An Ensemble of Cooperative Extended Kohonen Maps for Complex Robot Motion Tasks

Kian Hsiang Low
bryanlow@cs.cmu.edu
Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213-3890, U.S.A.

Wee Kheng Leow
leowwk@comp.nus.edu.sg
Department of Computer Science, National University of Singapore, Singapore 117543, Singapore

Marcelo H. Ang, Jr.
mpeangh@nus.edu.sg
Department of Mechanical Engineering, National University of Singapore, Singapore 119260, Singapore

Self-organizing feature maps such as extended Kohonen maps (EKMs) have been very successful at learning sensorimotor control for mobile robot tasks. This letter presents a new ensemble approach, cooperative EKMs with indirect mapping, to achieve complex robot motion. An indirect-mapping EKM self-organizes to map from the sensory input space to the motor control space indirectly via a control parameter space. Quantitative evaluation reveals that indirect mapping can provide finer, smoother, and more efficient motion control than does direct mapping by operating in a continuous, rather than discrete, motor control space. It is also shown to outperform basis function neural networks. Furthermore, training its control parameters with recursive least squares enables faster convergence and better performance compared to gradient descent. The cooperation and competition of multiple self-organized EKMs allow a nonholonomic mobile robot to negotiate unforeseen, concave, closely spaced, and dynamic obstacles. Qualitative and quantitative comparisons with neural network ensembles employing weighted sum reveal that our method can achieve more sophisticated motion tasks even though the weighted-sum ensemble approach also operates in continuous motor control space.

1 Introduction

Goal-directed, collision-free motion in a complex, dynamic, and unpredictable environment is an important task for an autonomous mobile robot

Neural Computation **17, 1411–1445** (2005)

operating alone or in a team. In particular, this task is widely employed in service and field robotics (Shastri, 1999), which includes sewer inspection (Hertzberg, Christaller, Kirchner, Licht, & Rome, 1998), cleaning and housekeeping (Fiorini, Kawamura, & Prassler, 2000), surveillance (Rybski, Stoeter, Gini, Hougen, & Papanikolopoulos, 2002), sensor network coverage (Howard, Matarić, & Sukhatme, 2002), search and rescue (Davids, 2002), and tour guides (Burgard et al., 1999). All these applications require the mobile robot to perform target- or goal-reaching movements while avoiding undesirable and potentially dangerous impact with obstacles or other robots on its team. The robot motion control problem can be stated succinctly as follows: Given an initial state described by the sensory input vector $\mathbf{u}(0)$ in the sensory input space $\mathcal{U}$, determine a collision-free sequence of motor control vectors $\mathbf{c}(t)$, $t = 0, \ldots, T-1$, in the motor control space $\mathcal{C}$ that moves the robot toward a desired goal state described by $\mathbf{u}(T) \in \mathcal{U}$.

Three general classes of algorithms have been investigated for learning sensorimotor control, which is required for this task: multivariate regression, reinforcement learning, and feature mapping. The first approach formulates the problem as a nonlinear multivariate regression problem and trains a multilayer perceptron (MLP) to perform continuous mapping from $\mathcal{U}$ to $\mathcal{C}$ (Pomerleau, 1991; Sharkey, 1998; Tani & Fukumura, 1994). It offers good generalization capability. However, prior to training the network, training samples have to be collected for every time step $t = 0, \ldots, T-1$ to define the quantitative error signals. This sample collection process can be very difficult and tedious, if not impossible, for a mobile robot.

The reinforcement learning approach (Kaelbling, Littman, & Moore, 1996; Sutton, 1998) circumvents the above difficulty by providing a qualitative success or failure feedback only at the end of executing the motor control sequence. It estimates how well each previously executed motor control vector $\mathbf{c}(t)$ contributes to the overall success or failure of achieving the desired goal and modifies the algorithm accordingly. The training process tends to converge slowly due to sparse reinforcements and imprecise estimate of each motor control vector's contribution.

