focus attention in *future* frames. Instead of attempting to reconstruct the current input, the network is trained to predict the *next* input, see Figure 6. The prediction is trained in a supervised manner, by using the next set of inputs to appear in the time sequence as the targets. The targets (the next

1. One exception to this is that features which are always present, such as inputs which contain nearly constant values over time, can be reconstructed regardless of relevance. However, these are easily eliminated through subtraction of the average image. Further, it should be noted that inputs which are constant during training and testing do not cause problems for networks; they are used by the network as bias inputs.

inputs) may contain noise or extraneous features. However, since the hidden units only encode information to solve the task, the network will be unable to construct the noise or extraneous features in its prediction.

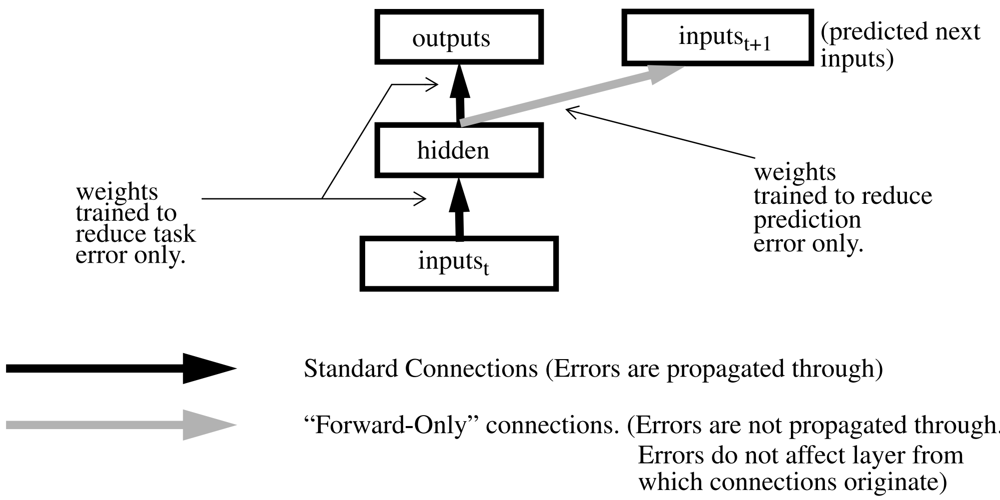

Figure 6: Using the activation of the hidden layer to predict the next inputs. The weights between the input and hidden layers are trained to only reduce task error, *not* prediction error. Extra hidden layers for prediction can be used.

3.1 Using the Differences Between Expectation and Realization

To this point, a method to create an expectation of what the next inputs will be has been described. There are two fundamentally different ways in which to interpret the difference between the expected next input and the actual next input. The first interpretation is that the difference between the expected and the actual input is the point of interest because it is a region which was ***not*** expected. This is applicable in the obstacle-avoidance system. This use of expectation has also been applied to anomaly detection in time-series data [Baluja, 1996].

In the second interpretation, the difference between the expected and actual inputs is considered noise. This interpretation is used throughout the rest of this paper. Processing is de-emphasized from the regions in which the difference is large. This procedures makes the assumption that there is enough information in the previous inputs to predict the values of the inputs which are important to the task in the next image. As will be shown, this method has the ability to remove spurious features and noise.

4. USING EXPECTATION TO REMOVE IRRELEVANT/DISTRACTING FEATURES

The expectation which was derived in the previous section can be used to filter noise from the current input (see Figure 7). For example, in a system which uses images scaled between -1.0 and +1.0 as input, a saliency map is created by using the absolute differences between the expectation of input image_{t+1} (derived from input image_t) and the actual input image_{t+1}. This difference image is scaled to the range of 0.0 to 1.0; with smaller differences closer to 1.0. The result is the saliency map. In order to emphasize these regions for the input, the saliency map is multiplied, pixel by pixel, with input image_{t+1}. The results after multiplication are used as inputs for time (t+1). This has the effect of lowering the activation (towards 0.0) for the inputs which do not match the expectation. The inputs which match the expectation are left unaltered.

The filtering procedure described above works well when the background of the image is a value close to 0.0. As will be shown in the lane-marking detection task in Section 5, this effectively eliminates distractions from the portions of the image in which the road appears, since road pixels often have intensities close to 0.0. It should be noted that this type of filtering will not work in domains in which the background value is not 0.0. In these domains, if the background can be determined, the activation can be moved towards that value. It should also be noted that the background does not necessarily have to be the same for all pixels in the inputs. Further, in tasks in which the background value cannot be determined, the expectations and saliency map can be used as extra inputs into the neural network [Baluja, 1996][Baluja & Maxion, 1996].

