

1994-01

# A real-time unsupervised neural network for the low-level control of a mobile robot in a nonstationary environment

---

<https://hdl.handle.net/2144/2140>

*"Downloaded from OpenBU. Boston University's institutional repository."*

**A REAL-TIME, UNSUPERVISED NEURAL NETWORK FOR THE LOW-LEVEL  
CONTROL OF A MOBILE ROBOT IN A NONSTATIONARY ENVIRONMENT**

Eduardo Zalama, Paolo Gaudiano, and Juan Lopez Coronado

**January 1994**

**Revised: July 1994**

**Technical Report CAS/CNS-94-002**

*Neural Networks*, in press

Permission to copy without fee all or part of this material is granted provided that: 1. the copies are not made or distributed for direct commercial advantage; 2. the report title, author, document number, and release date appear, and notice is given that copying is by permission of the BOSTON UNIVERSITY CENTER FOR ADAPTIVE SYSTEMS AND DEPARTMENT OF COGNITIVE AND NEURAL SYSTEMS. To copy otherwise, or to republish, requires a fee and/or special permission.

Copyright © 1994

Boston University Center for Adaptive Systems and  
Department of Cognitive and Neural Systems  
111 Cummington Street  
Boston, MA 02215

# A real-time, unsupervised neural network for the low-level control of a mobile robot in a nonstationary environment

Eduardo Zalama, Paolo Gaudiano, Juan López Coronado

**Affiliations:** E. Zalama and J. López Coronado are at the Universidad de Valladolid, Dept. de Ingeniería de Sistemas y Automática. P. Gaudiano is at Boston University, Dept. of Cognitive and Neural Systems.

**Acknowledgments:** We would like to thank the neural network group of the Systems and Control Engineering Department of the ETSII of the University of Valladolid for their valuable contribution to the realization of this work: J.M. Cano, F.J. Díaz, Y.A. Dimitriadis, L.J. de Miguel, C.G. Moreno, A. Muñoz, J. Pérez, I. Villanueva and S. M. Jennings. We also wish to thank D. Bullock and an anonymous reviewer for useful comments. P. Gaudiano is supported by a Sloan Fellowship (BR-3122), and AFOSR:F49620-92-J-0499. He was supported by a Visiting Professor Fellowship while at the University of Valladolid.

**Corresponding Author:** Address all reprint requests to Paolo Gaudiano, Boston University, Dept. of Cognitive and Neural Systems, 111 Cummington Street, Boston, MA 02215, USA.

**Running title:** Unsupervised robot controller

**Keywords:** associative learning, DIRECT model, inverse kinematics, odometric mapping, VAM learning, VITE model

*Neural Networks*, in press

## Abstract

This article introduces a real-time, unsupervised neural network that learns to control a two-degree-of-freedom mobile robot in a nonstationary environment. The neural controller, which is termed *neural NETWORK MOBILE Robot Controller* (NETMORC), combines associative learning and Vector Associative Map (VAM) learning to generate transformations between spatial and velocity coordinates. As a result, the controller learns the wheel velocities required to reach a target at an arbitrary distance and angle. The transformations are learned during an unsupervised training phase, during which the robot moves as a result of randomly selected wheel velocities. The robot learns the relationship between these velocities and the resulting incremental movements. Aside from being able to reach stationary or moving targets, the NETMORC structure also enables the robot to perform successfully in spite of disturbances in the environment, such as wheel slippage, or changes in the robot's plant, including changes in wheel radius, changes in inter-wheel distance, or changes in the internal time step of the system. Finally, the controller is extended to include a module that learns an internal odometric transformation, allowing the robot to reach targets when visual input is sporadic or unreliable.

# 1 Introduction

This article introduces a real-time, unsupervised neural network controller that can learn to guide a mobile robot towards a target located at an arbitrary location in a 2-D workspace. The robot's movements are controlled by selecting the angular velocity of each of two driving wheels. The neural network controller must compensate in real-time for unexpected miscalibrations, such as changes in the robot's plant, and changes in the environment, such as moving targets or slippery floors. The focus of this article is on low-level control. Specifically, we are concerned with the control of simple movements toward stationary and moving targets. The controller we propose is based on existing neural network models of biological sensory-motor control. The controller requires no information about the robot's structure, and it is resistant to a variety of disturbances.

The proposed neural network controller combines Vector Associative Map (VAM) learning (Gaudiano & Grossberg, 1991) and associative learning within an architecture similar to the DIRECT model of arm trajectory generation (Bullock, Grossberg, & Guenther, 1993). The controller in the present application learns a transformation between spatial and velocity coordinates. Learning takes place through a *circular reaction* (Piaget, 1963), whereby the neural network learns to control the robot through a sequence of spontaneously generated random movements. The random movements enable the neural network to learn the relationship between angular velocities applied at the wheels and the incremental displacement that ensues during a fixed time step.

Because of the nature of the VAM/DIRECT architecture, which in this instance continuously calculates a vectorial difference between desired and actual velocities, the robot can move to arbitrary distances and angles even though during the initial training phase it has only sampled a small range of displacements. Furthermore, the on-line error-correcting properties of the proposed architecture endow the controller with many useful properties, such as the ability to reach targets in spite of drastic changes in the robot's plant or other perturbations.

As with the DIRECT model, this neural network controller includes a module that learns the inverse transformation—from velocity to spatial coordinates—which can be used as proprioceptive information to guide the robot when visual information is sporadic or unreliable.

The article begins with a description of the problem, a brief overview of the VAM and DIRECT models of arm trajectory generation, and a description of the proposed architecture. Next, comparison of analytical and numerical solutions shows that the controller learns the correct transformation required for guiding the robot toward arbitrary targets. A series of simulations illustrate the controller's ability to track stationary or moving targets under normal or perturbed conditions. The controller is also tested on trajectory-tracking problems of the type used to test more classical control algorithms. The article then presents simulations showing adaptation to statistically significant perturbations. A final result illustrates how the robot can perform movements using learned information about the robot's structure that can complement or supplant visual information. The article concludes with a discussion of related models, and an outline of future work.

## 2 The control of mobile robots

The control of mobile robots by means of sensory information is a fundamental problem in robotics. The autonomous, unsupervised control of a mobile robot in a novel environment is particularly challenging as it requires a degree of flexibility and self-organization that is difficult to achieve with approaches from classical control theory. Several researchers have attacked some of the problems in mobile robot control using techniques from engineering and artificial intelligence

(Arkin, 1990; Baloch & Waxman, 1991; Brooks, 1992; Jones & Flynn, 1993; Kaelbling, 1992; Nagata, Sekiguchi, & Asakawa, 1990; Pfeifer & Verschure, 1992). Problems that have been researched vary from sensor fusion to path planning and intelligent navigation. In this article we focus instead on low-level control, that is, aspects of mobile robot control that involve simple movements to stationary or moving targets. We will show that an adaptive neural network architecture can carry out simple reaching movements robustly and efficiently.

In recent years neural networks have emerged as possible tools for control (*e.g.*, Antsaklis, 1990; Bekey & Goldberg, 1993; Miller, Sutton, & Werbos, 1990a; Widrow & Stearns, 1985). Neural networks have been employed for several types of control, including for example simple supervised control (Albus, 1975; Kraft & Campagna, 1990; Miller, Hewes, Glanz, & Kraft, 1990b), system identification (Goodwin & Sin, 1984; Widrow & Stearns, 1985), and inverse system identification (Kawato, Fukukawa, & Suzuki, 1987; Kuperstein & Rubenstein, 1989; Miyamoto, Kawato, Setoyama, & Suzuki, 1988). In the realm of robot control, several authors have utilized standard error-correcting neural networks such as backpropagation and LMS to train robots to reach targets or follow prescribed trajectories (Bekey & Goldberg, 1993; Dedieu & Mazer, 1992; Hashimoto, Kubota, Sato, & Harashima, 1992; Nagata et al., 1990; Nguyen & Widrow, 1990; Pomerleau, 1991; Varela & Bourguine, 1992; Wada & Kawato, 1993; Wilson & Rock, 1993). However, many of these models suffer from the requirement of supervision during the training phase. Furthermore, supervised neural networks rely on the user's knowledge to ensure that the environment encountered during learning is statistically representative of the environment encountered during normal operation. Hence most of the proposed supervised neural networks operate inaccurately in novel environments, or when the conditions of the problem are altered.

The appeal of neural networks comes partly from the observation that mother nature appears to have solved the problem of unsupervised control of movement in novel environments; the ability to move through a novel environment is crucial to the survival of all animals. Whatever solution has been devised by natural organisms must be extremely flexible and robust, as it applies to motor systems as diverse as walking (bipedal or quadrupedal), flying, and swimming. Supervised, error-based neural networks appear to lack the fundamental ingredients that lead to the robust and flexible control found in humans and animals.

In recent years several neural network models have been developed for the unsupervised control of movement in humans and animals, especially with respect to eye movements (Grossberg & Kuperstein, 1989) and arm movements (Bullock & Grossberg, 1988; Gaudio & Grossberg, 1991; Bullock et al., 1993). One noteworthy characteristic of these neural network models is their closeness to known neural structures. In many cases, these models have been shown to generate movements whose characteristics match accurately those of movements measured in humans and animals (Bullock & Grossberg, 1988; Bullock et al., 1993). Some of these models have also been shown to be effective in robotics applications (Kuperstein, 1991; Kuperstein & Kottas, 1993).

The aim of this article is to apply a combination of the VAM (Gaudio & Grossberg, 1991) and DIRECT (Bullock et al., 1993) models for the control of visually-guided arm movements to the control of a mobile robot. Specifically, we propose a low-level adaptive neural controller that enables the robot to reach targets registered via sensors. The controller does not require knowledge of the robot's kinematics, it is robust to noise and various forms of perturbations, and it is able to carry out low-level control tasks such as reaching a stationary or moving target. We begin with a brief review of the main characteristics of the VAM and DIRECT models from which our neural controller is derived.

## 2.1 The VAM model

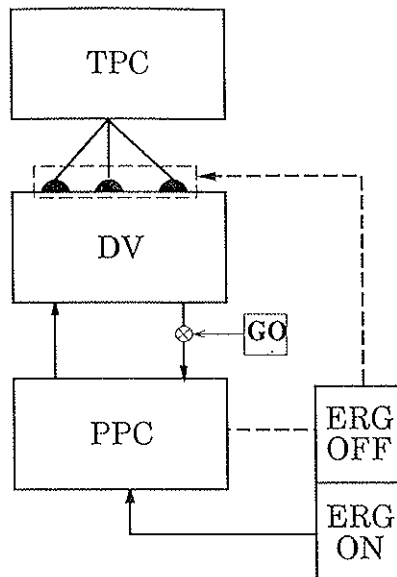


Figure 1: Schematic diagram of the Adaptive VITE model of Gaudiano and Grossberg (1991). Nodes in the TPC population represent the target position of the arm in motor or spatial coordinates. Nodes in the PPC population generate outflow muscle commands, and thus represent an internal copy of the arm's present position. Nodes in the DV population calculate a vectorial difference between PPC and TPC, thus representing discrepancy between present and target positions. ERG ON/OFF activation generates alternating active and quiescent phases during initial training. Modifiable synapses (semicircles within dashed box) carry out a transformation from TPC coordinates to PPC coordinates by means of the VAM learning law. Adapted from Gaudiano and Grossberg (1991) by permission.

The VAM (Gaudiano & Grossberg, 1991) is a neural network model for unsupervised, real-time, error-based learning of intermodal or intramodal mappings, which was initially developed as a neural model of adaptive generation and control of arm movements in humans and animals. Gaudiano and Grossberg (1991) have shown how, using an initially untrained VAM architecture, spontaneous random movements can be used to self-organize associative maps for the execution of visually guided voluntary arm movements. The VAM architecture was first proposed as an adaptive version of the Vector Integration To Endpoint (VITE) model of arm trajectory generation (Bullock & Grossberg, 1988). In the VITE model (Fig. 1), the Target Position Command (TPC) population represents a target joint configuration in muscle coordinates, that is, the TPC contains pairs of cells whose activations represent the agonist-antagonist muscle commands corresponding to a desired joint angle; the Present Position Command (PPC) population generates outflow movement commands that contract the muscles, and thus represents an internal measure of the present joint angle; the Difference Vector (DV) population continuously calculates the discrepancy between target and present joint configurations.

During normal operation in the calibrated VITE model, the PPC integrates nonzero DV activation, thus changing the joint angle. This integration continues until the PPC becomes equal to the TPC, at which time DV activation equals zero and no further PPC integration takes place. The GO signal multiplicatively gates the DV-PPC integration loop so as to control the velocity of integration, and thus the velocity of the resulting arm trajectory. When the GO signal is zero, a movement can be "primed" without actually taking place. Hence the GO signal encodes both desired velocity and the intent to move.