The third approach uses a self-organizing feature Map (SOFM) (Kohonen, 2000) such as the extended Kohonen map (EKM) (Ritter & Schulten, 1986) that self-organizes to partition the continuous input (or output) space into localized regions. The generalization capability of the feature map arises from its self-organization during training such that each neuron is trained to map a localized sensory region to a desired motor control output. As compared to predefined, uniform partitioning of the feature space (Kuperstein, 1991; Schaal & Atkeson, 1998; Zalama, Gaudiano, & Coronado, 1995), self-organization may lead to better performance and learning efficiency because more neural resources are automatically allocated to frequently encountered sensory regions during learning (Martinetz, Ritter, & Schulten, 1990; Santamária, Sutton, & Ram, 1998). This approach increases the resolution of the sensory representation in the frequently encountered regions. Such a behavior is reminiscent of biological sensorimotor systems where frequently practiced movements become more fluid and accurate.

This article describes a new feature map approach to learning sensorimotor control: cooperative EKMs with indirect mapping. An indirect-mapping EKM (Low, Leow, & Ang, 2002) differs from existing direct-mapping methods (Cameron, Grossberg, & Guenther, 1998; Heikkonen & Koikkalainen, 1997; Rao & Fuentes, 1998; Ritter & Schulten, 1986; Smith, 2002; Touzet, 1997; Versino & Gambardella, 1995) in two ways:

1. Direct-mapping methods map a sensory input directly to a motor control command. In contrast, our indirect-mapping approach maps a sensory input indirectly to a motor control command through control parameters.
2. As a consequence, the indirect-mapping approach maps continuous sensory input space to continuous motor control space (see section 3.1 for detailed discussion). On the other hand, direct-mapping methods map continuous sensory input space to discrete motor control commands.

The motor control space is often discretized into a set of commands to be used by reinforcement learning algorithms (Millán, Posenato, & Dedieu, 2002; Santamária et al., 1998; Smith, 2002; Touzet, 1997), committee machines with voting schemes (Battiti & Colla, 1994; Hansen & Salamon, 1990; Kittler, Hatef, Duin, & Matas, 1998; Sharkey & Sharkey, 1997), and robot action selection mechanisms (Decugis & Ferber, 1998; Huntsberger & Rose, 1998; Maes, 1995; Rosenblatt, 1997). However, recent autonomous agent research in dynamical systems theory (Beer, 1995; Port & van Gelder, 1995) and reinforcement learning (Millán et al., 2002; Smart & Kaelbling, 2000) advocates operating in continuous motor control space, which enables our indirect-mapping method to provide finer, smoother, and more efficient motion control than does direct mapping (see section 4.1). Such a high degree of smoothness, flexibility, and precision in motion control is essential for efficiently executing complex tasks and interacting with humans.

It is well understood how a SOFM or EKM is used for learning sensorimotor control (Littmann & Ritter, 1996; Walter & Schulten, 1993). However, the nontrivial problem of combining multiple SOFMs or EKMs for sophisticated control (e.g., negotiation of unforeseen complex obstacles and cooperative multirobot tracking of moving targets; Low, Leow, & Ang, 2003) is not well studied. If solved poorly, the control outputs produced while performing a complex motion task may be unexpected or undesirable. For example, a widely used ensemble technique for motion control that combines neural network outputs via weighted sum (e.g., ensemble averaging and mixture of experts; Hashem, 1997; Haykin, 1999; Jacobs, 1995) causes the robot to be trapped easily by unforeseen, complex obstacles. This navigation issue is central to the robotics community as it is often encountered during robot motion in a real-world environment (Kim & Khosla, 1992; Koren & Borenstein, 1991; Rimon & Koditschek, 1992). Note that such a problem will arise even when SOFMs or EKMs are utilized in the weighted-sum ensemble (see section 4.2).