The chicken-and-egg problem mentioned earlier is now more evident. It is necessary to automatically determine the features which are important for solving the task before the task is solved. However, since the important features are task-specific, and are developed from the network's hidden layer, the task must

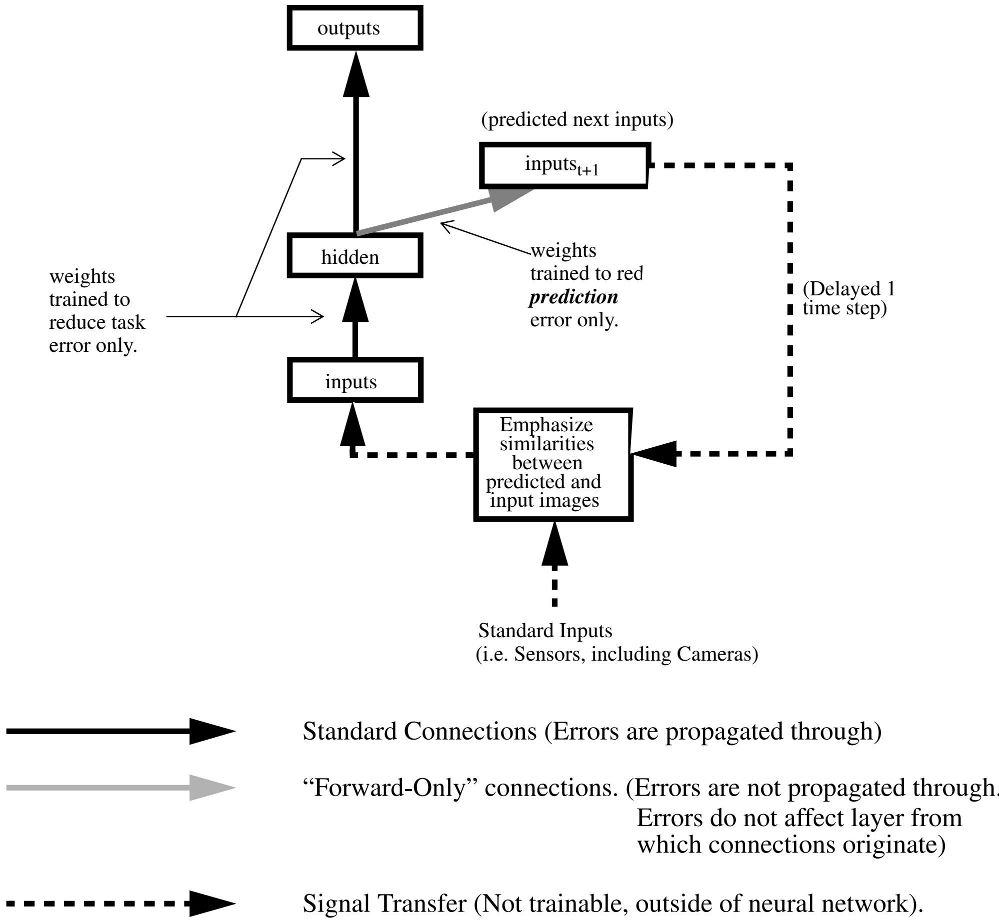

Figure 7: Prediction of next inputs, with feedback. The filtering is done to emphasize similarities between predicted and actual inputs.

first be solved. This problem is avoided because once a few of the important features are found (this is often possible without the feedback, since not all images contain the same extraneous features), the prediction becomes more accurate, and the expectations improve. The system can then bootstraps itself.

5. EMPIRICAL RESULTS: LANE TRACKING

ALVINN is an artificial neural network based perception system which learns to control CMU's Navlab vehicles by watching a person drive [Pomerleau, 1993] (see Figure 8). ALVINN's architecture consists of a single hidden layer backpropagation network. The input layer of the network is a 30x32 unit two dimen-

Figure 8: CMU Navlab-5 Vehicle.

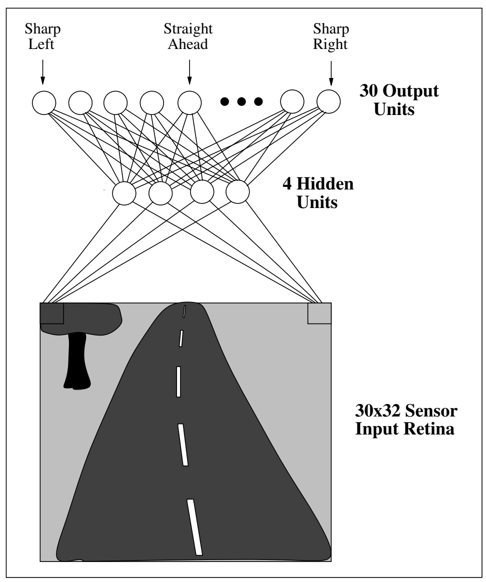

Figure 9: ALVINN architecture.

sional "retina" which receives input from the vehicle's video camera. The correct steering direction is determined from the activation of 30 output units, see Figure 9. The output units create a Gaussian centered around the correct steering direction, see Figure 10. If the Gaussian is centered around unit 1, this indicates the vehicle should make a sharp left, if the center is around unit 30, the vehicle should make a sharp right, etc. To teach the network to steer, ALVINN is shown video images from an onboard camera as a person drives. For each image, it is trained to output the steering direction in which the person is steering.