The VITE model is based on the assumption that the TPC and PPC populations encode information in identical coordinate systems, so that a meaningful difference can be calculated by

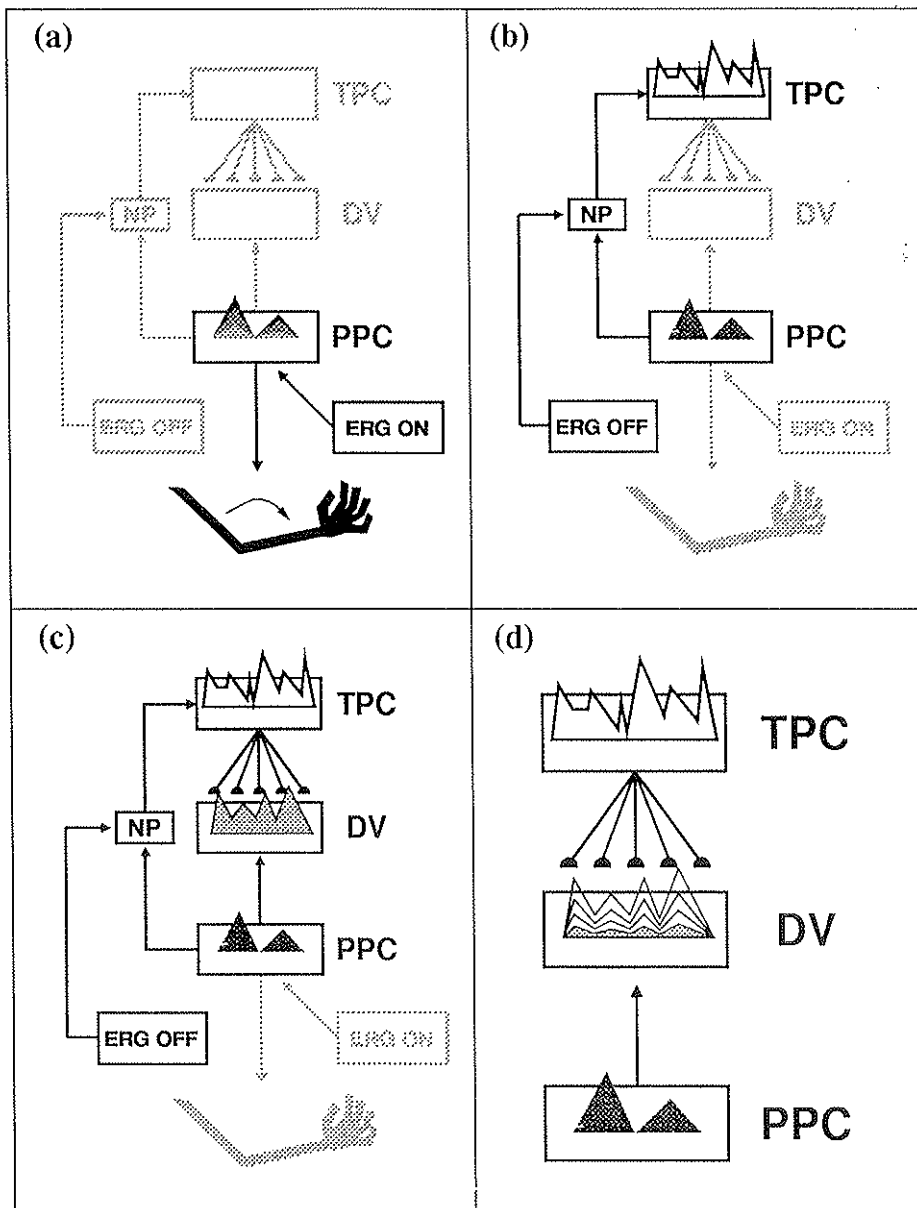


Figure 2: Illustration of a typical babble phase in the VAM. (a): The ERG ON generates random PPC activation, moving the arm. (b): After a short time activation is switched to the ERG OFF, signaling that the PPC activation is constant. ERG OFF activation opens the NP gate, copying PPC to TPC via an arbitrary transformation. (c): Any nonzero DV activity represents internal miscalibration. (d): The VAM learning law modifies TPC→DV synapses by reducing DV activation to zero. Reprinted from Gaudiano and Grossberg (1991) by permission.

the DV. The VAM learning law enables the VITE model to self-organize the correct transformation from TPC to DV when the TPC and PPC coordinate systems differ. Gaudiano and Grossberg (1991) proposed that learning in the VAM is the result of an internal measure of error that is registered at the DV during random, involuntary movements. During the initial part of the VAM's training phase, random movements are generated periodically at the PPC by an Endogenous Random Generator (ERG), as shown in Fig. 2a. The ERG is a module that contains two opposing channels: the ERG ON channel generates a short, random burst of activity that activates the PPC and causes a random arm movement; after a short time the ERG ON becomes inactive, and the ERG OFF generates a steady output signaling that the arm is stationary (Gaudiano & Grossberg, 1991). Activation of the ERG OFF channel allows the PPC to be copied to the TPC, either through an internal prewired transformation, or through an external feedback loop involving for instance the visual system (Fig. 2b). This operation forces the TPC to represent the same target as the PPC. If the system were properly calibrated, the DV should be zero. Consequently, any nonzero activity at the DV represents an internal measure of miscalibration (Fig. 2c). The VAM learning law changes the weights from the TPC to the DV in such a way that the DV is decreased to zero when the TPC and PPC are known to be the same (Fig. 2d), a condition that is controlled autonomously by the OFF phase of the ERG.

In summary, the DV performs complementary operations during the performance and learning phases. During performance, nonzero DV activation represents a discrepancy between the desired (TPC) and actual (PPC) arm configuration; integration at the PPC drives the DV to zero while moving the arm to its desired target. During learning, whenever the TPC and PPC are forced to be equal, a nonzero DV represents an internal measure of miscalibration; the VAM learning law changes the connections from the TPC to the DV so as to reduce the DV to zero, thus learning the correct calibration. It should be noted that although we refer to learning and performance phases as if they were distinct phases, in reality there is no need for these phases to be separated. In fact, Gaudiano and Grossberg (1991) have shown that the VAM learning will stabilize automatically once a correct calibration is achieved, and that learning will automatically resume in the event that the system's plant changes, for example as a result of growth or injury.

The VAM architecture has been shown to have the potential for broad applicability. For instance, the VAM architecture can be used to learn transformations between populations having similar or different coordinate systems. In an *intramodal* VAM, both the TPC and PPC are in joint coordinates, and learning assures that the amplitude coding at the TPC is congruent with the amplitude coding at the PPC. In an *intermodal* VAM, the TPC is in *spatial* coordinates (that is, the location of the active node within a population, rather than the activation level, encodes the location of a target in space), while the PPC is in joint coordinates. Because TPC and PPC neurons perform homologous functions, intra- and intermodal VAMs can be joined in a VAM *cascade* (Fig. 3), whereby the internal calibration of the VITE model and the calibration between spatial and motor coordinates are carried out in successive steps. The VAM cascade exhibits several useful properties, including the ability to learn invariant transformations when the spatial TPC depends jointly upon multiple systems, such as the position of the target on the retina, and the position of the eyes in the head (Gaudiano & Grossberg, 1991; Grossberg, Guenther, Bullock, & Greve, 1992). More detailed descriptions of the VAM and ERG models can be found elsewhere (Gaudiano & Grossberg, 1990, 1991).

### 2.1.1 The DIRECT model

In their recent work, Bullock et al. (1993) have extended the ideas of the VITE and VAM models to the situation in which the manipulator has more degrees of freedom than the space in which

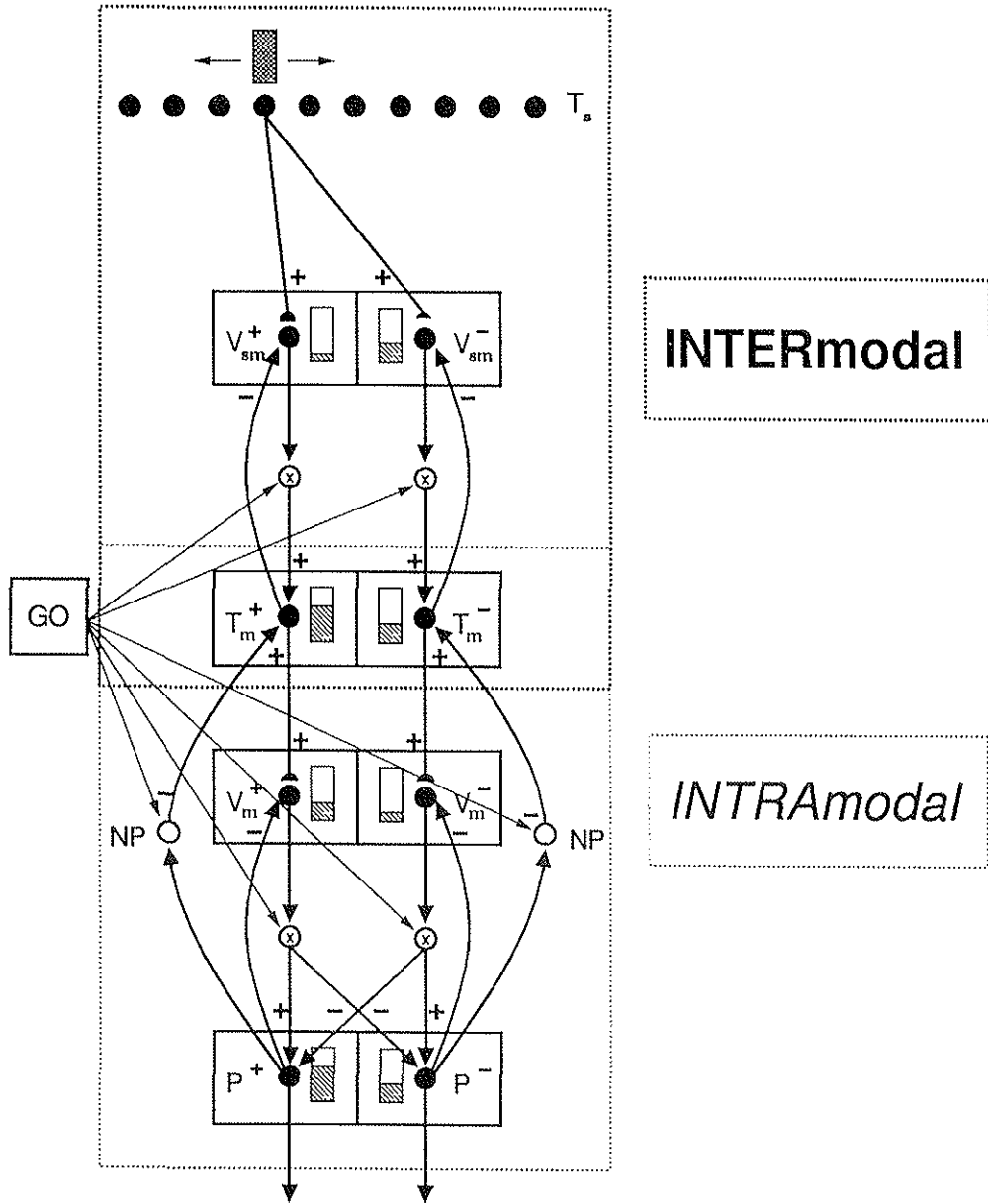


Figure 3: Diagram of a VAM cascade, combining intramodal (bottom) and intermodal (top) VAM modules. Adapted from Gaudio and Grossberg (1991) by permission.

it operates, as is the case, *e.g.*, for arm movements in humans. When this is the case, the inverse kinematic problem is ill-posed, that is, given a desired target in space it is possible to find many joint configurations for which the arm reaches the desired target. Bullock et al. (1993) propose that in this case the control of arm movements can be done more flexibly and accurately if the spatial target is used to generate a continuous sequence of desired *directions* of movement, which leads to appropriate changes in joint angles that move the arm in the desired spatial direction. Bullock et al. (1993) introduce the DIRECT model, in which spatial directions are transformed into joint rotations through a *direction-to-rotation transformation*. The acronym DIRECT stands for DIrection-to-Rotation Effector Control Transform, and it is a mnemonic device for the model's use of directions, rather than positions, to generate movement trajectories.

The use of spatial trajectory formation with directions leads to flexible arm control that can automatically compensate for several types of unexpected perturbations, including the use of a tool of arbitrary length at the end of the arm, blocking of one joint, or a systematic displacement of visual images through the use of prisms (Bullock et al., 1993).

In addition, the DIRECT model includes maps capable of learning the inverse mapping from joint angle increments to spatial coordinates of the end effector. This information can be used within an internal feedback loop to complement or supplant information from the visual system. The use of internal, or *proprioceptive* information about effector position allows accurate movements even when visual information is sporadic. This property of the DIRECT model can be very useful in robotics applications, where typically visual feedback is slow relative to the time scale of robot movements.

Fig. 4 illustrates the main components of the DIRECT model. The modules in the right column comprise a sequence of processing stages that enable visual targets to generate changes in joint angles, moving the end effector to its target. Beginning from the top of the figure, the first population (within the dashed box) calculates the location of the target in a 3-D, body-centered coordinate system, using for instance the models described by Greve, Grossberg, Guenther, and Bullock (1992), Grossberg et al. (1992), Guenther, Bullock, and Greve (1993); the next population calculates the spatial DV between the target and a 3-D representation of the end effector; the third population carries out a transformation from a spatial DV to the corresponding joint rotations (direction-to-rotation transformation); the direction-to-rotation transformation requires information about the present configuration of the arm, which is encoded in the final stage of the DIRECT model: output from this stage causes changes in joint angles and thus in the position of the end effector.

The left side of Fig. 4 illustrates the modules that process information about end-effector position that is used to update the spatial DV. Beginning at the top, the visual system can be used to calculate the 3-D body-centered position of the end effector, using the same circuitry that calculates the visual target position; the population at the bottom learns to transform the outflow movement commands into a correct spatial representation of the end effector position (this is the population that calculates the proprioceptive mapping mentioned above); the visual and proprioceptive information about end effector position are combined at the middle stage, which uses its multimodal inputs to generate the 3-D body-centered representation of the end effector that is used in the computation of the spatial DV. This multimodal stage is designed so as to allow visual information to dominate when it is available, and proprioceptive information otherwise.

In order to learn all of the necessary transformations, the DIRECT model relies on a training phase similar to that of the VAM, whereby random arm movements are generated, and the resulting movements are the basis of the internal calibrations. The reader interested in further details on the DIRECT model can consult the original article (Bullock et al., 1993).

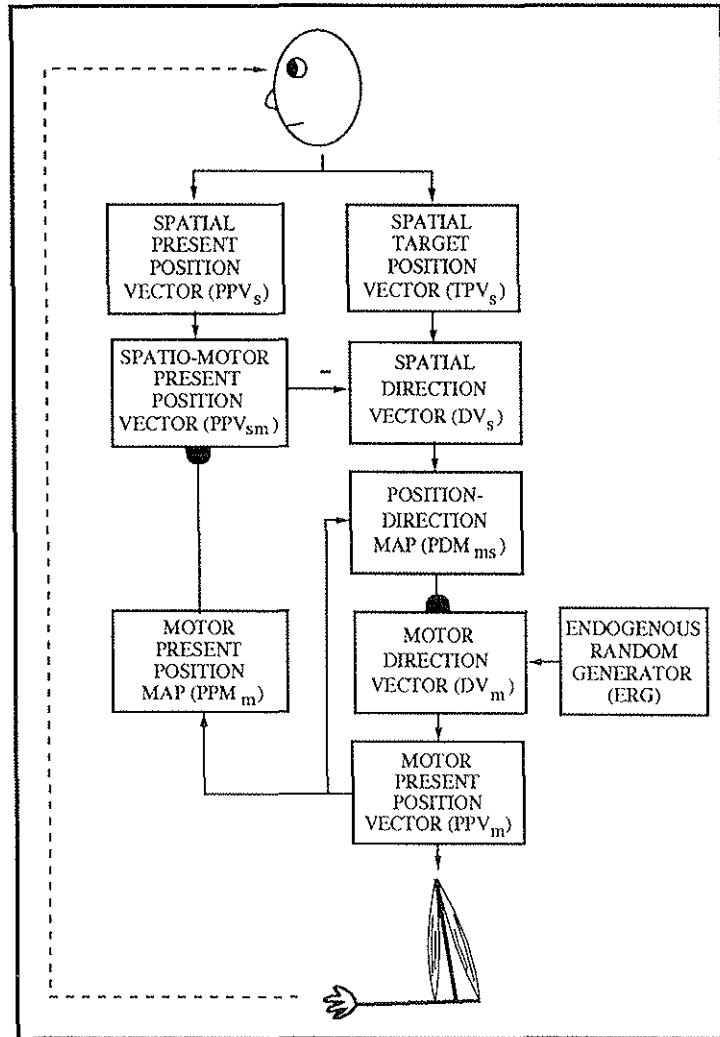


Figure 4: Diagram of the DIRECT model of Bullock et al. (1993). Modules in the right column enable the DIRECT model to move the arm to arbitrary targets in 3-D space. Modules in the left column enable the DIRECT model to learn the effect of its movements, thus allowing for blind reaching. Reprinted from (Bullock et al., 1993) by permission.