To solve this robot motion problem, we propose a new ensemble approach called cooperative EKMs. The cooperation and competition of multiple EKMs (Low, Leow, & Ang, 2003) that self-organize in the same manner can enable a nonholonomic mobile robot to negotiate unforeseen, concave, and closely spaced obstacles (see section 4.2). In contrast, a robot controlled by weighted-sum ensemble (Low, Leow, & Ang, 2002) may fail in such tasks even though the networks also use continuous motor control space (see section 4.2). Before proceeding to the details of cooperative EKMs, we will discuss some related work.

2 Related Work

In an MLP, all the training data are used to fit a single global model or representation. Therefore, during learning, all the network weights are susceptible to negative interference that may arise due to dynamically changing data distributions (Schaal & Atkeson, 1998). On the other hand, an EKM fits localized regions of data rather than the entire region of interest into local models or representations, thus localizing the effects of interference. Consequently, learning of a single new training datum affects fewer network weights in an EKM than in a MLP (Atkeson, Moore, & Schaal, 1997; Martinetz et al., 1990). The cost of training in an EKM is kept small by imposing a topology among the neurons such that each learning step involves a subset of neighboring neurons. Initially, the subsets are chosen large, resulting in rapid learning of the coarse sensorimotor mapping. As learning progresses, the size of the subsets is gradually reduced to refine the mapping more and more locally. This strategy allows computationally efficient and accurate training of many neurons and facilitates scaling up the EKM to a larger number of neurons for improved accuracy. An EKM also uses a smaller proportion of network weights for motor control prediction and has been reported to achieve more precise robot positioning than an MLP (Gorinevsky & Connolly, 1994; Jansen, van der Smagt, & Groen, 1995; van der Smagt, Groen, & van het Groenewoud, 1994). However, like other local model networks, an EKM suffers from the curse of dimensionality. That is, the proportion of training data lying within a fixed-radius neighborhood of a point decreases exponentially with an increasing number of dimensions of the input space.

Basis function network (BFN) is another type of local model network like EKM. However, it is architecturally different from EKM in that each incoming sensory input is reduced to activation strengths by basis functions, which are linearly mapped to a corresponding control output. To do so, the output weights of all BFN neurons are required in predicting the target-reaching motion. In contrast, the EKM uses only the winning neuron's output weights to map each sensory input to a control output (see section 3.1). During network learning, BFN updates all its output weights with each training datum. On the other hand, only the output weights of the winning neuron and its neighbors in the EKM are updated (see section 3.5). As such, BFN may experience much more interference during online learning than an EKM would.

Our indirect-mapping EKM resembles the EKM models of (Littmann and Ritter (1996) and Walter and Schulten (1993), which utilize locally linear mappings. In their models, each neuron stores both the motor control vector and the matrix of motor control parameters as output weights (see equation 3.1). On the other hand, each neuron in the indirect-mapping EKM stores only the matrix of motor control parameters. In the context of our article, their EKM models and indirect-mapping EKM, respectively, use 1800 and 1350 parameters in a network of 15 × 15 neurons (see Table 1). The extra parameters employed by their models are not necessary in achieving good target-reaching performance, as demonstrated by the indirect-mapping EKM in this letter (see section 4.1). Furthermore, we have shown that training the control parameters with recursive least squares enables faster convergence and better performance compared to gradient descent. Their EKM models (Littmann & Ritter, 1996; Walter & Schulten, 1993) have used gradient descent only to learn the control parameters.

It is typical for a robot to require a very large number of training data for accurate sensorimotor learning. The collection of these data for off-line training is a very difficult and tedious, if not impossible, task. Therefore, online learning is preferred over off-line to eliminate the need to store these data and avoid running complex batch training algorithms such as support vector machines (Schaal, Atkeson, & Vijayakumar, 2002). Hence, our focus in this article is on online learning.