Recently, there has been an emphasis on using the ALVINN system as a driver warning device. In one of

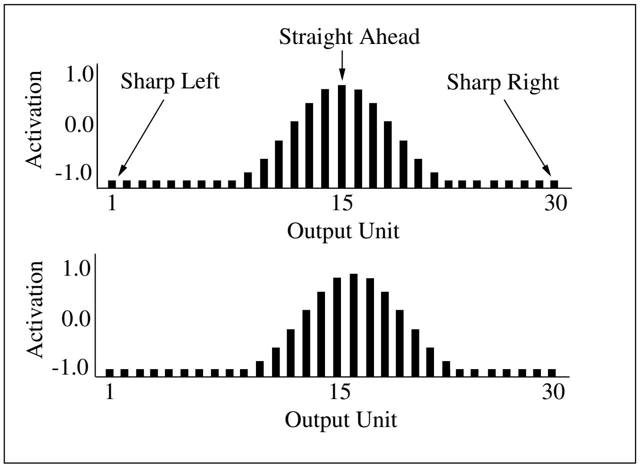

Figure 10: Gaussian Output Representation for the 30 output units, Top: Straight Ahead, Bottom: Slightly to the right. (from Pomerleau [1993]).

the proposed uses of this system, the model will warn drivers if they drift over lane markings (indicating that they may be entering a lane with on-coming traffic, or leaving the road, etc.). The system must be robust to distracting road features, such as road and off-road boundaries, cars, or other lane markings. Experiments for this task are described in the next section.

5.1 Experiment Details

Approximately 1200 images were gathered from a camera mounted on the left side of the Navlab 5 test vehicle, pointed downwards and slightly ahead of the vehicle. The images were gathered sequentially, at approximately 4-5 images per second. The Navlab was driven through city and residential neighborhoods around Pittsburgh, PA. The images were subsampled to a 30x32 pixel representation. In each of these images, the horizontal position of the lane marking in the 20th row of the input image was manually identified. The task is to produce a gaussian of activation in the outputs which is centered around the horizontal position of the lane marking in the 20th row of the image, given the full 30x32 pixel image as input. Sample input images and target outputs are shown in Figure 11.

In this task, it is vital to focus on only the relevant portions of the input. If all the raw input are used, the

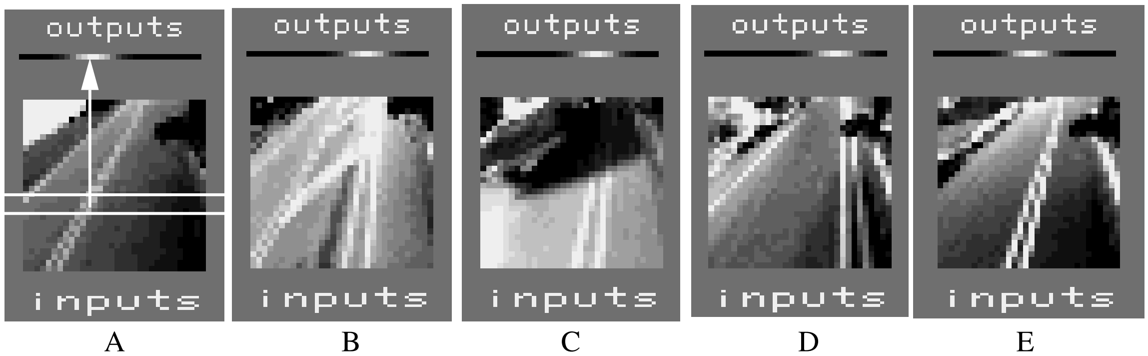

Figure 11: Five sample input images and target outputs. Image A shows the region from which the lane marking was manually selected.

artificial neural network can become confused by road edges, as shown in Figure 11a; by extraneous lane markings, as shown in Figure 11b, and reflections on the car itself (since the camera was located on the side of the vehicle), as shown in Figure 11d and e. With these images as input, an ANN may yield incorrect results, such as a Gaussian centered around the wrong output.