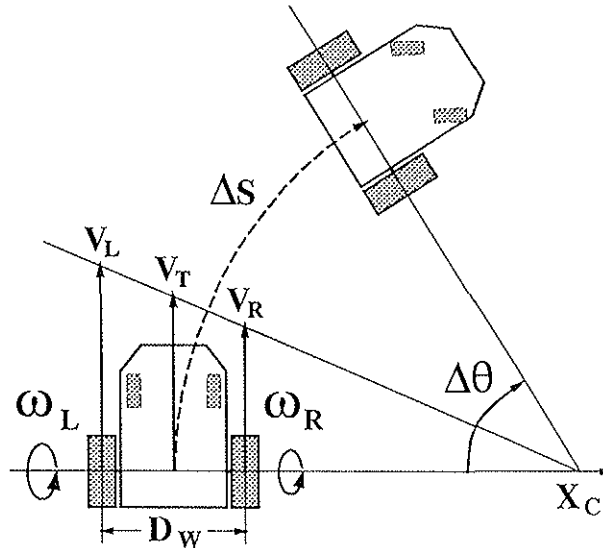


Figure 5: Illustration of the mobile robot. Angular velocities  $\omega_L$  and  $\omega_R$  are applied to robot's left and right rear wheels, respectively. One or two front wheels swivel freely. During a short time interval  $\Delta t$ , the robot moves along a circular path with center of rotation  $X_c$ , through a distance  $\Delta S$  and angle  $\Delta\theta$ . The relationship between angular velocities and the resulting movement depends on the robot's geometry.

## 2.2 Applying the VAM and DIRECT models to a mobile robot

The VAM and DIRECT models have been described in the context of generation and control of arm movement trajectories. However, the models possess a constellation of properties that can be equally useful in other types of control, including the control of a mobile robot. The VAM learning law is very stable, and does not require separate learning and performance phases. Similarly, learning is controlled internally, and can resume if the system becomes miscalibrated during performance. The DIRECT model suggests that incremental, rather than absolute coordinates, may be used for the control of movement towards a target. As mentioned above, the DIRECT model's use of an inverse mapping for proprioceptive information can compensate for a slow or sporadic visual system. Finally, the error-correcting nature of both the VAM and DIRECT models results in a robust type of control that ensures convergence to target.

We now introduce a neural network architecture that combines ideas from the VAM and DIRECT models in order to perform adaptive control of a mobile robot. We begin with a description of the robot system, and a discussion of the problems that arise in controlling this type of robot. We will then describe the architecture, and show several simulation results.

## 3 Model Description

### 3.1 Overview of the model

As diagrammed in Fig. 5, the robot is controlled by selecting a desired angular velocity for each of two rear wheels. One or two additional wheels in the front part of the robot can swivel freely and thus do not influence the robot's movements. The robot is assumed to have a visual or other sensory system that can calculate an egocentric (robot-centered) representation of the distance and angle to arbitrary targets. All targets are assumed to lie on the same surface as the robot, and thus

only two spatial coordinates are necessary.

The control of a two-wheel robot differs in several ways from the control of an arm, both in terms of constraints and goals of the control system. First, the robot cannot generate arbitrary movements: for instance, it cannot be displaced laterally with a single movement; second, the control signals represent *angular velocities*, rather than joint angles or rotations; third, the mobile robot (in the present configuration) does not possess redundant degrees of freedom; fourth, the workspace of the robot is not limited by the robot's structure: the robot must be able to reach targets at arbitrary distances and angles. The overall goal of the model for control of the mobile robot is to learn to drive the robot to arbitrary targets presented in the workspace, while autonomously compensating for several types of perturbation, including a change in the radius of the robot's wheels, wheel slippage, and a slow or sporadic visual system; in addition, statistically significant perturbations such as changes in wheel radius should lead to further on-line adaptation. In other words, the neural network must be able to function and adapt in a novel or nonstationary environment.

Because the robot must make movements to arbitrary targets, it is not feasible to train the robot to make every possible movement in an arbitrarily large workspace. We propose instead to use a self-organizing scheme for trajectory formation that is based on the movement-error-learning cycle of the VAM and DIRECT models. Specifically, during an initial training phase, the robot makes random movements by generating random velocities at the two wheels, and learns a mapping between the velocities and the resulting displacement. We show below how this displacement can be measured using a robot-centered or external frame of reference. After sufficient training, selection of a target (that is, selection of a desired angle and distance) leads to generation of wheel velocities that move the robot in the desired direction. The scheme we propose is thus similar to the DIRECT model's use of incremental movement directions.

An important aspect of this scheme is that during the learning phase the robot learns to make movements over short distances, or over small angles. However, during the performance phase, the distance and angle to a target are passed through a compressive nonlinearity whose range cannot exceed the range of learned movements. In this way, selection of a target at a great distance and angle from the robot leads to activation of the nodes representing the largest distance and angle commensurate with the movements achieved during learning. As a result, the robot begins to move while turning towards the target. As the angle and distance diminish to values that have been learned during training, the robot selects appropriate wheel angular velocities that approach zero as the robot approaches its target. Hence the robot is able to make effective movements even when the target is much farther than any movement the robot has learned to make.

Figure 6 illustrates the main components of the control architecture that carries out the desired task. We call this the Neural NETWORK MOBILE Robot Controller, or NETMORC. A version of the model that includes stages for proprioceptive learning is presented later. The basic NETMORC architecture is similar to the VAM model, including a 2-D Target Velocity Command (TVC) population, a Difference Vector (DV) stage, and Present Velocity Command (PVC) populations for each wheel.

At the top of the diagram, two spatial populations encode distance and angle. The information represented at these populations depends on whether the robot has made a random (training) or voluntary (target reaching) movement. In the training phase, an ERG module generates alternating ON and OFF phases (Gaudiano & Grossberg, 1991). During the ERG ON phase, a random pair of activations are generated at the Present Velocity Command (PVC) populations, whose outputs generate corresponding angular velocities  $\omega_L$  and  $\omega_R$  at the left and right wheels, respectively. The robot's visual system (represented by the eye symbol and dashed line on the left of the figure) calculates the distance and angle of the resulting movement at regular time intervals.

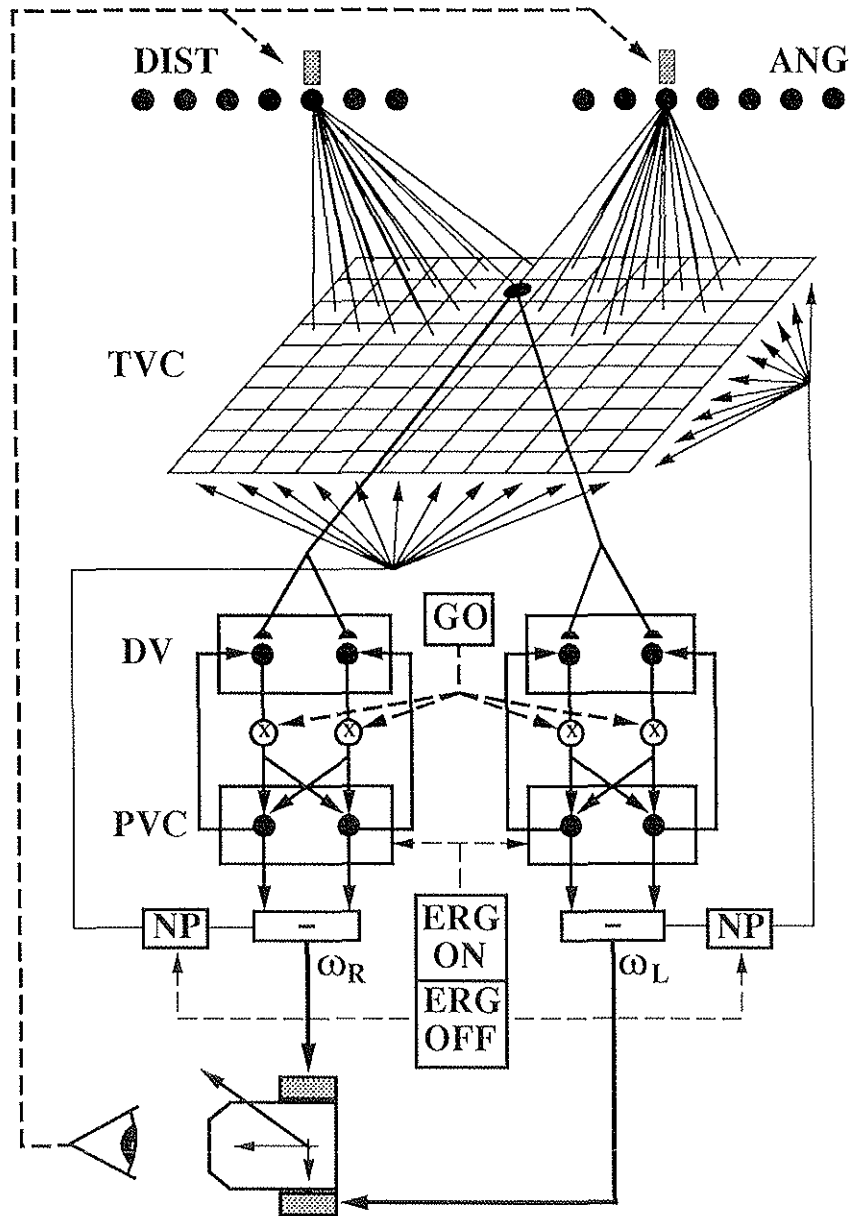


Figure 6: Architecture of the proposed NETMORC system. An intermodal VAM module learns to transform a 2-D target velocity map (TVC) into angular velocities  $\omega_L, \omega_R$  through the PVC populations, which control the robot's wheels. Fans of arrows from  $\omega_L$  and  $\omega_R$  to the TVC map represent the vector-to-spatial transform described in the text. Robot movements are registered by the visual system, which activates appropriate nodes in the ANG and DIST spatial populations (top). Connections between ANG (and DIST) nodes and the TVC map enable the NETMORC to learn a relationship between a requested displacement and the necessary wheel velocities. Semicircular endings used in other parts of the figure to represent modifiable synapses are omitted (to avoid clutter) from the pathways reaching the TVC from the DIST and ANG populations.

The calculated values, which can be positive or negative, are represented in the ANG and DIST populations.

When the ERG OFF becomes active, the PVC generates a constant velocity signal, causing the robot to move along a circular path, with radius of curvature and angular velocity that depend on  $\omega_L$  and  $\omega_R$  and on the robot's geometry, as described below. At this time the Now Print (NP) gate is activated, copying the PVC activations to the TVC via a vector-to-spatial transformation (represented by the thin line leading to a fan of arrows on either side of the TVC map). This transformation activates a single TVC node that uniquely represents a combination  $(\omega_L, \omega_R)$  of left and right angular velocities. Plausible neural mechanisms for this transformation have been described elsewhere (Grossberg & Kuperstein, 1989; Gaudiano & Grossberg, 1991). The TVC node that is activated by this transformation learns the correct PVC-generated angular velocities registered at the DV stage, via an intermodal VAM learning mechanism as described above. At the same time, the ANG and DIST nodes that have been activated by the visual system are able to learn which node in the TVC map is active via an outstar (associative) learning rule (Grossberg, 1968, 1982).

The ERG system is only active during an initial training phase. The purpose of the ERG module is to "bootstrap" the learning process. During performance trials, when the ERG system is inactive, the visual system calculates the distance and angle to a target. These calculated quantities are compressed to a small range (corresponding to the range of distances and angles generated during training), and lead to activation of one node in the ANG population and one node in the DIST population. Each of these nodes in turn activates a potentially large set of neurons in the TVC map, corresponding to all the combinations of left and right wheel velocities that can generate the same movement. We will show below that the set of TVC nodes activated by one ANG node, and the set of TVC nodes activated by one DIST node, form perpendicular flanks on the map. The node corresponding to the intersection of these flanks of activation, which receives the greatest input, corresponds uniquely to one  $(\omega_L, \omega_R)$  combination that is able to generate a displacement of the desired angle and distance. A winner-take-all competition within the TVC map selects the correct node, generating the desired velocities. (For computational efficiency we carry out the competition algorithmically, by selecting the node with the largest input.)

As the robot makes a movement in the desired direction, the robot's visual system continuously updates the distance and angle to the target. As a result, at subsequent time steps different ANG and DIST nodes that correspond to smaller distances and angles will be activated. This will typically lead to selection of a new node in the TVC that represents smaller velocities. Eventually, if the system is properly calibrated, the nodes representing zero angle and zero distance will become active, and the robot will select velocities  $\omega_L = 0$  and  $\omega_R = 0$ , stopping the robot.

The use of combined distance and angle information to select a unique pair of wheel velocities relies on the robot's structure, and depends on the robot's geometry. We now describe the transformation that occurs between angular velocities applied to the wheels and the angle and distance of the resulting movement. We will then show that the proposed architecture ensures the robustness of this control scheme.