Walter and Ritter (1996) have proposed an ensemble of multiple SOFMs called hierarchical PSOM (parameterized self-organizing maps) for learning sensorimotor control of the robot arm under different system contexts. PSOM is a variant of SOFM that can learn with a small set of training data and still achieve good accuracy. To be able to do so, PSOM performs interpolation on a continuous mapping manifold using a small set of predefined, data-independent basis functions and weight vectors constructed directly from the data samples. To ensure good interpolation, the data samples have to be topologically ordered to form the weight vectors. Furthermore, the minimization of the distance function in SOFM algorithm (see equation 3.2) turns into a continuous search problem for PSOM due to its continuous manifold. For hierarchical PSOM, each PSOM is trained separately, resulting in different weight values for different PSOMs. In contrast, for our cooperative EKMs, all EKMs are trained simultaneously to obtain the same input weight values (see section 3.5). Moreover, training of our EKMs is performed online rather than off-line, as is the case for PSOM.

In the absence of precise quantitative error signals for training, reinforcement learning algorithms can be used if qualitative feedback signals are available. Nevertheless, they suffer from problems of generalization and continuity. Many reinforcement learning methods encode discrete sensory states and motor commands (Dietterich, 2000; Mahadevan & Connell, 1992; Rohanimanesh & Mahadevan, 2003), which cannot apply directly to the continuous sensorimotor domains of the real-world control tasks. A priori discretization of the continuous space may introduce hidden states and weak generalization, if done poorly. By combining with function approximators (e.g., MLP or feature map) that are capable of generalizing across continuous sensory input and motor control output spaces (Baird, 1995; Gross, Stephan, & Krabbes, 1998; Millán et al., 2002; Santamária et al., 1998; Smart & Kaelbling, 2000; Smith, 2002; Touzet, 1997), this limitation can be overcome. However, generalizing with function approximators does not guarantee that the algorithms will learn to produce continuous motor commands that vary smoothly and accurately in response to continuous changes in sensory state. In effect, some reinforcement learning algorithms (Millán et al., 2002; Santamária et al., 1998; Smith, 2002; Touzet, 1997) that are combined with function approximators map from continuous sensory input space to discrete motor control commands, which is exactly what the direct-mapping EKM does (see section 3.1). The drawbacks of such a continuity problem will be demonstrated in section 4.1. To resolve this problem, some methods (Baird, 1995; Gross et al., 1998; Smart & Kaelbling, 2000) map to continuous motor control space but are burdened by very slow iterative search for the optimal action.

3 Ensemble of Cooperative EKMs

3.1 Overview. An EKM is a neural network that extends Kohonen's (2000) SOFM. Its self-organization of the input space is similar to Voronoi tessellation such that each tessellated region is encoded by the input weights of an EKM neuron. In addition to encoding a set of input weights that self-organize the sensory input space, the EKM neurons also produce outputs that vary with the incoming sensed inputs. The EKMs described in this article adopt an egocentric representation of the sensory input vector: $\mathbf{u}(t) = \{\alpha, d\}^T$, where α and d are the direction and the distance of a target location relative to the robot's current location and heading. At the goal state at time T, $\mathbf{u}(T) = (\alpha, 0)^T$ for any α.

In many proposed EKMs (Cameron et al., 1998; Heikkonen & Koikkalainen, 1997; Rao & Fuentes, 1998; Ritter & Schulten, 1986; Smith, 2002; Touzet, 1997; Versino & Gambardella, 1995), each sensory input $\mathbf{u}$ is mapped directly to a motor control command $\mathbf{c}$. In such a direct-mapping EKM (see Figure 1A), each neuron i has a sensory weight vector $\mathbf{w}_i = (\alpha_i, d_i)^T$ that encodes a tessellated region in $\mathcal{U}$ centered at $\mathbf{w}_i$. It also