5.1.1. Training Specifics

In training the network, there are several problems which must be addressed. The first is that there is only a limited amount of training data. Assuming that the driver has directed the car well, the center line has probably stayed within a small region of the input image. Therefore, the network has not been trained to recognize lane markings outside the middle regions of the image. The second problem is that because the prediction task attempts to forecast future inputs, it is important to prevent the network from memorizing the image transitions in the training set. For example, had the driver chosen a slightly different action, the location of the important features in the next set of inputs may have been different.

In order to alleviate the problems with the training set, the training set was augmented as follows. In training, extra input images were created by translating the original images to the left or right by up to 5 columns; the unknown regions in each row were filled in with the last known pixel value of each row. The output was also translated either to the left or right by the same amount as the image. This translation yields usable images because the camera is pointed downwards. This is a standard method of generating training data for situations which the driver may not have encountered, such as cases in which the lane marking is at the left edge of the input. If the camera had been pointed more ahead of the vehicle, a more sophisticated image transformation would have been required to maintain the correct perspective [Pomerleau, 1993].

The sequential nature of focusing attention dictated that these images could not simply be added to the training set. For example, an image at time *t*, which is translated 3 columns to the left, should not be followed by an image at time *t+1* which is translated 3 columns to the right. If it were, the important features would jump a large distance, and this is unlikely to happen in practice. To avoid this problem, the expanded training set is used in the following manner: the image at time step *t+1* is chosen at a random translation which differs by, at most, ±1 columns from the translation of the image at time step *t*. This ensures that there are no artificially generated large jumps of the important features between consecutive time steps. As the network is trained through many passes through the training set, images are seen with different translations.

The training data used in these experiments was collected from a single driver. Since this system is to be used in a wide variety of situations, the system must be made robust. For example, in many images, driving straight, or steering slightly to the left or to the right may be an acceptable maneuver. To ensure that the network learns these possible transitions, for each input image, many potential "next input" images are created. The network is trained to predict all of these synthesized images. These images are created by translating images which are temporally close to the current image, to the left and right of the current offset in the current image (in a manner similar to the one described in the previous paragraphs). The errors between the predicted next state and all of these images are used for training the network, since any of the images are reasonable expectations for the next input. One interesting aspect of the training data in this domain is that images from recent *previous* time-steps can be used for this procedure, since many of the features remain consistent for short periods of time. This is not possible in domains in which consecutive input

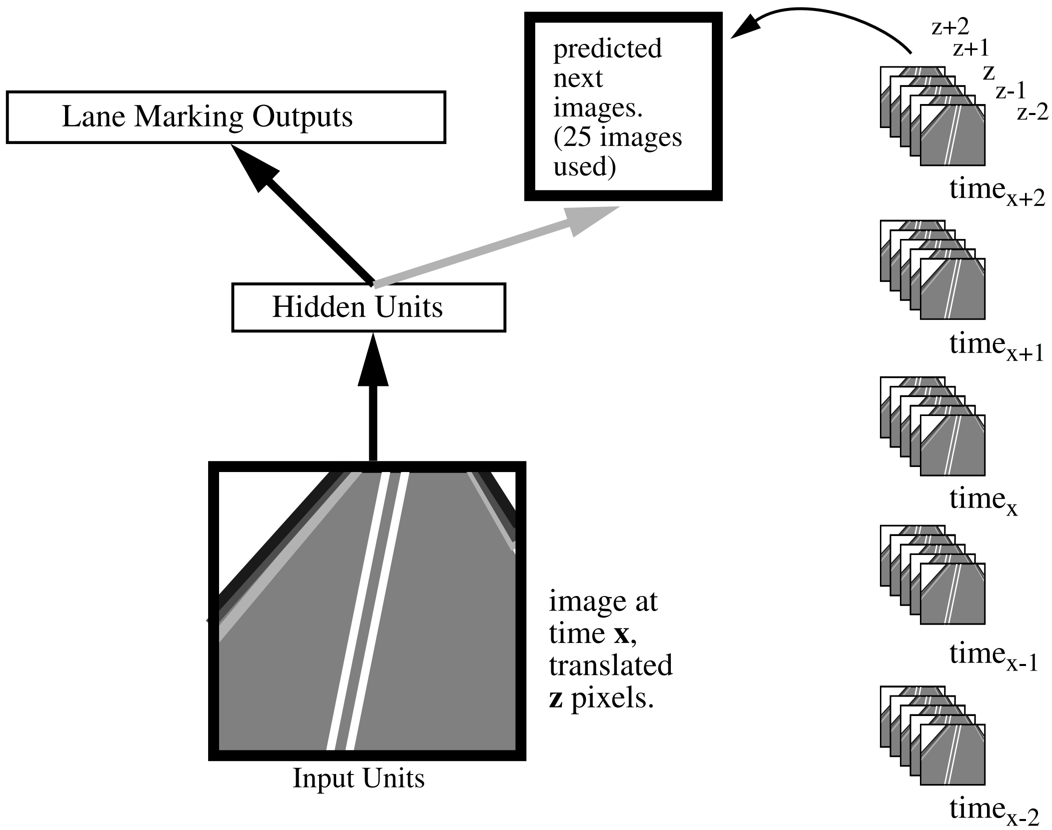

Figure 12: Training the network to account for multiple responses. Images are created by translating to the left and right ±2 columns of the input image's translation. Images from current, future, and previous time steps are also used.