### 3.2 Analysis of the robot's kinematics

Figure 5 shows the robot and a typical movement generated during a fixed time step  $\Delta t$  when angular velocities  $\omega_L$  and  $\omega_R$  have been selected. The angular velocities  $\omega_L$  and  $\omega_R$  are multiplied by the left and right wheel radii  $R_L$  and  $R_R$ , respectively, to determine the resulting (linear)

velocities  $V_L$  and  $V_R$ .<sup>1</sup> The combination of  $V_L$  and  $V_R$  and the distance  $d_W$  between the wheels jointly determine the robot's instantaneous center of rotation  $r_C$ , measured relative to a frame of reference centered between the robot's wheels:

$$r_C = \frac{d_W V_T}{V_R - V_L}, \quad (1)$$

where the instantaneous tangential velocity  $V_T$  is given by

$$V_T = \frac{V_L + V_R}{2}. \quad (2)$$

In the training phase, learning takes place during the ERG OFF phase, when the velocities are constant. In general, the velocities can be assumed to be constant for a short time period even during performance. Hence during a time  $\Delta t$ , given a pair of angular velocities  $\omega_L$  and  $\omega_R$  the robot will move along a circular path of radius  $r_C$ , rotating through an angle

$$\Delta\theta = \Delta t \frac{V_T}{r_C} = \Delta t \frac{V_R - V_L}{d_W}, \quad (3)$$

and covering a distance

$$\Delta S = \Delta\theta r_C = \Delta t V_T = \Delta t \frac{V_L + V_R}{2}. \quad (4)$$

Equations (3) and (4) show that the transformation between wheel velocity combinations  $(V_L, V_R)$  and distance (or angle) is many-to-one, that is, there exists an infinite number of  $(V_L, V_R)$  combinations that move the robot through a given distance  $\Delta S$  (or angle  $\Delta\theta$ ). However, these equations also show that specification of both  $\Delta S$  and  $\Delta\theta$  uniquely determines a combination  $(V_L, V_R)$ . This property of the robot's kinematics is fundamental to the function of the NETMORC model, as shown in the following sections.

### 3.3 Learning the velocity-to-displacement mappings

As described earlier, the robot carries out movements at random velocities during the training phase. At the end of each time step  $\Delta t$ , the robot calculates the angle  $\Delta\theta$  and distance  $\Delta S$  it has covered during that time step, and activates the corresponding ANG and DIST nodes (Fig. 6). In order to learn the one-to-many transformation from each ANG (or DIST) node to the corresponding pair of angular velocities  $(\omega_L, \omega_R)$ , it is first necessary to transform the angular velocities into a 2-D spatial map (labeled TVC in Fig. 6) representing  $\omega_L$  along one axis and  $\omega_R$  along the other axis. This is done through hard-wired vector-to-spatial transformations, which are depicted in Fig. 6 as thin dashed lines leading from the output of each channel, through the NP gates, and up to a fan of arrows reaching the TVC map. A simple neural network designed to carry out a 2-D vector-to-spatial transformation of this type was first proposed by Grossberg and Kuperstein (1989). In the present implementation, the transformation is carried out algorithmically through a piecewise-linear sigmoid function. Specifically, the node  $T_{i,r}$  is activated in response to angular velocities  $\omega_L$  and  $\omega_R$  according to:

$$l = \begin{cases} 0 & \text{if } N_L(\omega_L + M_L)/(2M_L) < 0 \\ N_L - 1 & \text{if } N_L(\omega_L + M_L)/(2M_L) \geq N_L \\ \text{int}[N_L(\omega_L + M_L)/(2M_L)] & \text{otherwise} \end{cases} \quad (5)$$

<sup>1</sup>Because of the constant relationship between linear and angular velocity, either of these quantities can be used to describe the robot's behavior. In the remainder of the article we generally use linear velocities.

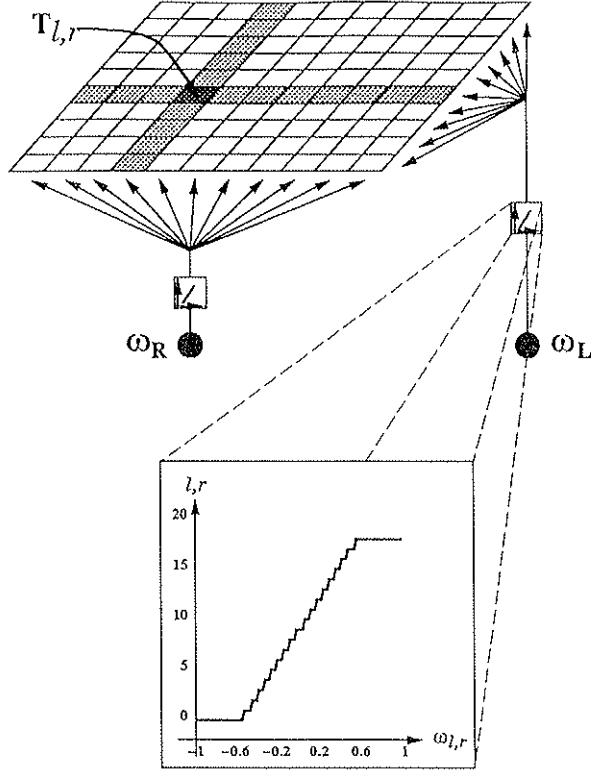


Figure 7: Illustration of the vector-to-spatial transformation that maps PVC activations into TVC spatial velocity map. A given PVC activation level in the left channel activates one row of TVC nodes whose position varies with PVC activation according to the function shown in the lower inset. PVC activation in the right channel likewise generates activation over a column of TVC nodes. Competitive interactions within the TVC map lead to activation of the node  $T_{l,r}$  at the intersection of the activated flanks, while activation of other nodes is suppressed.

and

$$r = \begin{cases} 0 & \text{if } N_R(\omega_R + M_R)/(2M_R) < 0 \\ N_R - 1 & \text{if } N_R(\omega_R + M_R)/(2M_R) \geq N_R \\ \text{int}[N_R(\omega_R + M_R)/(2M_R)] & \text{otherwise,} \end{cases} \quad (6)$$

where  $\text{int}(x)$  returns the integer value of  $x$  truncated towards zero;  $N_L$  and  $N_R$ , respectively, are the number of nodes in the  $L$  and  $R$  dimensions of the TVC map; and  $M_L$  and  $M_R$ , respectively, are the maximum left and right wheel angular velocities represented in the TVC map. In summary, Eqs. (5) and (6) map velocities in the range  $[-M_{(L,R)}, M_{(L,R)}]$  onto a population of nodes  $T_{l,r}$ , where  $0 \leq l < N_L$  and  $0 \leq r < N_R$ .

Figure 7 shows the piecewise-linear sigmoidal transformation for one dimension (inset), and a rendition of how Eqs. (5) and (6) jointly select a single node  $T_{l,r}$ . Each of the equations activates one row of nodes in the TVC map, with  $\omega_L$  and  $\omega_R$  activating perpendicular rows. Competitive interactions within the TVC map can lead to activation of the node receiving the largest inputs, which in this case is simply chosen to be the node at the intersection  $(l, r)$  of the perpendicular activated rows specified by equations (5) and (6).

The activated node  $T_{l,r}$  takes part in the learning of two distinct mappings: first, the active ANG and DIST nodes learn the activation at  $T_{l,r}$  through an associative learning law; second, the node  $T_{l,r}$  learns to generate the correct angular velocities  $\omega_L$  and  $\omega_R$  through VAM learning. The associative learning from the ANG and DIST maps to the TVC map is described in the next section;

the VAM learning between the TVC and the PVC is identical to that described by Gaudio and Grossberg (1991), and summarized in Appendix A.

To summarize the model's function, during the initial training phase the ERG ON channels generates random angular velocities  $\omega_L$  and  $\omega_R$  at the PVC populations. The ERG OFF channel activates the NP gate, leading to a transformation of the PVC activations into a 2-D spatial TVC map of angular velocities, represented by activation of a single node  $T_{l,r}$ . The selected TVC node learns to generate the PVC activations corresponding to the correct angular velocities by means of VAM learning. The ERG OFF channel also causes the robot's visual system to determine the distance and angle covered in a fixed, short time interval  $\Delta t$ . This information is represented by activation of one node in a 1-D map of distances (ANG), and another node in a 1-D spatial map of angles (DIST). The activated ANG and DIST nodes learn an association to the active TVC node. During performance, a target registered by the visual system at a particular distance and angle activates ANG and DIST nodes, and subsequently a TVC node that represent appropriate combinations of angular velocities required to move towards that target. We now turn to a description of the associative learning mechanism and of its properties.

### 3.4 Transformation of displacements to velocities

Associative learning between ANG or DIST nodes and the TVC is based on Grossberg's *outstar* learning law (Grossberg, 1968, 1982). The outstar law is a form of associative law in which learning is gated by presynaptic activation. In the present case we use a form of the outstar law that combines presynaptically gated learning and decay:

$$\frac{dZ_{i,l,r}^A}{dt} = A_i n_p (T_{l,r} - Z_{i,l,r}^A), \quad (7)$$

where  $Z_{i,l,r}^A$  refers to the weight connecting the  $i$ th node in the ANG map to the  $(l, r)$  node in the TVC map, and  $n_p$  represents activation of the Now Print gate, with  $n_p = 1.0$  when the Now Print is active (typically during ERG OFF phases) and  $n_p = 0.0$  otherwise. (The equation for DIST nodes is identical, with  $A$  being replaced by  $D$ , and therefore not shown for simplicity). This equation shows that the weights can only change when the ANG node is active ( $A_i > 0$ ). When  $A_i > 0$ , each weight  $Z_{i,l,r}^A$  is drawn towards the activation  $T_{l,r}$  of the corresponding TVC node. Hence when an ANG node samples the activation of TVC cells, the weight connecting the ANG node to the active TVC node (representing the angular velocities  $\omega_L$  and  $\omega_R$ ) increases in strength, while all other weights decrease slightly toward zero. The same holds for DIST nodes and their weights.

For simplicity and speed of computation, we have replaced the differential form of Eq. (7) with a discrete-time version. Specifically, since only one TVC node can be active at a given time, and since all node activations are always 1.0 (for the active TVC node) or 0.0 (for all others), Eq. (7) is replaced by

$$Z_{i,l,r}^A(t) = Z_{i,l,r}^A(t-1) + n_p \delta [1.0 - Z_{i,l,r}^A(t-1)], \quad (8)$$

where  $\delta$  is the learning rate. This equation is intrinsically gated by the presynaptic activation because it is invoked only for the active ANG (and DIST) node.

Eq. (8) only specifies an increase in strength for the weight connecting the active ANG (or DIST) node to the active TVC node. Because each ANG (and DIST) node is connected to all TVC nodes, it would be inefficient to adjust all the weights for the active node at each time step. Instead, the program keeps track of how many times each ANG or DIST node has been active; when this has happened a fixed number of times (usually twenty), we decrease all weights originating from the active ANG or DIST node by a small amount. Simulations have shown that this expedient leads

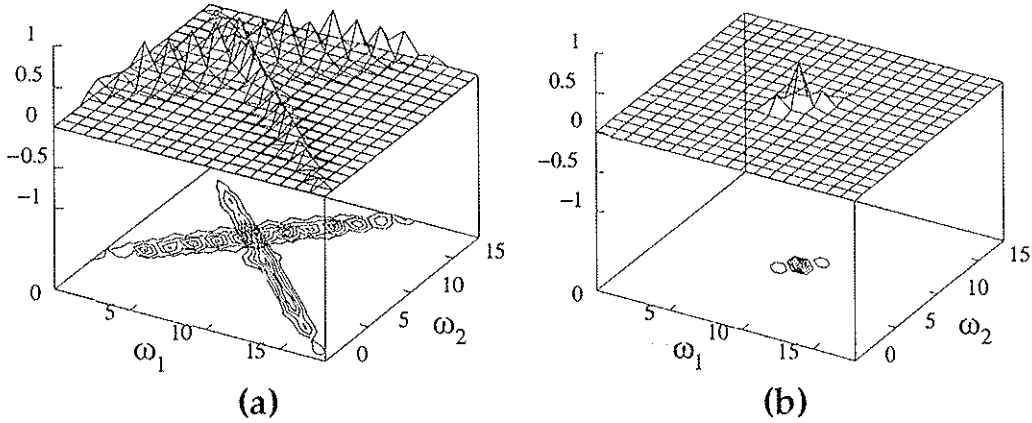


Figure 8: Surface and contour plots showing connectivity pattern from ANG and DIST nodes to the TVC map. (a): Superimposed plots of the inputs converging on the TVC map from activation of one ANG node and one DIST node. Note that activation patterns are perpendicular, and show a well-defined intersection. (b): Plot of the net TVC activation resulting from simultaneous activation of one ANG and one DIST node as in part (a). The TVC node at the intersection of the two flanks will be selected as a result of subsequent competitive interactions.

to learned maps that are virtually identical to those obtained with decay applied at every time step, while the computational load is significantly reduced.

### 3.4.1 Simulation of the displacement to velocity transformation

Equations (3) and (4) show the relationship between a given pair of angular velocities and the corresponding distance and angle covered in one time step. These equations show that any velocity combination such that  $\omega_L - \omega_R$  is constant leads to a displacement of constant angle  $\Delta\theta$ , while any velocity combination such that  $\omega_L + \omega_R$  is constant leads to a displacement of constant distance  $\Delta S$ . As a result, since the TVC map is a 2-D Euclidean map with  $\omega_L$  and  $\omega_R$  as axes, each node in the ANG map should be connected with a flank of TVC nodes forming a line of slope 1, while each node in the DIST map should be connected with a flank of TVC nodes forming a line of slope -1. Figure 8a shows surface and contour plots of the inputs converging upon the TVC map when one DIST and one ANG node are activated. As expected, the DIST and ANG nodes project perpendicular flanks with a clearly defined intersection, showing that the network has learned the correct transformations.

Figure 8b shows the net pattern of activation at the TVC map resulting from simultaneous activation of one node in the ANG map and one node in the DIST map, as in Fig. 8a. Just as competitive interactions are assumed to select a single winner during the training phase, the simultaneous activation of one ANG and one DIST node during performance leads to selective activation of the TVC node located at the intersection of the two flanks. This is accomplished by assuming that the TVC node activation is proportional to the *product* of the converging ANG and DIST activations, and algorithmically selecting the TVC node with the maximal activation. Specifically, given that the  $i$ th node in the ANG map and the  $j$ th node in the DIST maps are active, the activation of each node  $T_{l,r}$  is given by:

$$T_{l,r} = (A_i Z_{i,l,r}^A - \Gamma) (D_j Z_{i,l,r}^D - \Gamma) \quad (9)$$

where  $\Gamma$  is a small threshold. The resulting activation pattern (Fig. 8b) shows a clear peak at a

single node, which will be selected to read out its weights to the DV, and—as long as the GO signal is positive—lead to subsequent readout of the correct angular velocities  $\omega_L$  and  $\omega_R$ .

### 3.5 Learning with inaccurate sensory information

The results of the previous section are based on the assumption that the NETMORC model has access to an accurate representation of the angle and distance covered by the robot during a small random displacement. In a practical application this would require the sensory information to come from a fixed, external frame of reference. This information is only necessary during the training phase: during performance, the distance and angle to a target are always computed in a robot-centered reference frame. However, there may be practical applications in which an external sensory system is not available during training. This is particularly important if the NETMORC is continuously adapting to its environment, as described in Section 4.5. Alternatively, the exact movement can be derived easily from the angular velocities if the wheel radii are known, as shown earlier. However, there may be instances when the wheel radii are unknown or have changed unexpectedly.