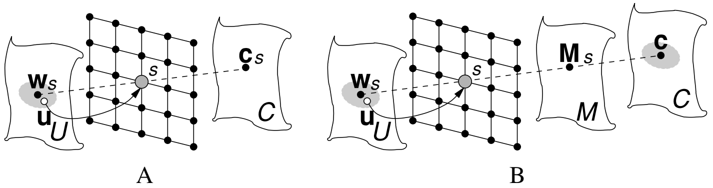

Figure 1: EKM architectures. (A) Neurons of a direct-mapping EKM map the sensory input space $\mathcal{U}$ directly to discretized points in the motor control space $\mathcal{C}$. (B) Neurons of an indirect-mapping EKM map the sensory input space $\mathcal{U}$ indirectly to the continuous motor control space $\mathcal{C}$ through the control parameter space $\mathcal{M}$. It resembles the EKM model of Walter and Schulten (1993), which stores both motor control vectors and matrices of control parameters as output weights. The indirect-mapping EKM stores only matrices of control parameters as output weights.

has a weight vector $\mathbf{c}_i$ that encodes the motor control outputs produced by the neuron. With an incoming sensory input $\mathbf{u}$, the winning neuron s is determined such that its sensory weight vector $\mathbf{w}_s$ is nearest to $\mathbf{u}$ (see equation 3.2). This winning neuron s outputs its motor control vector $\mathbf{c}_s$ to move the robot (see Figure 1A). Note that any incoming sensory input $\mathbf{u}$ that lies within the tessellated region encoded by $\mathbf{w}_s$ will produce the same motor control vector $\mathbf{c}_s$.

If sensorimotor control is a linear problem, then the motor control vector $\mathbf{c}$ would be related to the sensory input vector $\mathbf{u}$ by the linear equation

$$\mathbf{c} = \mathbf{M}\mathbf{u}, \tag{3.1}$$

where $\mathbf{M}$ is a matrix of motor control parameters. The control problem would be reduced to one of determining $\mathbf{M}$ from the training samples.

In practice, however, sensorimotor coordination is typically a nonlinear problem because a real motor takes a finite but nonzero amount of time to accelerate or decelerate in order to change speed. This problem is exacerbated in nonholonomic robots. A nonholonomic robot has restrictions in the way it can move due to kinematic or dynamic constraints such as limited turning abilities or momentum at high velocities (e.g., a car) (Arkin, 1998). Hence, a nonholonomic robot is much harder to control and to achieve smooth motion than a holonomic robot (Russell & Norvig, 1995).

To solve the nonlinear problem, our indirect-mapping EKM (Low, Leow, & Ang, 2002) is trained to partition the sensory input space $\mathcal{U}$ into locally linear regions. Each neuron i in the EKM has a sensory weight vector $\mathbf{w}_i$ similar to that of a neuron in the direct-mapping EKM. However, unlike the direct-mapping approach, the output weights of neuron i represent control parameters $\mathbf{M}_i$ in the parameter space $\mathcal{M}$ (see Figure 1B) instead of the motor

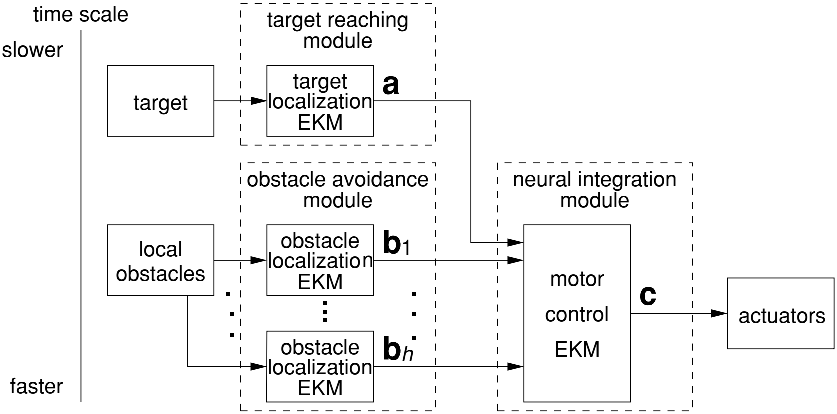

Figure 2: Framework of cooperative EKMs.

control vector $\mathbf{c}_i$. The control parameter matrix $\mathbf{M}_i$ is mapped to the actual motor control vector $\mathbf{c}$ by the linear model of equation 3.1.

