Adaptive obstacle avoidance with a neural network for operant conditioning: experiments with real robots

P. Gaudiano and C. Chang

Boston University Neurobotics Lab
Dept. of Cognitive and Neural Systems, 677 Beacon Street, Boston, MA 02215 USA
E-mail: {gaudiano,cchang}@cns.bu.edu Web: http://neurobotics.bu.edu

Abstract

We have recently shown that a neural network model of classical and operant conditioning can be trained to control the movements of a wheeled mobile robot. The neural network learns to avoid obstacles as the robot moves around without supervision in a cluttered environment. The neural network does not require any knowledge about the quality or configuration of the sensors. In this article we report results using our neural network with the real mobile robot **Khepera**.

1 Introduction

Neural networks have been used in a number of robotic applications (e.g., [2,14]), including both manipulators and mobile robots. A typical approach is to use neural networks for nonlinear system modeling, including for instance the learning of forward and inverse models of a plant, noise cancellation, and other forms of nonlinear control. In most of these cases a "generic" neural network—typically a multi-layer perceptron trained with the backpropagation learning rule—is embedded within a traditional control scheme, and its primary function is to learn the parameters that lead to stable control. However, for this sort of application the neural network is being used as little more than a function approximator with good generalization properties. Furthermore, this sort of approach does not lend itself well to less constrained problems, such as navigation of a mobile robot in an unknown environment.

An alternative approach is to solve a particular problem by designing a specialized neural network architecture and/or learning rule. It is clear that biological brains, though exhibiting a certain degree of homogeneity, rely on many specialized circuits designed to solve particular problems. For instance, the neocortex of most mammals tends to be fairly uniform across sensory modalities, but the majority of subcortical processing is highly specialized for each modality.

We have begun to investigate the usefulness of a large collection of specialized neural networks developed over the last three decades by Grossberg and colleagues (see [3,10,11]). These neural network models are characterized by their focus on self-organization and unsupervised learning, and thus appear especially promising for applications in robotics. Our own work began with a neural network that learns without supervision the forward and inverse kinematics of a differential-drive mobile robot [7,16].

More recently we have developed neural networks that learn to navigate a mobile robot in an unknown environment, avoiding moving or stationary obstacles while approaching a target [6]. In this article we summarize the neural network architecture and present results using two different robotics platforms. We begin with a description of the psychological underpinnings of the model we use.

2 Classical and operant conditioning

When an animal has to operate in an unknown environment it must somehow learn to recognize informative cues in the environment, and to predict the consequences of its own actions. This learning is possible for organisms in spite of what seem like insurmountable difficulties from a standard engineering viewpoint: noisy sensors, unknown kinematics and dynamics, nonstationary statistics, and so on.

Psychologists have identified classical and operant conditioning as two primary forms of learning that enable animals to acquire the causal structure of their environment. In the classical conditioning paradigm, learning occurs by repeated association of a *conditioned stimulus* (CS), which normally has no particular significance for an animal, with an *unconditioned stimulus* (UCS), which has significance for an animal and always gives rise to an *unconditioned response* (UCR). For example, a rat that is repeatedly shocked (UCS) shortly after a red light is turned on (CS) will associate the red light with fear (UCR), meaning that eventually presentation of the red light alone elicits a *conditioned response* (CR) resembling the fear response elicited by the shock itself. Hence, classical conditioning refers to the act of learning to recognize informative stimuli in the environment.

In the case of operant conditioning, an animal learns the consequences of its actions. More specifically, the animal learns to exhibit more frequently a behavior that has led to a reward, and to exhibit less frequently a behavior that has led to punishment. For example, a pigeon can be trained to peck at an illuminated key in order to receive a small food reward.

In the field of neural network research, it is often suggested that neural networks based on associative learning laws can model the mechanisms for classical conditioning, while neural networks based on reinforcement learning laws can model the mechanisms for operant conditioning. However,

0-8186-8138-1/97 $10.00

both of these classes of models are too simple to function in realistic, unstructured environments. This is not to say that associative learning and reinforcement learning do not exist in some form or another in biological organism. Instead, the problem seems to lie in the use of rather simple, "monolithic" networks designed around each particular neural network law. At least two fundamental problems cannot be solved with these sorts of neural networks: first, the majority of neural networks function only as long as the inputs and outputs are controlled and timed carefully with respect to each other; second, most neural networks have no means for learning to discriminate "good" inputs from "bad" inputs on the basis of an internal value system.