Suppose that the robot has on-board sensory hardware that enables it to calculate the distance  $d$  and angle  $\alpha$  to an arbitrary target (both quantities are measured in robot-centered coordinates). For example, this information could be derived from a pair of cameras, or from a combination of cameras and range finders. Can the robot deduce the angle and distance of its movements by evaluating the change in angle and distance of an arbitrary landmark resulting from its displacement? Mathematically, this problem is ill-posed: for a given movement, the change in angle and distance of a fixed landmark depends on the position of the landmark relative to the robot. Assuming for instance that the robot has made a small straight-ahead displacement, the robot would register a correct angle of zero degrees only if the landmark is straight ahead. As a further example, if the robot is moving along a circular path (which is usually the case during the ERG OFF phase, since the angular velocities are constant), then the visual system will register no change in angle or distance relative to a landmark at the instantaneous center of rotation, regardless of the length of the movement and (constant) radius of curvature. Hence if the sensory system is mounted on the robot, it is generally not possible for the NETMORC model to determine the robot's movement by observing the displacement of landmarks in the environment. We now show that in spite of this difficulty, the NETMORC model is able to learn a good approximation of the displacement-to-velocity transform on the basis of information from an on-board sensory system, by monitoring the relative displacement of landmarks in the environment.

Figure 9a shows the relationship between the actual angle ( $\Delta\theta$ ) and distance ( $\Delta S$ ) covered during a constant-velocity movement in a fixed time step  $\Delta t$ , and the measured change in angle ( $\alpha_1 - \alpha_2$ ) and distance ( $d_1 - d_2$ ) between the robot and an arbitrary, fixed target. It is assumed that the robot only has access to the distances  $d_1$  and  $d_2$  and angles  $\alpha_1$  and  $\alpha_2$ .

From simple geometric considerations one can show that the angles  $\alpha_1$  and  $\alpha_2$  are related to the angle  $\Delta\theta$  by:

$$\alpha_1 - \alpha_2 = \Delta\theta - \gamma \quad (10)$$

where  $\gamma$  is the angle subtended by the robot's movement relative to the landmark. Eq. (10) shows that the robot's estimate ( $\alpha_1 - \alpha_2$ ) of the actual angle  $\Delta\theta$  is correct only for  $\gamma = 0$ , which corresponds to a straight-line movement with a dead-ahead landmark. However, if the angle between robot and target over a long series of movements is random with a mean of zero, the measurement errors will likewise average out to zero. If the learning rate  $\delta$  in Eq. (8) is small, the robot will learn the correct ANG to TVC transformation.

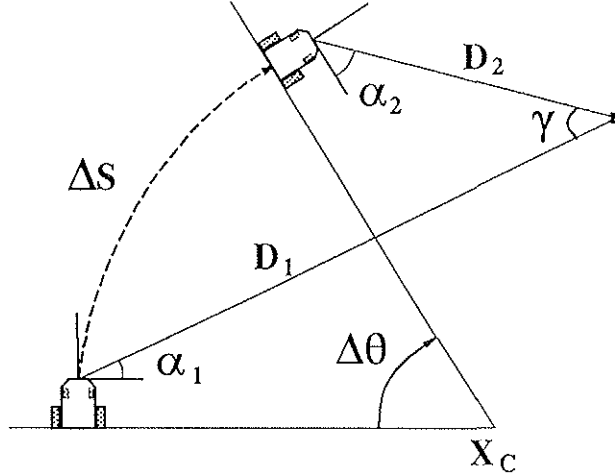


Figure 9: Diagram of the relationship between the angle  $\Delta\theta$  and distance  $\Delta S$  covered during robot movement along a circular path, and the initial and final angles ( $\alpha_1$  and  $\alpha_2$ , respectively) and distances ( $D_1$  and  $D_2$ , respectively) between the robot and an arbitrary target (small black square).

We tested this prediction by selecting a single stationary landmark during the entire random training phase. The resulting weight distribution was almost identical to the one acquired through external sensory information (Fig. 8a). Similar results obtain if a random landmark is chosen during each ERG OFF phase, or if the robot selects landmarks that happen to be in its visual field, as long as no systematic bias is introduced in the sampling. A similar result holds true for the calculated distance  $\Delta S$  of a movement. In this case each DIST node can be activated by a large combination of  $\omega_R$  and  $\omega_L$ , but on average the most likely velocity combinations are the correct ones. All of the results in the remainder of the article were checked using weights learned with either type of sensory system, with nearly identical results. The only difference arises when the robot is performing blind movements, as described later.

This completes our description of the main components of the NETMORC model. A more detailed analysis of the stability of the proposed mapping will appear in a future manuscript. We now demonstrate the model's performance on several types of movement control.

## 4 Model performance

The previous sections described the NETMORC model and showed that this model is able to learn the transformations necessary for movement to an arbitrary target. We now illustrate the performance of the mobile robot when it is controlled by the NETMORC.

The NETMORC is first trained by letting the robot perform many random movements, based on cyclic ERG activations. Performance is then tested on a variety of tasks. In general, a single simulation consists of defining the robot's initial position and orientation, as well as the position of the target, or the starting point of the trajectory to be followed. Each trial is begun by presenting the target to the robot, and turning on the GO signal to initiate movement.

The GO signal controls the integration rate of the PVC, so that effectively it controls the robot's acceleration. In all simulations below, the GO signal  $g$  is set to zero at the beginning of each movement, and is ramped up over time to a maximum value of  $g_{\max}$  according to the equation:

$$g(t) = \frac{g_{\max} t^2}{0.5 + t^2} \quad (11)$$

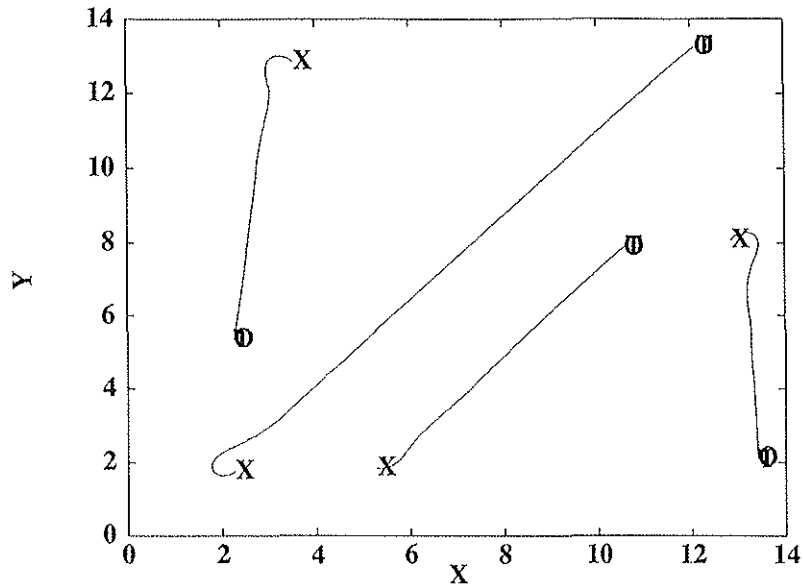


Figure 10: Spatial paths followed by the robot during reaching movements to four targets (T), with four different starting positions (X). Final positions (O) are always within 5cm of target. Not that for most of the movements the robot is initially facing away from the target, as evident from the initial curvature in the robot's path. X and Y dimensions are in meters.

The GO signal thus rises quickly to half of  $g_{max}$  in 0.5 seconds, and then asymptotically approaches its maximum value. Unless otherwise noted, we chose  $g_{max} = 2.0$  in all simulations below.

#### 4.1 Reaching arbitrary targets

As a first example, we illustrate the model's most basic competence, viz., its ability to guide the mobile robot to an arbitrary target in space. Figure 10 depicts the robot performing a series of movements with different starting positions (marked by X), target positions (marked by T), and final positions (marked by O) for each movement. The robot is able to perform movements to targets that appear anywhere in the workspace, near or far, in front or behind. In all cases, unless specified, the goal consists of bringing the midpoint between the robot's driving wheels to the target with an accuracy of 0.05 meters (with a robot that is 1.2 meters long and 0.6 meters wide). Note that during training the robot learns both forward and backward movements, so that in theory it can back up to any target behind it. In this case the program only allows selection of negative velocities if the target is within 1 meter.

Figure 11 provides more detailed data on a movement to target similar to the longest movement in Fig. 10. In all parts of the figure, the solid lines represent data from movements with  $g_{max} = 2.0$ , while the dashed lines represent data for movements with  $g_{max} = 1.0$ , to illustrate the effect of the GO signal.

The angle and distance to target during the movement are displayed in Fig. 11a and b, respectively. The target was initially located directly behind the robot (initial angle of approximately  $180^\circ$ ), at a distance of approximately 15 meters. The robot has completed its turn in approximately ten seconds (Fig. 11a), and reaches the target in about 50 seconds (Fig. 11b).

Figures 11c and d show the angular and linear velocities during the same movement, again for two different choices of  $g_{max}$ . The robot's angular velocity rises quickly to its maximum value and then returns near zero with only minor oscillations. The linear velocity rises almost linearly

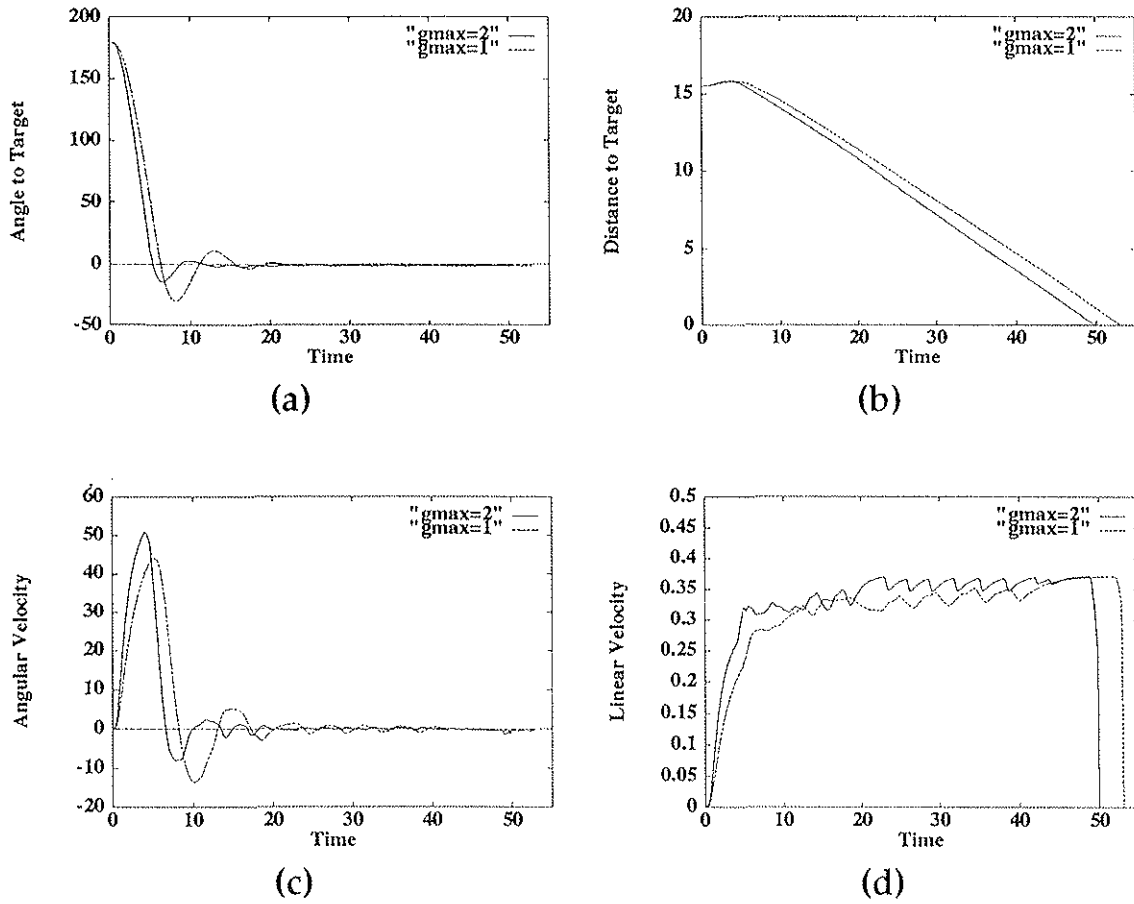


Figure 11: Detailed plots of the robot's movement characteristics along a path similar to the longest one in Fig 10. For all plots X axis represents time in seconds. Dashed lines: results when  $g_{max} = 1.0$ ; solid line: results when  $g_{max} = 2.0$ . (a): Angle to target (degrees). (b): Distance to target (meters). (c): Angular velocity (degrees/second). (d): Linear velocity (meters/second).

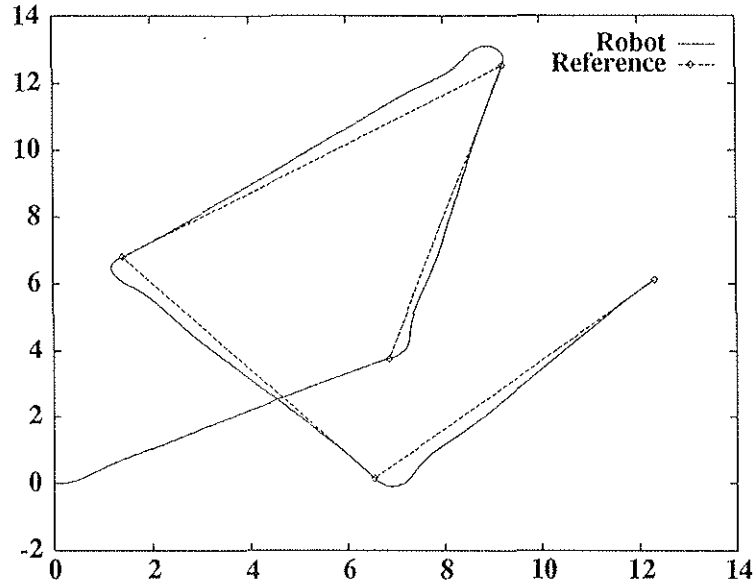


Figure 12: Path followed by the robot (solid line) while reaching a sequence of targets (diamond symbols). Initial position is at (0,0), and GO signal starts at zero. A new random target is selected as soon as the robot has reached the previous target, without reducing the GO signal. Note that robot swings around each target towards new target because of large GO signal.