images have large differences. In the experiment presented here, 25 "next input" images (5 time steps x 5 translations per time step) were paired with each input image. This process is displayed in Figure 12.

The first 800 images were used for training. The last 400 images were used for testing. Both training and testing images were used with translations. Overtraining was not encountered because of the use of random translations through training which kept the training set changing.

5.1.2. Results

After training in the manner described above, the results of this experiment were very promising. The lane tracker was able to remove various distractions from the images. Figure 13 shows 4 image triplets. The left image of each triplet displays the original, unfiltered, input image$_t$. The middle image is the predicted

$image_t$, made by the network with input of $image_{t-1}$. The right image is an image which has been filtered, pixel by pixel, by the absolute difference between the original $image_t$ and predicted $images_t$ (it has been filtered by the saliency map). The larger the difference between actual and predicted image, the more the pixel's activation is reduced. In triplet A, the edge of the road is bright enough to cause distractions. Since the bright edge was not expected, it is de-emphasized. In triplet B, the two lane markings may confuse the lane tracker, and cause it oscillate between the lane markings. In this image, the distracting lane-marking (the one on the left) is removed, since the lane marker on the right was correctly tracked in images before the distractor lane-marker appeared. In triplet C, a passing car is de-emphasized. The network does not have a model to predict the movement of passing cars, since these are not relevant for the lane-marker detection task. In each of these three triplets, the confusing attributes are removed from the image (shown in the right image of each triplet). Triplet D shows the limitations of this method; although the majority of the distracting lane markings are eliminated, all of them are not removed. Nonetheless, the most distracting lane marking was removed, and the ANN was not confused by the resulting image.

The expectations which are shown in column 2 of Figure 13 show that the expectation of where to find the lane marking and the road edges is not precisely defined. This is vital to ensure that the important features do not fall outside of the expected area. This is accomplished by the training method described in the previous section, which attempts to account for the many possible transitions from one time step to the next. If the important features fall outside the area in which they are expected, they may be de-emphasized in processing.

Quantitatively, the performance of the lane-tracker with the saliency map, measured by the difference between the estimated and actual position of the lane marking, revealed an average improvement of 20% over the lane-tracker without the saliency map. This improvement was revealed over multiple runs, with random initial weights in the ANN and different random translations chosen for the images during training and testing. Although a significant improvement, the improvement was not greater because many of the

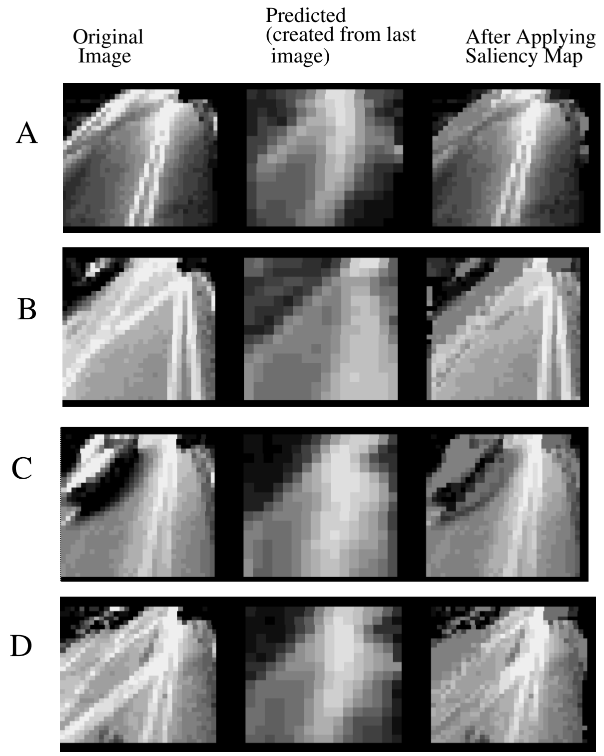

Figure 13: Left: original input image$_t$. **Middle:** predicted image$_t$, made by the network with input of image$_{t-1}$. **Right:** image which is filtered, pixel by pixel, by the difference between the original image$_t$ and predicted images$_t$. The larger the difference between actual and predicted image, the more the pixel's activation is reduced to 0.0 (a value of intermediate gray). The right image is used as the input to the neural network. The images are scaled between value of -1 and +1.

images in the test set do not contain noise. In the images without noise, a standard ANN can be used to accurately estimate the lane marking position. However, in domains such as lane-marker tracking or road following, tracking the wrong lane-marking even in only a few images may be catastrophic. Since the system will be designed to take control of the vehicle in hazardous situations, under no circumstances may the system be distracted by spurious lane markings or similar appearing features.