The first of these problems has been aptly dubbed the *synchronization problem* by Grossberg [8,9]: how can learning between a CS and a UCS occur reliably even though they are presented at different times on different trials? The second problem can be discussed in the context of *motivation*, the internal force that produces actions on the basis of the momentary balance between our needs and the demands of our environment [5]: somehow humans and animals are able to learn the affective value of different stimuli, and learning is constrained to those cues and events that are affectively meaningful to them.

The ability to identify and discriminate what is good and what is bad is essential for an organism that must survive in an unstructured environment. In practice neural networks are never cast into the real world and left to fend for themselves, learning to recognize which things are good and which are bad. We are attempting to use neural networks for exactly this purpose.

3 Controlling a mobile robot through operant conditioning

In 1971, Grossberg proposed a model of classical and operant conditioning, which was designed to account for a variety of behavioral data on learning in vertebrates. The model was refined in several subsequent publications. In 1987, Grossberg & Levine described a computer simulation of a major component of the conditioning circuit. This model was used to explain a number of phenomena from classical conditioning. Our implementation of Grossberg's conditioning circuit, which follows closely that of Grossberg & Levine [12], is shown in fig. 1.

In this model the *sensory cues* (both CSs and UCS) are stored in Short Term Memory (STM) within the population labeled S, which includes competitive interactions to ensure that the most salient cues are contrast enhanced and stored in STM while less salient cues are suppressed. In the present model the CS nodes correspond to activation from the robot's range sensors. Note that there is no knowledge built into the network about (1) the kind of sensor information (e.g., infrared or sonar), or (2) the position of the sensor on the robot's body.

At the bottom of the figure the *drive node* D is represented, and conditioning can only occur when the drive node is active. The drive node in our model is active when the robot collides with an obstacle, which could be detected through a bump sensor, or when any one of the range sensors indicates that an obstacle is closer than the sensor's minimum range. The drive node and the nodes in the population labeled P, are represented as triangular nodes

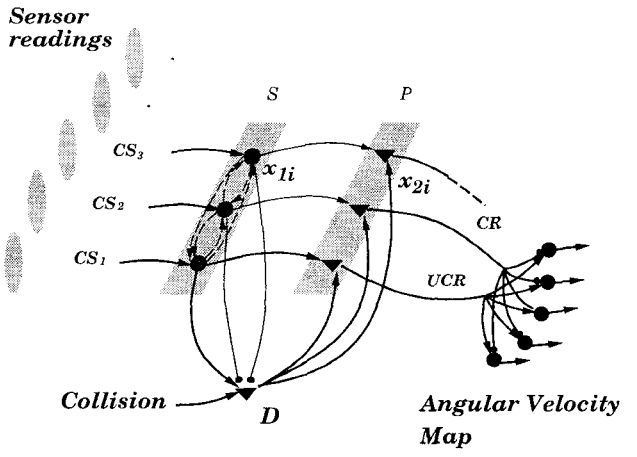


Figure 1: Conditioning model for obstacle avoidance. The range sensor activities represent the CSs. A collision detector activates the UCS. Motor learning occurs at a population coding the robot's target angular velocity. After conditioning, the pattern of activity across the range sensors can predict a collision and modify the robot's angular velocity to avoid the obstacle.

and are *polyvalent* cells, which means that they require the convergence of two types of input in order to become active.

Finally, the neurons at the far right of the figure represent the response (conditioned or unconditioned), and are thus connected to the motor system. In our model the motor population consists of nodes (i.e., neurons) encoding desired angular velocities, i.e., the activity of a given node corresponds to a particular desired angular velocity for the robot. For instance, the leftmost node corresponds to turning left at the maximum rate, the central node corresponds to straight ahead, and so on. When driving the robot, activation is distributed as a Gaussian centered on the desired angular velocity. The use of a Gaussian leads to smooth transitions in angular velocity even with few nodes, and it is also crucial for our obstacle avoidance scheme, as explained below.

The output of the angular velocity population is decomposed algorithmically into left and right wheel angular velocities. A gain term can be used to specify the maximum possible velocity. As an alternative, the transformation from the angular velocity population to actual wheel velocities can be done adaptively with another neural network [7].