(constant acceleration) to a maximum value of 0.3 meters/second, and then drops, again linearly, back to zero as the robot reaches the target. This type of velocity profile is characteristic of all the simulations we present in this article.

Comparison of the solid and dashed lines in Fig. 11d illustrates the effect of the GO signal maximum value  $g_{\max}$ : in both cases, the asymptotic maximum velocity is the same. However, doubling  $g_{\max}$  increases the slope of the initial portion of the velocity profile, *i.e.*, the acceleration. Interestingly, Fig. 11c-d shows that the higher value of the GO signal *reduces* the oscillations in angular and linear velocity. This is because a larger GO signal makes the system respond more rapidly to requested velocity changes. In a realistic model including dynamics, a higher GO signal would be expected to cause more oscillations.

## 4.2 Reaching a sequence of targets

In some cases, it may be desirable to guide the robot through a sequence of targets. For instance, the robot might be instructed to pick up a series of objects, or to navigate through a building or other environment by indicating a series of way points. In this case, a new target can be selected as soon as a given target is reached, and the robot's GO signal does not have to be reset to zero until all targets in the sequence have been reached. Hence the robot will follow a more or less continuous trajectory that is constrained to pass through all way points. Figure 12 illustrates the performance of the robot as it reaches a sequence of five random targets, with each target appearing only after the robot has reached the prior target. In this example the GO signal is not reset at each point, so the robot's path is seen to swing past each target as if due to momentum. If more accurate control is necessary, the GO signal can be reset to zero after each target has been reached. In this case each movement would be similar to those shown in Fig. 10.

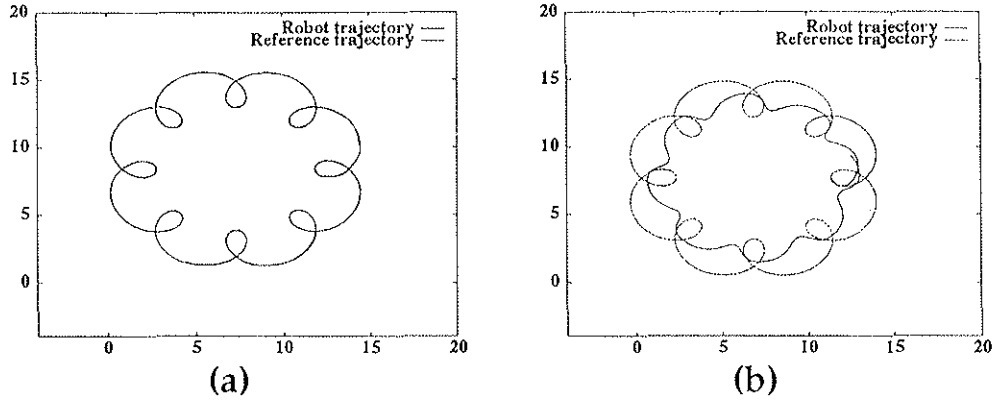


Figure 13: Robot path in space (solid line) while following a reference trajectory (dashed line). All dimensions in meters. (a): the robot is able to follow the trajectory when the moving target is slow ( $k_1 = 6.0$  meters,  $k_2 = 0.01$  radians/second). (b): If the target moves at a speed much greater than the robot's maximum speed ( $k_2 = 0.05$  radians/second), the robot constantly tracks the moving target but is not able to follow the exact trajectory.

### 4.3 Following a trajectory

Because the ANG and DIST nodes are updated at every time step through sensory information, the NETMORC can guide the mobile robot to a moving target. In this case, the robot will follow a trajectory that is directed toward the target at each time step (or as frequently as the sensory information is updated). If the target is moving at a velocity inferior to the robot's maximum velocity, the robot will reach the target and subsequently follow the target's trajectory accurately. This behavior is illustrated in Fig. 13a. The trajectory in this case is described by a pair of parametric equations describing the  $x$  and  $y$  components of the movement:

$$T_x(t) = k_1 \cos(k_2 t) - k_1/4 \cos(9k_2 t) \quad \text{and} \quad (12)$$

$$T_y(t) = k_1 \sin(k_2 t) - k_1/4 \sin(9k_2 t). \quad (13)$$

When the target is following a curvilinear trajectory, if the robot is not very close to the target, or if the target's speed is greater than the robot's maximum speed, the robot will not follow the target's trajectory, but rather it will follow the path that keeps it constantly aimed toward the target. This type of behavior, which is illustrated in Fig. 13b, may be desirable in certain environments wherein the robot's trajectory does not need to be specified explicitly.

For those situations in which the trajectory is important (for instance, when working within an environment with obstacles), a trajectory can be specified as a series of way points as shown in the previous section. The ability to use either type of trajectory control leads to a very flexible control system that can be customized to a variety of environments.

It is interesting to point out that for this simulation, the NETMORC was trying to place the moving target at a point centered between the wheels, but now 20cm in front of the rear axle. This is necessary because for sufficiently slow targets the robot will be almost always at the desired location. If the NETMORC tries to place the target directly under the axle, the robot exhibits oscillations while tracking the target. The oscillations result when the robot has momentarily reached the target, but the target makes a highly curved movement, which leads to rapid, large changes in the relative angle between the robot and the target. The NETMORC tries to compensate

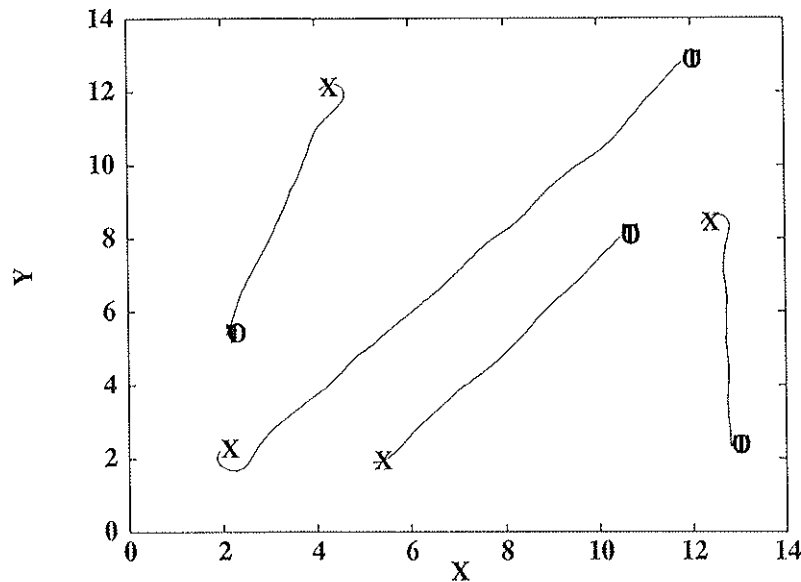


Figure 14: Spatial paths followed by the robot during reaching movements to four targets when random wheel slippage occurs. All conventions as in Fig. 10.

for this by generating rapid turns, which lead to oscillations. Selecting a reference point on the robot eliminates this problem. The ability to select an arbitrary reference point *after* the robot has been trained is another example of the NETMORC's flexibility.

#### 4.4 Performance under perturbations

One of the main reasons for basing the NETMORC on the VAM and DIRECT architectures is because these neural networks exhibit real-time error correction abilities. The NETMORC in fact can easily compensate for a variety of perturbations. The previous section already illustrated the model's ability to compensate for changes in target position. We now demonstrate the NETMORC's ability to compensate for two other types of perturbation.

Figure 14 shows the NETMORC's ability to compensate for wheel slippage. In this simulation, each wheel is independently perturbed with a 0.5 probability per second (in other words, on average each wheel is perturbed every 2 seconds). The perturbation consists of reducing the wheel's velocity by a random amount to mimic the effect of slippage. Specifically, at each time step (0.2sec), with a 10% probability the velocity is set to a random value (uniform distribution) between zero and the wheel's actual velocity, as specified by the PVC activations. The figure shows that although the resulting movement trajectories are not as smooth, the robot is still able to reach the target successfully.

Figure 15 shows the robot's performance under a more dramatic perturbation: here we have changed the radius of both wheels, reducing one wheel from 25cm to 15cm, and the other wheel from 25cm to 20cm. This change affects the robot's performance in two ways. First, the overall reduction in wheel size leads to a reduction in the effective maximum velocity. Although the robot will move more slowly, it will still be able to reach the target, because nodes corresponding to nonzero velocities will continue to be selected until the target is reached. Second, selection of a node that normally generates equal angular velocities (and thus a linear trajectory) leads to different linear velocities generated by each wheel because of the different radii. As a result, the robot is unable to follow the normal straight trajectory (dotted line). However, because the

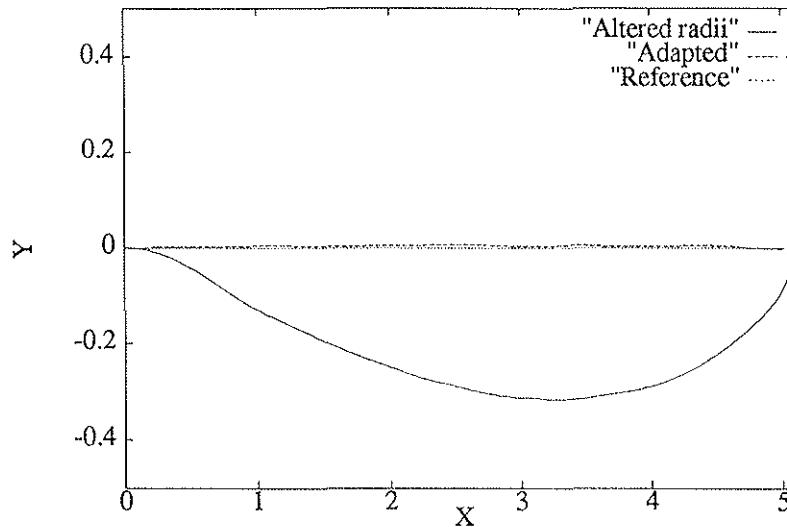


Figure 15: Path followed by the robot during movement to a target located straight ahead at a distance of 5 meters. The solid line shows that the robot follows a curved path when the wheel radii are changed drastically. The dashed line shows that the robot learns to compensate for this change, so that after further training the robot follows a nearly straight trajectory. For comparison, the thin dotted line is the straight line connecting the starting position and target. Note that Y-axis scale differs from X-axis scale (all dimensions in meters).

NETMORC relies on visual information, the robot is still able to reach the target, albeit following a curved trajectory (solid line). A similar result was obtained with the DIRECT model (Bullock et al., 1993) when the visual information was passed through distorting prisms that rotated the entire visual field by a fixed amount.

#### 4.5 Long-term adaptation to perturbations

The controller's ability to compensate instantly for sudden, unexpected perturbations in the environment (such as a target changing position or transient wheel slippage) derives from its use of continuous visual feedback. In addition, any statistically significant perturbation in the robot's plant leads to adaptation that will tend to correct for the perturbation. To illustrate this property of the model, we changed the wheel radii as before, and then allowed the robot to perform a new series of random, endogenously generated movements. The dashed line in Fig. 15 illustrates the robot's reaching behavior after the new training phase: the robot has learned to compensate for the new wheel radii, and it is again able to follow a straight trajectory when necessary.

The real-time nature of the learning laws used in the NETMORC suggests that it should be possible for the robot to continuously adapt even during normal performance. The example used above is an extreme perturbation that is unlikely to occur during performance; it is more likely that during normal function the robot's plant will undergo gradual changes, such as a progressive decrease in wheel radii resulting from normal wear. It would be desirable for the NETMORC to adapt to gradual changes without the need for retraining.

The mapping from TVC to PVC depends only on the internal transformation described in section 3.3, and thus is unaffected by the change in wheel radii. The mapping from ANG and DIST populations to the TVC map needs to be modified to reflect the new relationship between the angular velocities  $\omega_L$  and  $\omega_R$ , and the distance and angle of the resulting displacement.

Changing wheel radii causes a shift in the DTV (displacement-to-velocity) mapping, *i.e.*, in the relationship between each ANG (or DIST) node and the TVC map. The decay term in the associative learning equation (8) ensures that each ANG and DIST node will “forget” the old pattern of weights and learn a new one. If the robot is subjected to a new training phase, the remapping will take place in the model as in the initial training phase described above. However, special measures must be taken in order to allow the robot to learn the new mapping while it is performing reaching movements. This is because activation of a given ANG (or DIST) node has different meanings during random and during voluntary movements: during a random movement, the active ANG node reflects the angle through which the robot has rotated during the preceding fixed time step  $\Delta t$ ; in contrast, during an erroneous voluntary movement, the active ANG node represents the angle between the robot and the target. Hence a miscalibration that causes the robot to turn five degrees to the right during a fixed time step  $\Delta t$  will cause the target to move five degrees *to the left* relative to the robot.

In order to circumvent this problem, the robot has to keep track of the distance and angle traveled through each time interval  $\Delta t$  even during the performance phase. For simplicity, we simply allow NETMORC to monitor both quantities at all times, *i.e.*, the robot is aware both of its own movements and the target’s location. During performance, the robot activates ANG and DIST nodes representing the target. At fixed time intervals  $\Delta t$  the robot temporarily activates the ANG and DIST nodes that correspond to the robot’s own movement, and learning takes place before the target coordinates are reinstated.

This solution, while easy to implement in software or hardware, may be difficult to implement in a biological neural network because it requires continuous, rapid switching between two complementary types of information within the same population. A more plausible solution may be to intersperse short periods of learning during voluntary movements. Nonetheless, the solution we adopt leads to continuous, efficient recoding of the DTV mapping.

## 5 Robustness

The previous sections illustrate the flexibility of the NETMORC in terms of both rapid and long-term adaptation to perturbations. These properties arise from the intrinsic error-reducing properties of the mappings we used. In this section we show that the DTV mapping endows the NETMORC with additional properties of fault-tolerance and robustness.

The parallel, distributed nature of neural network architectures typically leads to robustness and to graceful degradation in response to noise and other forms of perturbation. Typically, this property of neural networks arises from the redundant nature of the distributed information. In the following sections we show that the NETMORC enjoys the usual fault-tolerant characteristics of neural networks, and that the nature of the DTV mapping leads to additional robustness. As a result, the proposed model exhibits robustness and flexibility that make it ideally suited for practical implementations in novel and nonstationary environments.

### 5.1 Adding noise to the weights

The DTV mapping described in Sec. 3.4, contains a significant amount of redundancy. For instance, Fig. 8 shows that each ANG and DIST node activates several TVC nodes. This redundancy suggests that the network should be resistant to random pruning of weights.