To elaborate, the direct-mapping approach maps all the sensory inputs $\mathbf{u}$ in a tessellated region in the sensory input space $\mathcal{U}$, represented by a neuron s, to the same discrete point $\mathbf{c}_s$ in the motor output space $\mathcal{C}$, that is, $\mathbf{c} = \mathbf{c}_s$. Thus, only a small number of points in $\mathcal{C}$ are represented by the neurons' outputs (i.e., $\mathcal{C}$ is very sparsely sampled). In contrast, our indirect-mapping approach maps each $\mathbf{u}$ in a local region in $\mathcal{U}$ to a different point $\mathbf{c}$ in $\mathcal{C}$ through equation 3.1. Since this mapping is linear and continuous, the indirect-mapping approach maps a region in $\mathcal{U}$ to a region in $\mathcal{C}$ (see Figure 1B). This method permits finer, smoother, and more efficient sensorimotor control of the robot's target-reaching motion compared to the direct-mapping approach (see section 4.1).

Cooperative EKMs (Low, Leow, & Ang, 2003) are implemented by connecting an ensemble of EKMs into three modules: target reaching, obstacle avoidance, and neural integration (see Figure 2). The target localization EKM in the target-reaching module is activated by the presence of a target within the robot's target-sensing range. The EKM receives a sensed target location and outputs corresponding excitatory signals to the motor control EKM in the neural integration module at and around the locations of the sensed target.

The obstacle localization EKMs in the obstacle avoidance module are activated by the presence of obstacles within the robot's obstacle-sensing range. Each EKM receives a sensed obstacle location and outputs corresponding inhibitory signals to the motor control EKM in the neural integration module at and around the locations of the sensed obstacles.

The motor control EKM in the neural integration module serves as the sensorimotor interface, which integrates the activity signals from the EKMs for cooperation and competition to produce an appropriate motor signal to the actuators. This motor signal allows a robot to approach a target and negotiate obstacles.

The cooperative EKM's framework allows the modules to operate asynchronously at different rates, which is the key to preserving reactive capabilities. For example, the target-reaching module operates at about 256 ms between servo ticks while the obstacle avoidance module can typically operate faster at intervals of 128 ms. The neural integration module is activated as and when neural activities are received.

3.2 Target Reaching. The target-reaching module uses the target localization EKM to self-organize the sensory input space $\mathcal{U}$. Each neuron i in the EKM has a sensory weight vector $\mathbf{w}_i = (\alpha_i, d_i)^T$ that encodes a region in $\mathcal{U}$ centered at $\mathbf{w}_i$. Based on each incoming sensory input $\mathbf{u}$ of the target location, the target localization EKM outputs excitatory signals to the motor control EKM in the neural integration module (see section 3.4).

3.2.1 Target Localization. The target localization EKM is activated as follows. Given a sensory input $\mathbf{u}$ of a target location:

1. Determine the winning neuron s in the target localization EKM. The winning neuron s is the one whose sensory weight vector $\mathbf{w}_s = (\alpha_s, d_s)^T$ is nearest to the input $\mathbf{u} = (\alpha, d)^T$:

   $$D(\mathbf{u}, \mathbf{w}_s) = \min_{i \in \mathcal{A}(\alpha)} D(\mathbf{u}, \mathbf{w}_i). \tag{3.2}$$