3.1 The training cycle

In our model the range sensors initially do not propagate activity to the motor population because the initial weights are small or zero. The robot is trained by allowing it to make random movements in a cluttered environment. Specifically, we systematically activate each node in the angular velocity map for a short time, causing the robot to cover a certain distance and rotate through a certain angle depending on which node is activated. Whenever the robot collides with an obstacle during one of these movements (or comes very close to it), the nodes corresponding to the largest (closest) range sensor measurements just prior to the

collision will be active. Activation of the drive node allows two different kinds of learning to take place: the learning that couples sensory nodes (infrared or ultrasounds) with the drive node (the collision), and the learning of the angular velocity pattern that existed just before the collision.

The first type of learning follows an associative learning law with decay. This learning enables the most active sensory cues to accrue strength in their connection to the drive node (on the left side of fig. 1), so that eventually they will be able to generate their own drive signal, and thus activate the polyvalent cells P, and ultimately a motor response. The primary purpose of this learning scheme is to solve, at least to a degree, the *synchronization problem* described above, by ensuring that learning occurs only for those CS nodes that were active within some time window prior to the collision (UCS).

The second type of learning, which is also of an associative type but inhibitory in nature, is used to map the sensor activations to the angular velocity map (on the right side of fig. 1). By using an inhibitory learning law, the polyvalent cell corresponding to each sensory node learns to generate a pattern of inhibition that matches the activity profile active at the time of collision. For instance, if the robot was turning right and collided with an obstacle, the range sensor neuron most active shortly before the collision will learn to generate an inhibitory Gaussian centered upon the right-turn node in the angular velocity population.

In general, the nodes representing range sensors on the right side of the robot will be most active just prior to a collision when the robot is moving to the right, and likewise for other nodes. It is for this reason that there is no need to encode any information about the sensor geometry in the network: each node learns to suppress those movements that caused collisions with objects that the node "saw coming". In practice there are occasions in which the robot might collide, for example, on its left side even though it is turning to the right. However, on average this is a rare event: the gradual nature of learning and the use of a small decay rate jointly ensure that each node learns a pattern reflecting the statistical distribution of collisions foreseen by that node, which in turn reflects the position of the sensor on the robot's body.

3.2 Generating avoidance movements

Once learning has occurred, the activation of the angular velocity map is given by two components (fig. 2). An excitatory component, which is generated directly by the sensory system, reflects the angular velocity required to reach a given target in the absence of obstacles. We have shown previously how this signal could be derived from the sensors [7]; for simplicity here we assume that the angular velocity is proportional to the angle between the robot's heading and the target. The second, inhibitory component, generated by the conditioning model in response to sensed obstacles, moves the robot away from the obstacles as a result of the activation of sensory signals in the conditioning circuit.

One reason for using a Gaussian distribution of activity is depicted in fig. 2. When an excitatory Gaussian is combined with an inhibitory Gaussian at a slightly shifted position, the resulting net pattern of activity exhibits a maximum peak that is shifted in a direction *away* from the peak of the inhibitory Gaussian. Hence, the presence of an obstacle to the right causes the robot to shift to the left, and *vice versa*. Further details on this operation can be found elsewhere [6].

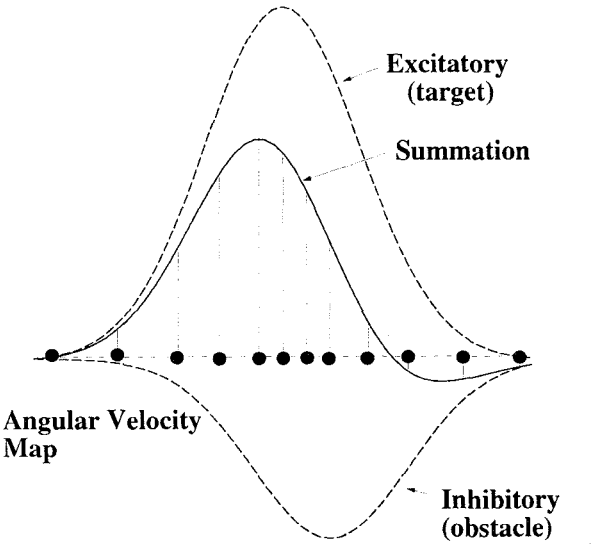


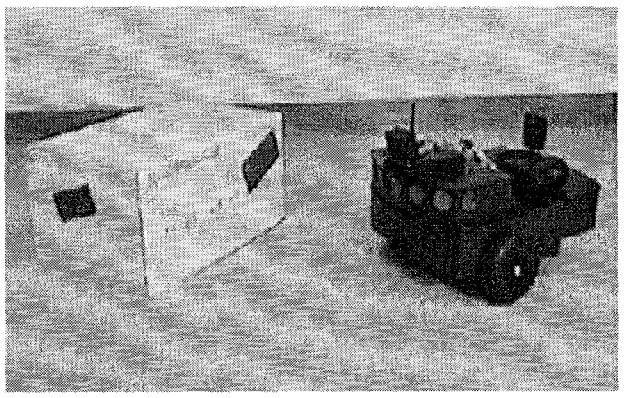
Figure 2: Positive Gaussian distribution represents the angular velocity without obstacles, and negative distribution represents activation from the conditioning circuit. The summation represents the angular velocity that will be used to drive the robot. Notice how the peak of the excitatory Gaussian is shifted by the inhibitory Gaussian.