We have tested the robustness of the DTV map by setting to zero a random fraction of the weights from the ANG and DIST maps to the TVC map. Figure 16 shows the movement to a

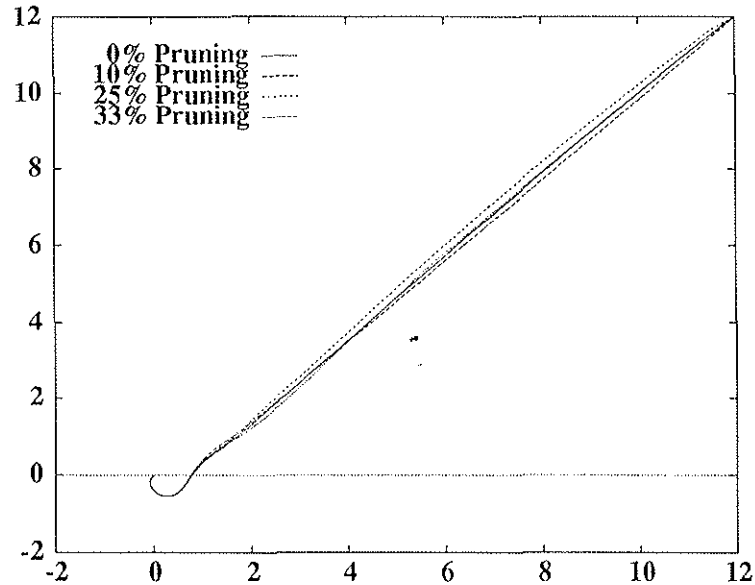


Figure 16: Path followed by the robot starting from the origin (0,0), and reaching a target located at (12,12). The robot's initial orientation ( $225^\circ$ ) is directly opposite to the target. The different traces show the effect of pruning 0%, 10%, 25% or 33% of the weights. The robot's performance is nearly identical in all cases.

target initially located approximately 15m directly behind the robot. The four traces show the performance when the fraction of weights set to zero is equal to 0, 10%, 25%, and 33%. The figure shows that the robot's movement remains largely accurate even with 33% of the weights set to zero. Pruning 50% of the weights usually led to instabilities as certain combinations of ANG and DIST nodes gave no convergent activation at the TVC map.

This sort of systematic pruning can be used to increase the NETMORC's speed and efficiency without disrupting performance. For instance, when the robot is performing a long movement (*i.e.*, when the target is activating the maximum DIST node), the NETMORC can use a random 25% subset of the weights to the TVC map for selection of the desired velocities  $\omega_L$  and  $\omega_R$  at each time step, thus increasing the speed of the computation.

## 5.2 Inverse mapping for blind reaching

As described earlier, the DIRECT model includes maps capable of learning the inverse mapping from joint angle increments to spatial coordinates of the end effector. A similar inverse mapping can be added to the NETMORC, so that the robot learns how its movements modify the position of the target. In other words, the robot can learn about its odometry. Once this map is learned, the robot can update the position of the target on the basis of internal information. This property of the model is extremely useful in robotics, especially when visual information is sporadic or subject to temporary disturbances. For example, if visual signals from a camera must be processed for object recognition, the rate of update of visual information may be very slow, forcing the robot to operate slowly. The inverse mapping allows the robot to navigate securely while the visual information is being updated.

An additional advantage of this scheme is that the robot can scan the environment for additional stimuli while maintaining its heading toward a prescribed goal via the inverse mapping. For example, suppose the robot is navigating down a hallway that includes doorways or turns. Then the robot could set internal targets near the end of the hallway, and update its position internally

while scanning doorways and other hallways.

Figure 17 illustrates the NETMORC architecture including the circuitry used to guide the robot in the absence of visual information in the upper left portion of the figure. During normal operation, the angle and distance to target are updated via visual feedback, activating the nodes labeled  $ANG_V$  and  $DIST_V$ . This information is encoded in vector coordinates, that is, the amplitude of activation of the  $ANG_V$  (or  $DIST_V$ ) node represents the angle (or distance) to target. This information is transformed via a compressive vector-to-spatial transformation (represented in Fig. 17 as the square boxes containing a sigmoid) of the type described earlier, and activates the spatial  $ANG$  and  $DIST$  populations as above.

The  $ANG_V$  and  $DIST_V$  populations are used also to calculate incremental angle and distance covered during a fixed time step. The incremental distance ( $\Delta DIST$ ) and angle ( $\Delta ANG$ ) are derived from the visual information by subtracting from the  $ANG_V$  (or  $DIST_V$ ) signal a delayed copy of itself, the result of which is represented by the activation of the  $\Delta ANG$  and  $\Delta DIST$  nodes in Fig. 17. During training, the switches at the top of the diagram would be changed to the *learn* position, so that the  $DIST$  and  $ANG$  populations are receiving incremental, rather than absolute information about distance and angle.

The  $\Delta DIST$  and  $\Delta ANG$  populations serve another purpose, namely, they allow the robot to learn the effects of its own movements. Whenever the robot makes a movement, be it voluntary or random, the active TVC node is allowed to learn the resulting  $\Delta ANG$  and  $\Delta DIST$  activities via a VAM module (in this case the  $\Delta ANG$  and  $\Delta DIST$  populations correspond to the PPC population). In essence, this additional VAM is learning the robot's odometry, that is, the relationship between wheel angular velocities ( $\omega_L, \omega_R$ ) and the resulting distance and angle covered in a fixed time step. Figure 17 shows only the connection from one TVC node to the DV populations which are also connected to the  $\Delta DIST$  and  $\Delta ANG$  populations. In reality all TVC nodes makes similar connections.

During performance, suppose that a target has been initially registered at the  $DIST_V$  and  $ANG_V$  populations. Suppose also that the visual feedback is then disabled. The active TVC node reads out the ( $\Delta DIST, \Delta ANG$ ) pair that it has learned. The target's distance and angle are subsequently updated by decrementing  $ANG_V$  and  $DIST_V$  at a rate proportional to the  $\Delta ANG$  and  $\Delta DIST$  activations. This is represented by the thick dashed arrows connecting  $\Delta DIST$  and  $\Delta ANG$  to  $DIST_V$  and  $ANG_V$  respectively. As the movement progresses, at each time step the distance and angle to target are decremented by an amount dictated by the currently active TVC node. If the TVC nodes have learned the correct transformation,  $DIST_V$  and  $ANG_V$  will eventually be reduced to zero, at which point the robot will stop moving and the  $\Delta DIST$  and  $\Delta ANG$  populations will also go to zero.

Figure 18 shows that the correct transformation has been learned. Figure 18a shows the robot reaching targets when the visual information is only updated every 5 seconds.<sup>2</sup> Figure 18b) shows the robot reaching targets when visual information is not available at all after the target has been acquired visually. In both parts of this figure, the actual target locations are marked with T. Note that even in the absence of visual feedback, the robot's movement is fairly accurate over significant distances. The robot is thus able to reach targets when sensory feedback is sporadic or altogether missing.

It should be noted that the weights used for these simulations had been learned on the basis of an external visual system (cf. Sec. 3.5). Using a robot-mounted visual system during learning leads to a systematic underestimate of the distance covered during a movement with a particular

---

<sup>2</sup>The robot is moving at a maximum velocity of 0.3m/s, and the longest movement in the figure takes approximately 50 seconds. Hence during the longest movement, the target is updated visually less than ten times.

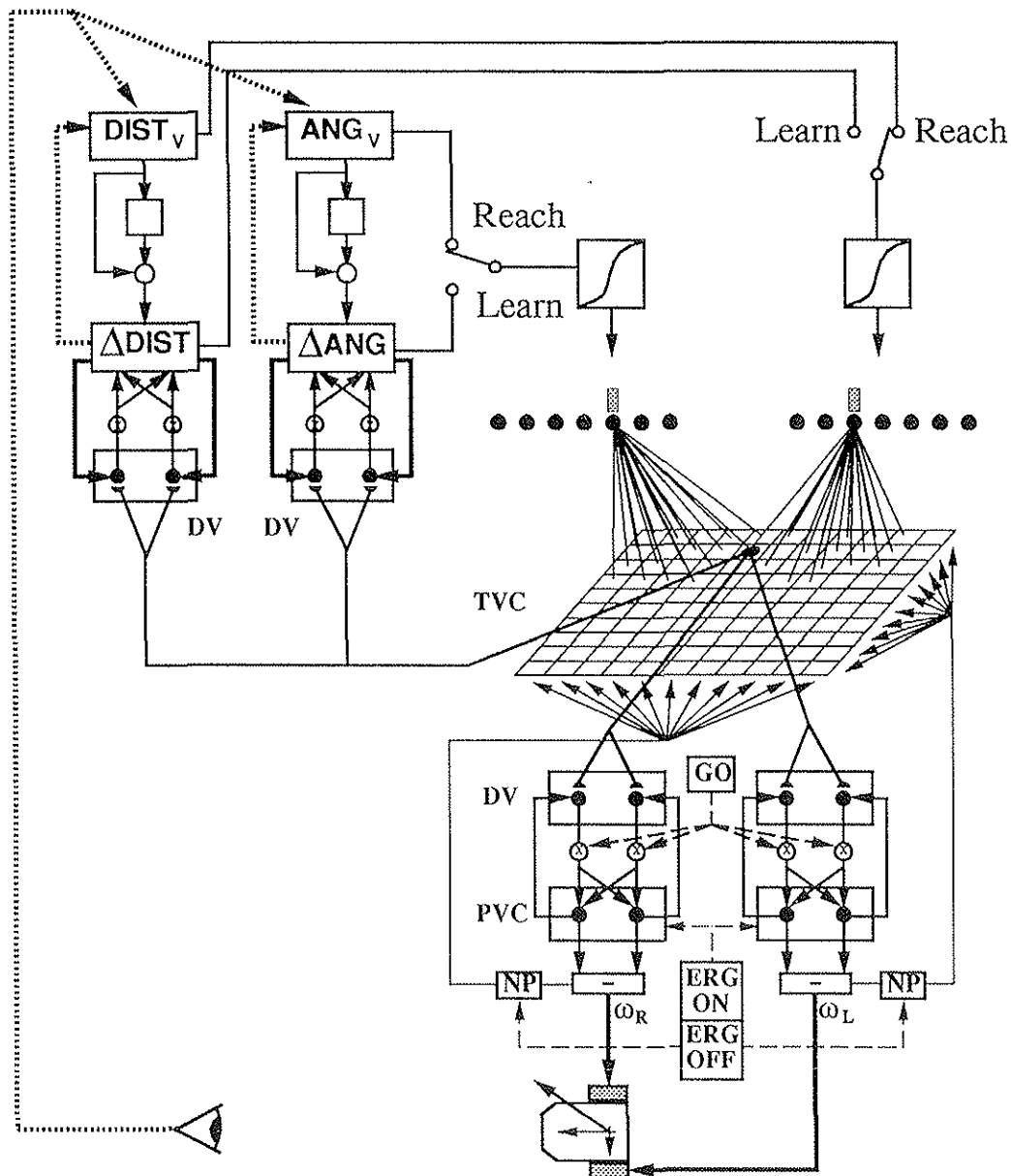


Figure 17: Diagram of the complete NETMORC architecture, including circuits for learning the robot's odometry (upper left corner). Each TVC node learns incremental distance ( $\Delta DIST$ ) and angle ( $\Delta ANG$ ) via a VAM module. During training, visual inputs ( $DIST_V$ ,  $ANG_V$ ) are used to calculate incremental displacements through a delay-and-subtract circuit. During blind reaching, incremental distance and angle are used to update  $DIST_V$  and  $ANG_V$  (thick dashed arrows). See text for details.

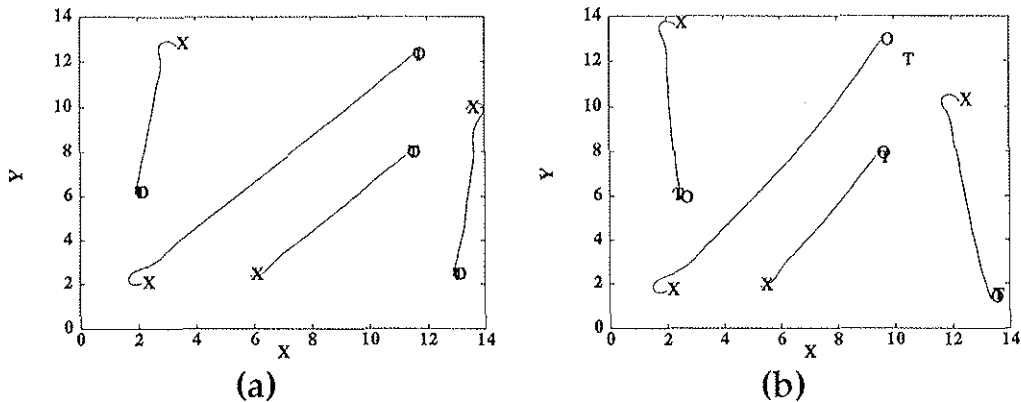


Figure 18: Spatial paths followed by the robot during reaching movements to four targets when visual information is sporadic or missing. All conventions as in Fig. 10. (a): Visual information about target is only updated every five seconds. (b): No visual information is available to the robot after the movement begins.

combination of velocities  $\omega_L$  and  $\omega_R$ . Hence a completely blind movement in this case moves the target in the correct direction, but too far. However, if the visual information is updated periodically, as in Fig. 18a, we have found this not to be a problem.

## 6 Discussion

In this article we have described NETMORC, a neural architecture for the control of a mobile robot. The main characteristic of this architecture that distinguishes it from other neural controllers is that it does not require supervision during the training phase. Supervised neural networks for robot control (Hashimoto et al., 1992; Jenkins & Yuhas, 1992; Nagata et al., 1990; Nguyen & Widrow, 1989; Pomerleau, 1991; Wilson & Rock, 1993) require user knowledge to ensure that the environment during learning is statistically representative of the environment encountered during normal operation. This problem becomes critical when it is necessary to operate in unstructured environments or when the conditions of operation change.

After an initial training phase, the NETMORC can guide the mobile robot to an arbitrary target on the basis of sensory information. The real-time error-correcting properties of the architecture allow the robot to reach targets on a given trajectory even in the presence of unexpected disturbances, such as wheel slippage or a moving target. Furthermore, the NETMORC is always capable of learning, allowing it to adapt to changes in the plant due to wear and tear that may result from normal operation, or from sudden modifications of the plant.

Another important property of NETMORC is that it can learn continuously, *i.e.*, it is not necessary to separate the learning phase from the operational phase. This property affords incremental and continuous learning, and adaptation to plant changes such as wear and tear of wheels, and other miscalibrations that may result from normal operation.

Finally, we have shown how NETMORC can also learn the robot's odometry, enabling it to update the target position on the basis of internal feedback, without further sensory information.