In terms of filtering, there are important differences between this application and other studies conducted in non-visual domains. First, the inputs in this domain are highly redundant. Therefore, the filtering does not have to be perfect. If important pixels are accidentally removed, there are many other pixels which contain enough information for recovery. Second, it is important to eliminate as many of the irrelevant pixels as possible, since they may contain information which directly conflicts with the correct information. Third, this task emphasizes the importance of dynamically re-evaluating the relevance of each of the inputs. As the images change, the relevant pixels change. Methods which do not consider dynamic re-evaluation of relevance are described in the next section.

One might be tempted to think that since the actual next image is filtered based upon the differences between it and the expected image, that it should be possible to just use the expected image as input, and ignore the actual image (provided that the predictions are accurate). However, this is not the case. Recall that the training procedure attempts to capture many possible valid transitions between images. Therefore, as can be seen in Figure 13, the location of the lane-markings is not precisely specified. Instead, the lane-markings are expected within a *region* of the input. The actual lane-markings, which are located within this region, give the basis for the prediction of the next lane-markings. This new information is incorporated to make accurate future predictions. If only the expected image had been used, the expected image would continually get more diffuse, until all information is lost. This is somewhat analogous to Kalman Filters, in which each measurement is used to revise the prediction of future measurements; the longer the predictions are continued with no readings, the larger the error in prediction will be.

It should be noted that expectations can also be used when the regions of interest are spatially discontinuous in consecutive time-steps. There is no prerequisite for smooth transitions of focusing regions, as is commonly accepted for biological systems [Umilta, 1988]. Examples of this are explored in depth in [Baluja, 1996].

6. RELATED WORK

This section describes the relation of expectation-based selective attention to input-relevance studies, Dickmanns' work with active vision, and other recurrent neural network architectures.

6.1 Previous Studies of Input-Relevance

Many studies of input relevance make the assumption that the best selected features for each decision point is the minimal set of features which is needed to classify every point [John, *et al.* 1994]. However, in many visual domains the smallest set of inputs needed to classify every example may be the entire set of inputs, while the set needed to classify each individual example may be very small. For example, consider visually tracking a moving object across a large pixel-based input. The goal may be to report the value of some attribute of the object, such as orientation. The inputs which are relevant to the task will change as the object moves. Assuming the object appears everywhere in the input with equal probability, no input can be considered more important than another when the entire training set is considered. However, when individual examples are considered, the position of the object gives relevance information; only the pixels which correspond to the position of the object are relevant. In many vision applications, choosing the smallest subset of relevant points at the training set level will provide little or no filtering. Assessment of input relevance for many vision tasks must be done dynamically rather than as a pre-processing step. However, if a measure of static relevance is desired, it can be obtained by integrating the saliency of inputs over time [Baluja, 1996].

6.2 Dickmanns' 4D Approach to Dynamic Vision

The work by Dickmanns [Dickmanns, 1992] is closely related to the work in this paper. Dickmanns' and his colleagues have studied the control of an autonomous vehicle with active vision. In contrast to the *covert* form of attention that has been the subject of this paper, active vision concentrates on *overt* attention. Active vision systems have mechanisms that can control camera parameters, such as orientation, focus, zoom, etc., to the requirements of external stimuli. A review of some active vision systems can be found in [Swain & Stricker, 1993][Swain, 1994][Blake & Yuille, 1992]. The underlying premise of Dickmanns work has been in using prediction of the future state to help guide attention, for example, by controlling the direction of a camera to acquire accurate position of landmarks.

Strong models of the vehicle motion, the appearance of objects of interest (such as the road, road-signs, and other vehicles), and the motion of these objects are encoded in the system; the motion of the vehicle and the other objects in the scene are modeled independently. Their system does not use learning. Initially, low level pixel processing is used to find where objects in the scene are located. If all of the objects in the scene are found and good models are available, a forward projection can be made of the scene in the next time steps. By examining the differences between the prediction and the actual scene, the models can be refined. Many of the models used by Dickmanns are based on extended Kalman Filters.