4 Obstacle avoidance with real robots

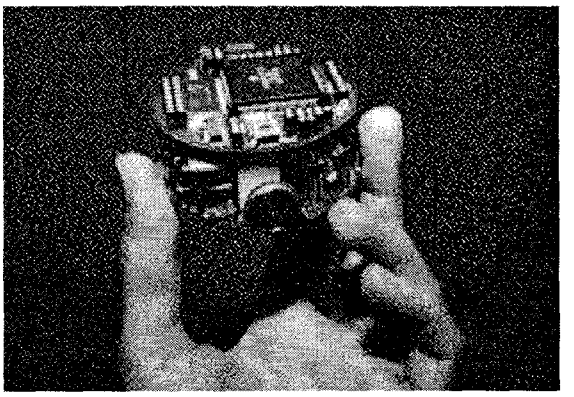
We have implemented the model just described on two real mobile robots: the Pioneer 1 (Real World Interface, Jaffrey, NH), shown in Fig. 3(**a**), is a small (14" wide, 18" long, 9" tall), two-wheel differential-drive robot with five forward-facing and two side-facing sonar range finders; the Khepera (K-team SA, Préverenges, Switzerland), shown in Fig. 3(**b**), is a miniature (2.2" diameter) differential-drive robot with eight infrared proximity sensors, six of which cover the frontal 180^{o}, and two facing backwards. In our experiments we have ignored the two rear-facing infrareds, using only the six frontal sensors, which cover the range [-90,+90]. We have previously reported our results using simulators, and also using the Pioneer 1 [4,6]. We focus here on the results using the Khepera.

It is worthwhile to note that the program requires essentially no modifications in order to run on these two robots, the only difference being that the infrared sensors on the Khepera return larger values for closer objects, while the sonars on the Pioneer return smaller values for closer objects, which required inversion to be represented as a CS (of course, there is also a difference in the number of CS nodes.)

An interesting difference between the two robots is that the sonars are accurate in the range of few inches to several feet, while the infrareds are only accurate over a range of a couple of inches. As one might expect, this led to somewhat different behaviors, with the Pioneer tending to perform gradual avoidance movements, and the Khepera performing abrupt, last-minute swerving movements.



(a)



(b)

Figure 3: (a) The Pioneer 1 robot; (b) the Khepera robot.

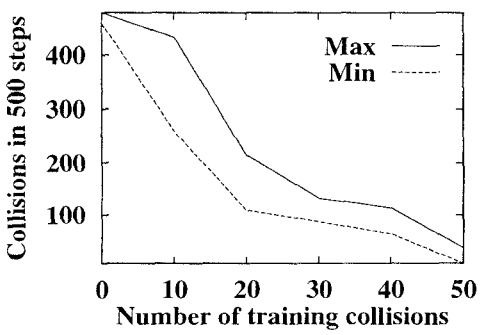


Figure 4: Learning curve for the Khepera robot, measured as the number of collisions in 500 steps as a function of the total number of collisions experienced during training. Min and Max refer to the best and worst learning curves out of a set of five training trials.

4.1 Training the Khepera

The goal of the training phase is to give each CS node the opportunity to sample several movements that lead to collisions. In practice we found that it is sufficient for each CS node to be active during only a handful of collisions. We have also found that with the Khepera it is possible to carry out the training using the *Khepera Simulator Package*,[1] and then transferring the weights to the program controlling the real robot. This has two main advantages: (1) systematic training of all nodes can be done more rapidly, and (2) we are not smashing our $2,500 miniature robot into the wall a few hundred times whenever we wish to train it. The Khepera simulator includes an accurate model of the real robot, and in fact we found that the results can be transferred directly to the real robot.