The ability to adapt—without supervision—to instantaneous perturbations and to statistically significant changes in the environment, makes this type of neural controller particularly useful in unknown environments or environments in which frequent perturbations might occur. For instance, it is hoped that this technology will be the basis for autonomous robots that can operate

without supervision in environments not suited for humans, such as hazardous/contaminated sites, or for space missions.

## 6.1 Comparison to other approaches

It is difficult to compare the present system to other neural network approaches to robot control, as the VAM and DIRECT models have not been utilized, to our knowledge, for similar applications. One exception is a recent article by Aguilar and Contreras-Vidal (1993), who used a modified VITE model within NAVITE, a network for navigation of a 2-DOF mobile robot. The VITE model in that article consisted of two channels used to drive two DC motors, one controlling velocity in the X direction, the other controlling velocity in the Y direction. In addition, Aguilar and Contreras-Vidal (1993) modeled the dynamics of the DC motors driving an inertial load. Furthermore, the NAVITE model makes use of sensory information to avoid obstacles and perform path planning tasks. The NAVITE thus represents an interesting complementary application of the VITE model. It may be fruitful in the future to combine NAVITE's dynamic control and path planning abilities with the low-level kinematic control of NETMORC.

Another related, but much more complex mobile robot has been described by Baloch and Waxman (1991). The Mobile Adaptive Visual Navigator (MAVIN) mobile robot includes numerous unsupervised and supervised neural networks for sensory perception, oculomotor control, pattern recognition, and reinforcement learning. MAVIN exhibits a number of complex behaviors while it learns to navigate through environments that include several objects, such as cars, tanks, and buildings. The scope of that work is thus much broader than the scope of the present model, as NETMORC would only be comparable to a single component of the multi-network MAVIN. Whereas it uses similar learning laws to achieve various types of unsupervised learning in a mobile robot, the MAVIN project has focused on problems such as the impact of mobility-dependent viewpoint variability on object recognition, and on continuous maintenance of camera orientation toward targets during approach. The MAVIN project complements the present work by demonstrating the extensibility of unsupervised learning designs to additional domains of competence critical for mobile robots.

Aside from these applications, a search of the literature suggests that most neural network applications in robotics deal with higher levels of control, such as navigation, planning, and obstacle avoidance (Antsaklis, 1990; Bekey & Goldberg, 1993; Varela & Bourgine, 1992). We are not aware of other neural network models that deal with the level of low-level control that we have described here. We are preparing a publication in which the NETMORC architecture is compared with traditional control systems on a series of low-level tasks such as those shown here.

## 6.2 Future directions

We plan to extend this research in a number of ways. First, we are carrying out a detailed analysis of the NETMORC, including proofs of stability and analysis using traditional control systems techniques (Gaudio, Zalama, & López Coronado, 1995). Part of this effort includes a comparison of the performance of NETMORC to the performance of a stable nonlinear controller (Kanayama, Kimura, Miyazaki, & Noguchi, 1990) on comparable tasks. We have also begun to implement NETMORC in hardware using a commercially available mobile robot (Zalama, 1993).

The modularity of the architecture allows its integration in other more complicated symbolic or neural architectures, which may include planification and control operations. We are developing additional circuits to allow obstacle avoidance and navigation in the presence of reduced or inaccurate exteroceptive information (Zalama, 1993).

Finally, although we have studied one particular type of mobile robot, it will be interesting to see if the ideas developed in this article can be easily generalized to other types of mobile robots.

## Reference

- Aguilar, J. M., & Contreras-Vidal, J. L. (1993). NAVITE: a neural network system for sensory-based robot navigation. Tech. rep. CAS/CNS-TR-93-005, Boston University.
- Albus, J. (1975). A new approach to manipulator control: the cerebellar model articulation controller (CMAC). *Journal of Dynamic Systems, Measurement, and Control*, *37*, 220–227.
- Antsaklis, P. J. (1990). Neural networks in control systems. *IEEE Control Systems Magazine*, *10*, 3–5. Special issue on neural networks in control systems.
- Arkin, R. C. (1990). Integrating behavioral, perceptual, and world knowledge in reactive navigation. *Robotics and Autonomous Systems*, *6*, 105–122.
- Baloch, A., & Waxman, A. (1991). Visual learning, adaptive expectations, and behavioral conditioning of the mobile robot MAVIN. *Neural Networks*, *4*, 271–302.
- Bekey, G. A., & Goldberg, K. Y. (Eds.). (1993). *Neural Networks in robotics*. Kluwer, Boston, MA.
- Brooks, R. A. (1992). Artificial life and real robots. In Varela, & Bourguine (Varela & Bourguine, 1992).
- Bullock, D., & Grossberg, S. (1988). Neural dynamics of planned arm movements: Emergent invariants and speed-accuracy properties during trajectory formation. *Psychological Review*, *95*, 49–90.
- Bullock, D., Grossberg, S., & Guenther, F. (1993). A self-organizing neural network model for redundant sensory-motor control, motor equivalence, and tool use. *Journal of Cognitive Neuroscience*, *5*, 408–435.
- Dedieu, E., & Mazer, E. (1992). An approach to sensorimotor relevance. In Varela, & Bourguine (Varela & Bourguine, 1992), pp. 88–95.
- Gaudiano, P., & Grossberg, S. (1990). A self-regulating endogenous generator of sample-and-hold random training vectors. In Caudill, M. (Ed.), *Proceedings of the International Joint Conference on Neural Networks*, pp. 213–216 Hillsdale, NJ. Erlbaum.
- Gaudiano, P., & Grossberg, S. (1991). Vector associative maps: Unsupervised real-time error-based learning and control of movement trajectories. *Neural Networks*, *4*, 147–183.
- Gaudiano, P., Zalama, E., & López Coronado, J. (1995). An unsupervised neural network for real-time, low-level control of a mobile robot: noise resistance, stability, and hardware implementation. . Submitted.
- Goodwin, G., & Sin, K. (Eds.). (1984). *Adaptive Filtering Prediction and Control*. Prentice-Hall, Englewood Cliffs, N.J.
- Greve, D., Grossberg, S., Guenther, F., & Bullock, D. (1992). Neural representations for sensory-motor control I: Head-centered 3-D target positions for opponent eye commands. *Acta Psychologica*, *82*, 115–138.

- Grossberg, S., Guenther, F., Bullock, D., & Greve, D. (1992). Neural representations for sensory-motor control II: Learning a head-centered visuomotor representation of 3-D target positions. *Neural Networks*, 6, 43–68.
- Grossberg, S. (1968). Some physiological and biochemical consequences of psychological postulates. *Proceedings of the National Academy of Sciences, USA*, 60, 758–765.
- Grossberg, S. (Ed.). (1982). *Studies of Mind and Brain: neural principles of learning, perception, development, cognition and motor control*. Reidel, Boston.
- Grossberg, S., & Kuperstein, M. (1989). *Neural dynamics of adaptive sensory-motor control* (Second edition). Pergamon Press, New York.
- Guenther, F., Bullock, D., & Greve, D. and Grossberg, S. (1993). Neural representations for sensory-motor control III: Learning a body-centered visuomotor representation of 3-D target positions. Tech. rep. CAS/CNS-TR-93-045, Boston University.
- Hashimoto, H., Kubota, T., Sato, N., & Harashima, F. (1992). Visual control of robotic manipulator based on neural networks. *IEEE Transactions on Industrial Electronics*, 39, 490–496.
- Jenkins, R. E., & Yuhas, B. P. (1992). A simplified neural-network solution through problem decomposition: the case of the truck backer-upper. *Neural Computation*, 4, 647–649.
- Jones, J. L., & Flynn, A. M. (1993). *Mobile Robots: Inspiration to Implementation*. A K. Peters, Wellesley, MA.
- Kaelbling, L. P. (1992). An adaptable mobile robot. In Varela, & Bourgine (Varela & Bourgine, 1992).
- Kanayama, Y., Kimura, Y., Miyazaki, F., & Noguchi, T. (1990). A stable tracking control method for an autonomous mobile robot. In *Proceedings of IEEE International Conference on Robotics and Automation*, pp. 384–389 St. Louis, MO. IEEE.
- Kawato, M., Fukukawa, K., & Suzuki, R. (1987). A hierarchical network model for control and learning of voluntary movement. *Biological Cybernetics*, 57, 169–185.
- Kraft, L. G., & Campagna, D. P. (1990). A comparison between CMAC neural network control and two traditional adaptive control systems. *IEEE Control Systems Magazine*, 10, 36–43.
- Kuperstein, M., & Rubenstein, J. (1989). Implementation of an adaptive neural controller for sensory-motor coordination. *IEEE Control Systems Magazine*, 9(3), 25–30.
- Kuperstein, M. (1991). INFANT neural controller for adaptive sensory-motor coordination. *Neural Networks*, 4, 131–145.
- Kuperstein, M., & Kottas, J. A. (1993). Catching INFANT: Neural controller for grasping moving objects in 3D. In *Proceedings of the World Congress on Neural Networks*, Vol. 3, pp. 135–140 Hillsdale, NJ. Lawrence Erlbaum.
- Miller, W. T., Sutton, R. S., & Werbos, P. J. (Eds.). (1990a). *Neural networks for control*. M.I.T. Press, Cambridge, Mass.

- Miller, W. T. I., Hewes, R. P., Glanz, F. H., & Kraft, L. G. I. (1990b). Real-time dynamic control of an industrial manipulator using a neural-network-based learning controller. *IEEE Transactions on Robotics and Automation*, 1, 1–9.
- Miyamoto, H., Kawato, M., Setoyama, T., & Suzuki, R. (1988). Feedback-error-learning neural network for trajectory control of a robotic manipulator. *Neural Networks*, 1, 251–265.
- Nagata, S., Sekiguchi, M., & Asakawa, K. (1990). Mobile robot control by a structured hierarchical neural network. *IEEE Control Systems Magazine*, 10, 69–76.
- Nguyen, D., & Widrow, B. (1989). The truck backer-upper: an example of self-learning in neural networks. In *Proceedings of the International Joint Conference on Neural Networks*, Vol. 2, pp. 357–363.
- Nguyen, D., & Widrow, B. (1990). Neural networks for self-learning control systems. *IEEE Control Systems Magazine*, 10, 18–23.
- Pfeifer, R., & Verschure, P. (1992). Distributed adaptive control: a paradigm for designing autonomous agents. In Varela, & Bourgine (Varela & Bourgine, 1992).
- Piaget, J. (1963). *The origin of intelligence in children*. Norton, New York.
- Pomerlau, D. A. (1991). Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3, 88–97.
- Varela, F. J., & Bourgine, P. (Eds.). (1992). *Toward a practice of autonomous systems: Proceedings of the First European Conference on Artificial Life*. MIT Press, Cambridge, MA.
- Wada, Y., & Kawato, M. (1993). A neural network model for arm trajectory formulation using forward and inverse dynamics models. *Neural Networks*, 6, 919–932.
- Widrow, B., & Stearns, S. (Eds.). (1985). *Adaptive Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ.
- Wilson, E., & Rock, S. M. (1993). Experiments in control of a free-flying space robot using fully-connected neural networks. In *Proceedings of the World Congress on Neural Networks*, Vol. 3, pp. 157–162 Hillsdale, NJ. Lawrence Erlbaum.
- Zalama, E. (1993). *Arquitectura neuronal no supervisada para el control de un robot móvil en entornos no estacionarios*. Ph.D. thesis, University of Valladolid, Spain.

## A VAM model equations

The VAM model has been described in detail by Gaudiano and Grossberg (1991). We summarize here the main equations of the VAM model for completeness. Readers are encouraged to consult the original publication for further details.

Each VAM circuit consists of an agonist-antagonist pair of channels. This opponent architecture is required in order to generate outputs that can be positive or negative. Each of the robot's driving wheels is controlled by an independent VAM module. The equations for the left and right VAM modules are identical, although in the following equations the *L* and *R* subscripts are omitted for clarity. Unless otherwise indicated, each equation below describes the behavior of variables for

the agonist (positive angular velocity) channel, labeled by the (+) superscript. The corresponding equations for the antagonist channel—omitted for simplicity—can be obtained by exchanging every (+) superscript with a (−) superscript, and *vice versa*.

### Present Velocity Command

Each PVC module contains two opponent nodes. The difference of activation in the two nodes determines the magnitude of the angular velocity signal sent to the wheel to which the PVC is connected. The behavior of one PVC node is given by:

$$\frac{dP^+}{dt} = (1 - P^+)(gV^+ + R^+) - P^+(gV^- + R^-) \quad (14)$$

During normal operation, the PVC nodes receive opponent inputs  $V^+$  and  $V^-$  from the DV nodes. These are gated by the GO signal  $g$ , which controls integration rate. If  $g = 0$  no change in PVC will take place even if the DV activations are nonzero. During the initial training phase, the PVC nodes receive random inputs  $R^+$  and  $R^-$  that cause random changes in wheel velocities. The random inputs are not gated by the GO signal.

### Difference Vector

Each DV module also contains two opponent nodes. Each node receives inhibitory inputs from the corresponding PVC node, as well as excitatory inputs from active nodes in the TVC map:

$$\frac{dV^+}{dt} = \alpha \left( -V^+ + \sum_{l,r} T_{l,r} W_{l,r}^+ - P^+ \right) \quad (15)$$

where  $\alpha$  is an integration rate. The summation shows that each DV receives inputs from all nodes  $T_{l,r}$  in the TVC map, each weighted by a modifiable connection  $W_{l,r}^+$ . Equation 15 shows that the DV activation tracks the difference between the active TVC node (gated by its weights) and the PVC node. If a single TVC node  $T_{l,r}$  is active, and if the integration rate  $\alpha$  is large, Eq. 15 is always in approximate equilibrium relative to the PVC activation, and it can thus be approximated with the linear equation  $V^+ = T_{l,r} W_{l,r}^+ - P^+$ , showing that the DV continuously tracks the difference between PVC and TVC, the latter weighted by adaptive connections.

### VAM Learning

The VAM module makes it possible for the weights connecting TVC and DV to learn a correct mapping, so that activation of a given TVC node always gives rise to the desired angular velocities. This is achieved with the following learning law:

$$\frac{dW_{lr}^+}{dt} = (-\beta W_{lr}^+ - \gamma V^+) \quad (16)$$

where  $\beta$  and  $\gamma$  are constants that determine the integration rate and the absolute magnitude of the weights. This learning rule is gated by presynaptic activation, that is, it is only applied when a TVC node is active.

The readers are referred to Gaudiano and Grossberg (1991) for a detailed discussion of this learning rule and how it can be gated during learning to generate rapid, stable learning.