The work by Dickmanns and the work presented here differs in several key aspects. The first is in the amount of *a priori* knowledge that is used. In Dickmanns' approach, a tremendous amount of problem specific information is incorporated into creating the models which are used for prediction and control. In the approach presented in this paper, the main object is to automatically learn, from examples, the features which should be attended; therefore, very little knowledge about the domain is assumed. Learning is used for detecting the important features, developing the control strategy from the detected features, and also generating the predictive model. Second, Dickmanns' uses all of the previous time steps to make predictions. By using recursive estimation procedures, such as Kalman Filters, information from all of the previous images is combined to create predictions. In the experiments conducted here, only the single previous image was used for prediction (although this can easily be extended, as described later). Finally, the work by Dickmanns' attempts to explicitly model the transitions of the objects in the road with respect to the vehicle. The study presented here did not address the problem of obstacle avoidance or object detection; only road following was considered. However, an extension of this work for obstacle detection is to compare the prediction of the road with the actual image, and to emphasize the differences. This would highlight potential obstacles. This extension does not track the obstacles; instead, it treats each frame independently. To track the obstacles, like Dickmanns approach, a separate procedure must be used. One can consider using a separate obstacle tracking system which employs expectations to predict the location of obstacles in subsequent time steps.

In designing a real-world application for public deployment, all of the available problem-specific knowledge should be used. Nonetheless, the integration of learning into portions of the design of Dickmanns' system may reduce the amount of hand-coded knowledge which must be acquired and integrated. Further, using learning/adaptation in execution may increase the reliability of the system in unanticipated conditions, such as changing or adverse weather conditions or periods of sudden lighting change as may be encountered when entering a tunnel [Pomerleau, 1995].

6.3 Relations to Other Recurrent Neural Networks

The use of the feedback connections proposed are related to other recurrent neural networks [Jordan, 1989][Mozer, 1989]. The largest difference between these and the architecture used here is in the problems which these architectures are designed to address. Most of the recurrent networks which have been explored in the literature have attempted to address the problem of sequence recognition and reproduction. The architecture presented here is not suited to these tasks since the feedback is restricted to be the prediction of the next inputs, rather than the network's state. Another large difference is that in the networks used here, the feedback units are explicitly trained in a supervised manner. This is in contrast to architectures in which the context units are trained in a manner similar to the training of the hidden units – in which they are trained to reduce the error in a subsequent output layer.

7. SUMMARY

The main contribution of this paper is a method of creating task-specific expectations of subsequent inputs. The task-specific expectations can be used to attenuate inputs which do not match their expected values. In the task of lane-marker detection, this provided a means to exploit the task's temporal nature. The resulting system is robust to many forms of distracting features in the inputs. For lane marking detection, the algorithm is able to avoid being misdirected by extra lane markings, passing cars, and other potentially confusing features.

Unlike many of the previous studies which use neural networks to predict future states, this work has presented an algorithm which relies on the *limited* accuracy of the neural network's future state prediction. The premise of this algorithm is that future event prediction cannot be perfect. In particular, a network trained to perform a specific task cannot accurately predict the future inputs when the future inputs contain noise or when they contain features which are unrelated to the task. By analyzing the difference between what is expected in the next time step and what is actually present in the next time step, it is possible to estimate which of the features in the input retina are noise, and which are important to completing the particular task. Based upon the found features, revised expectations can be formed for subsequent time-steps.

8. ALTERNATIVE IMPLEMENTATIONS & FUTURE WORK

In this paper, we presented one method of using task-specific expectations – using them for filtering the next set of inputs. There are many alternate ways of creating and using these expectations. Five interesting options are described below.

In this study, the expectations were based on a linear combination of the activation of the hidden units. This is not a necessity for the mechanisms developed here; a hidden layer dedicated to the prediction task can be added. Also, if multiple hidden layers are necessary to solve the task, the prediction layer can be connected to all of the hidden layers. The crucial detail is to ensure that the errors which are propagated back from the prediction do not effect the representations which are developed in the hidden layers for solving the task.

During the initial phases of training, when the prediction is little more than random noise, filtering based on prediction can degrade input images. Once task-specific features are developed in the hidden units, more accurate predictions are made. One way to both reduce the training times and to potentially improve performance is to begin the feedback after some initial learning of the prediction has progressed. This will avoid the early stages of training when the inputs are filtered by essentially random feedback.

In this work, unpredicted features were de-emphasized from the inputs. After feeding the filtered inputs through the ANN, the resulting outputs were interpreted as the location of the lane marking. A direction for future research is to use expectations in conjunction with the original inputs to create a probabilistic output which specifies multiple potential locations of lane markings and their associated confidence values.

Instead of using the expectations of the next inputs for filtering, as was done in this study, they can be used for determining when large changes have occurred in the inputs. For example, there may be a sudden termination of the lane marking, or the road-type may change. The occurrence of these changes may be detectable by examining the magnitude of the difference between the expected input image and the actual input image. This is similar to the original use of IRRE [Pomerleau, 1994], but this method has the advantage of incorporating temporal information. By using expectations in this manner, images may trigger warning conditions because they are unexpected given the previous images seen.