In order to generate a wide range of movements, during the training phase we turn on each node in the angular map for a brief time until a collision is registered, then switch to a new angular map node and repeat the process. We can achieve good avoidance behavior in this way with only a few collisions for each node. Fig. 4 illustrates the learning process. We obtained this curve in the following way: starting with all the weights in the network set to zero, we turn on one node in the angular map and let the robot collide with an obstacle, generating a small amount of learning, then turn on another node, and so on. At regular intervals during the training phase we temporarily disable learning and allow the robot to move from a new starting position for a total of 500 steps through the algorithm, and measure for how many of the 500 steps the robot detected a collision. On the first trial, before any learning has taken place, as soon as the robot collides it remains stuck against the obstacle, so the number of collisions is very close to 500. By the time we have trained through 50 collisions (total: meaning that each of the six sensors, on average, has sampled fewer than ten collisions) the robot is able to navigate with virtually no collisions.

Following this brief training phase on the simulator, the weights are used by a program that directly controls the real Khepera. We have let the Khepera run for very long times without detecting any collisions. The robot tends to move somewhat abruptly as it approaches a wall, though this depends in part on the robot's speed: as it begins to move at speeds of more than a few centimeters per second, the sensors almost become bump detectors, with activities going from zero to maximum within one or two time steps. This, however, is much more a characteristic of the robot and its sensors than of the scheme we have used.

One reason for the abruptness of the robot's movements is that as soon as one sensor is close enough to a wall, it will cause the robot to start rotating by a relatively small amount. However, as the robot rotates nearly in place, the next sensor will also see the wall, causing an even further turn, and so on. It is not unusual, depending on the speed and angle of approach, to see the Khepera turn through angles of 90° or more. There are various ways of controlling the smoothness of the avoidance movements. One possiblity is to redefine a collision as requiring simultaneous maximal activation of *two* adjacent sensors. This solution in fact generates significantly smoother movements, but presents to problems: first, this definition of collision pre-

[1] Written at the University of Nice Sophia-Antipolis by Olivier Michel, downloadable from the World Wide Web at `http://wwwi3s.unice.fr/~om/khep-sim.html`

sumes knowledge about which sensors are adjacent to each other, which we would rather avoid (though a simple coincidence detector could be easily implemented neurally). Second, the robot tends to lose its ability to "see" approaching corners or narrow obstacles that only activate a single sensor.

Another possibility would be to impose some gradual dynamics on the rapidity with which the robot can rotate. We have successfully implemented a similar scheme on the Pioneer, but as one might expect there is a marked trade-off between how quickly the robot can turn and how successful it will be at avoiding the obstacle. With the short range of the Khepera's infrared sensors, it is especially important to allow for rapid avoidance movements.

5 Discussion

We have presented some results using a neural network model of classical and operant conditioning to learn obstacle avoidance in real mobile robots. The model is able to learn rapidly and efficiently to avoid obstacles without the need for supervision, and without any knowledge of the nature or geometrical configuration of its sensors, as demonstrated by our ability to use the model on two very different robotics platforms. The ability to work with real sensors in a real world is a testament to the model's robustness. We would like to suggest that the success of our work is due largely to the validity of the model we used, a model that was the result of decades of research on neural and behavioral aspects of animal learning.

There are many ways in which the model could be extended and improved. In fact, the implementation of Grossberg & Levine [12], which we followed fairly closely, is known to have some problems. Many of the problems, however, arise from the fact that this is only one piece of a significantly more complex neuropsychological theory [8,9]. One of the first ingredients that we would like to add is multiple drive nodes representing different, or even opposite affective states, and a richer array of sensor nodes. For instance, rather than relying solely on range sensors and on collision detector, we are planning to add light sensors and a *reward* node that is activated by the proximity of a bright light source. Such an extension requires a change to the neural network to accommodate the interactions between opponent drives, as implemented for instance in [13]. We have begun to design the new network, and plan to use the Khepera's infrared sensors, which can also double as ambient light detectors.

We are not the first to suggest that models of animal learning are of potential use in robots, nor are we the first to use some of Grossberg's models for this purpose. The most impressive achievements along this front have been reported by Baloch & Waxman [1]. Their work, which was done on the robot MAVIN, includes a much richer and more complex model than the one presented here, and its focus is on visual navigation. However, they too used a variant of Grossberg's conditioning circuit as part of their overall control scheme. The main difference is perhaps in our approach, which is striving to achieve rapid adaptive control that is independent of the platform on which the neural network is being used. Nonetheless, the work of Baloch & Waxman [1] is very impressive and we look forward to extending our own models to include some of the capabilites demonstrated for MAVIN.