A method to enhance the amount of information that is preserved from the previous steps is to not only feedback the prediction of the next state (as is currently done), but also to feedback the outputs and/or the hidden layers of the network from previous time steps. This can be implemented by standard recurrent networks. By allowing the network to remember its previous action, the correct steering direction can be determined when there is little or confusing information contained in the current input image.

9. REFERENCES

Ahmad, S. (1991), *VISIT: An Efficient Computation Model of Human Attention*. Ph.D. Thesis. University of Illinois at Urbana Champaign.

Allport, A. (1989) "Visual Attention", in: Posner, M., ed., *Foundations of Cognitive Science*, MIT Press, Cambridge, Ma., 631-683.

Baldi, P. & Hornik, K. (1989), "Neural Networks and Principal Components Analysis: Learning from Examples without Local Minima", *Neural Networks* 2 (1989) 53-58.

Baluja, S. (1996) *Expectation-Based Selective Attention.* Ph.D. Thesis. School of Computer Science, Computer Science Department. Carnegie Mellon University. October, 1996.

Baluja, S. & Maxion, R. (1996) "Artificial Neural Network Based Detection and Classification of Plasma-Etch Anomalies". To appear in: *The Journal of Intelligent Systems.*

Blake, A. & Yuille, A. (1992) (editors) *Active Vision*. MIT Press, Cambridge, MA.

Broadbent, D.E., (1982) "Task Combination and Selective Intake of Information", in: *Acta Psychologica*

50 253-290.

Clark, J. & Ferrier, N. (1992), “Attentive Visual Servoing”, in: *Active Vision.* A. Blake & A.Yuille, eds., (MIT Press, Cambridge, MA) 137-154.

Cottrell, G.W. & Munro, P. (1988) “Principal Component Analysis of Images via back-propagation.” *Proc. Soc. of Photo-Optical Instr. Eng.*, Cambridge, MA.

Dickmanns, E. “Expectation-based Dynamic Scene Understand” (1992), in (eds.) Blake & Yuille, *Active Vision*, MIT Press, Cambridge Massachusetts, pp. 303-334.

John, G.H., Kohavi, R., Pfleger, K. (1994), “Irrelevant Features and the Subset Selection Problem”, in: *Proceedings of the Eleventh International Conference on Machine Learning*, Morgan Kaufmann, San Mateo, CA. 121-129.

Jacobs, R.A., Jordan, M.I., Barto, G.A. (1990), “Task Decomposition Through Competition in a Modular Connectionist Architecture: The What and Where Vision Tasks”, COINS Technical Report 90-27.

Jordan, M.I, (1989) “Serial Order: A Parallel, Distributed Processing Approach”, in: J.L. Elman and D.E. Rumelhart, eds., *Advances in Connectionist Theory: Speech*, Hillsdale: Erlbaum.

Jordan, M.I, Jacobs, R.A. “Hierarchical Mixtures of Experts and the EM Algorithm”, *Neural Computation,* Vol.6, No. 2, March, 1994. 181-214.

Koch, C. & Ullman, S. (1985) “Shifts in Selective Visual Attention: Towards the Underlying Neural Circuitry”, in: *Human Neurobiology 4* (1985) 219-227.

Kramer, M. (1991) “Nonlinear Principal Component Analysis Using Autoassociative Neural Networks”, in: *AIChE Journal* 37:2, 233-242.

Mozer, M.C. (1989) “A Focused Back-Propagation Algorithm for Temporal Pattern Recognition”. *Complex Systems* 3, 349-381.

Pomerleau, D.A. (1993), *Neural Network Perception for Mobile Robot Guidance*, Kluwer Academic Publishing, Boston, Massachusetts.

Pomerleau, D.A. (1995), “RALPH: Rapidly Adapting Lateral Position Handler”, *Proceedings of the 1995 Symposium on Intelligent Vehicles*.

Swain, M. (1994) (editor) *International Journal of Computer Vision* - Special Issue on Active Vision II. Volume12, Numbers 2/3, April 1994.

Swain, M.J. & Stricker, M.A. (1993) “Promising Directions in Active Vision”, in *International Journal of Computer Vision - Special Issue on Active Vision I.* Volume 11, Number 2, October, 1993.

Thorpe, C., (1991) “Outdoor Visual Navigation for Autonomous Robots”, in: *Robotics and Autonomous Systems* 7, 85-98.

Umilta, C., (1988) “Orienting of Attention”, in: (eds.) Boller, F. & Grafman, J., *Handbook of Neuropsychology* V1. Elsevier, Amsterdam. 175-193.