It is also useful to compare our approach to results published recently by Pfeifer & Verschure [15], which also utilizes a form of associative learning to learn avoidance behavior (and also approach but under special conditions). One main difference is that their model presumes the existence of an avoidance behavior, the success of which depends on built-in knowledge about the sensors. For instance, the right sensor nodes learn to generate a behavior of "turn left," and so on. In contrast, our approach obviates the need for this sort of knowledge by adding the inhibitory "operant" learning. Furthermore, because it learns to associate sensor activities specifically with the built-in avoidance reflexes, the model of Pfeifer and Verschure is not learning new behaviors, but simply learning to generate built-in behaviors at the right time. This is somewhat different in our case, where the robot begins only with the ability to move forward at one of several angles. Notice that if learning were totally removed, the model of Pfeifer and Verschure would still be able to avoid collisions through its built-in behaviors, whereas our robot would continue to collide and get stuck until it randomly rotated by an amount that set it free (until the next collision). In any event, the similarities between our models are striking, and a combined approach might be very interesting to pursue.

Acknowledgments

Our work is supported the mediaCenter GmbH in Friedrichshafen, Germany, and by grants from the Office of Naval Research (N00014-96-1-0772 and N00014-95-1-0409).

References

[1] A. Baloch and A. Waxman. Visual learning, adaptive expectations, and behavioral conditioning of the mobile robot MAVIN. *Neural Networks*, 4:271–302, 1991.

[2] G. A. Bekey and K. Y. Goldberg, editors. *Neural Networks in robotics*. Kluwer, Boston, MA, 1993.

[3] G. A. Carpenter and S. Grossberg, editors. *Pattern Recognition by Self-Organizing Neural Networks*. MIT Press, Cambridge, MA, 1991.

[4] C. Chang and P. Gaudiano. Neural competitive maps for reactive and adaptive navigation. In *Proceedings of the 2nd International Conference on Computational Intelligence and Neuroscience*, 1997. March 2, 1997, Research Triangle Park, NC.

[5] C. Dorman and P. Gaudiano. Motivation. In M. Arbib, editor, *Handbook of brain theory and neural networks*. MIT Press, Cambridge, MA, 1994. In press.

[6] P. Gaudiano, E. Zalama, C. Chang, and J. López Coronado. A model of operant conditioning for adaptive obstacle avoidance. In P. Maes, M. J. Mataric, J. Meyer, J. Pollack, and S. W. Wilson, editors, *From Animals to Animats 4*, pages 373–381, Cambridge, MA, 1996. MIT Press.

[7] P. Gaudiano, E. Zalama, and J. López Coronado. An unsupervised neural network for real-time, low-level control of a mobile robot: noise resistance, stability, and hardware implementation. *IEEE SMC*, 26:485–496, 1996.

[8] S. Grossberg. On the dynamics of operant conditioning. *Journal of Theoretical Biology*, 33:225–255, 1971.

[9] S. Grossberg. A psychophysiological theory of reinforcement, drive, motivation and attention. *Journal of Theoretical Neurobiology*, 1:286–369, 1982.

[10] S. Grossberg, editor. *Studies of Mind and Brain: neural principles of learning, perception, development, cognition and motor control.* Reidel, Boston, 1982.

[11] S. Grossberg, editor. *Neural Networks and Natural Intelligence.* MIT Press, Cambridge, MA, 1989.

[12] S. Grossberg and D. Levine. Neural dynamics of attentionally modulated Pavlovian conditioning: blocking, interstimulus interval, and secondary reinforcement. *Applied Optics*, 26:5015–5030, 1987.

[13] S. Grossberg and N. A. Schmajuk. A neural network architecture for attentionally-modulated Pavlovian conditioning: Conditioned reinforcement, inhibition, and opponent processing. *Psychobiology*, 15:195–240, 1987.

[14] W. T. Miller, R. S. Sutton, and P. J. Werbos, editors. *Neural networks for control.* M.I.T. Press, Cambridge, MA, 1990.

[15] R. Pfeifer and P. Verschure. Distributed adaptive control: a paradigm for designing autonomous agents. In F. J. Varela and P. Bourgine, editors, *Toward a practice of autonomous systems*, pages 21–30. MIT Press, Cambridge, MA, 1992.

[16] E. Zalama, P. Gaudiano, and J. López-Coronado. A real-time, unsupervised neural network model for the low-level control of a mobile robot in a nonstationary environment. *Neural Networks*, 8(TR-94-002):103–123, 1995.