   The difference $D(\mathbf{u}, \mathbf{w}_i)$ is a weighted difference between $\mathbf{u}$ and $\mathbf{w}_i$,

   $$D(\mathbf{u}, \mathbf{w}_i) = \beta_\alpha(\alpha - \alpha_i)^2 + \beta_d(d - d_i)^2, \tag{3.3}$$

   where β_α and β_d are constant parameters. The minimum in equation 3.2 is taken over the set $\mathcal{A}(\alpha)$ of neurons encoding very similar angles as α:

   $$|\alpha - \alpha_i| \leq |\alpha - \alpha_j|, \quad \text{for each pair } i \in \mathcal{A}(\alpha),\ j \notin \mathcal{A}(\alpha). \tag{3.4}$$

   In other words, direction has priority over distance in the competition between EKM neurons. This method allows the robot to quickly orientate itself to face the target while moving toward it. An EKM contains a limited set of neurons, each of which has a sensory weight vector $\mathbf{w}_i$ that encodes a point in the sensory input space $\mathcal{U}$. The region in $\mathcal{U}$ that encloses all the sensory weight vectors of these neurons is called the local workspace $\mathcal{U}'$. Even if the target falls outside $\mathcal{U}'$, the nearest neuron can still be activated (see Figure 3A).

2. Compute output activity a_i of neuron i in the target localization EKM:

   $$a_i = G_a(\mathbf{w}_s, \mathbf{w}_i). \tag{3.5}$$

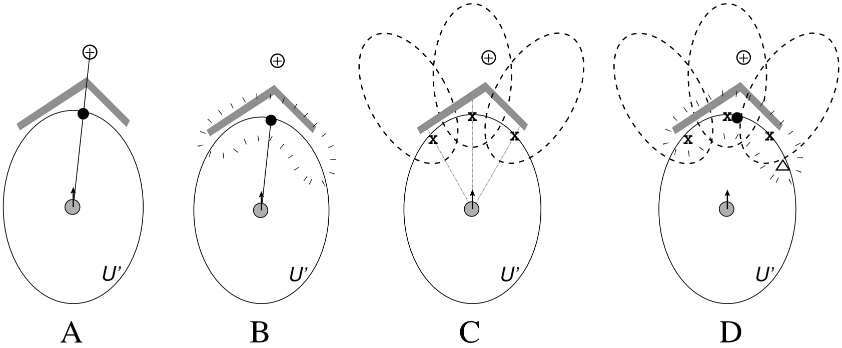

Figure 3: Conceptual description of cooperative EKMs. (A) In response to the target ⊕, the nearest neuron (black dot) in the target localization EKM (ellipse) of the robot (gray circle) is activated. (B) The activated neuron produces a target field (dotted region) in the motor control EKM. (C) Three of the robot's sensors detect obstacles and activate three neurons (crosses) in the obstacle localization EKMs, which produce the obstacle fields (dashed ellipses). (D) Subtraction of the obstacle fields from the target field results in the neuron at △ to become the winner in the motor control EKM, which moves the robot away from the obstacle.

The function G_a is an elongated gaussian:

$$G_a(\mathbf{w}_s, \mathbf{w}_i) = \exp\left(-\frac{(\alpha_s - \alpha_i)^2}{2\sigma_{a\alpha}^2} - \frac{(d_s - d_i)^2}{2\sigma_{ad}^2}\right). \tag{3.6}$$

Parameter σ_{ad} is much smaller than $\sigma_{a\alpha}$, making the gaussian distance sensitive and angle insensitive. These parameter values elongate the gaussian along the direction perpendicular to the target direction α_s (see Figure 3B). This elongated gaussian is the target field, which plays an important role in overcoming concave obstacles. The effects of these parameters on the robot's target-reaching capabilities will be examined in section 4.2.

The output activities of the neurons in the target localization EKM are aggregated in the motor control EKM to produce a motion that moves the robot toward the target. This will be explained in section 3.4. In the next section, we present the obstacle localization EKMs, which are activated in a similar manner as the target localization EKM.

3.3 Obstacle Avoidance. The obstacle avoidance module uses obstacle localization EKMs. The robot has h directed distance sensors around its body for detecting obstacles. Hence, each activated sensor encodes a fixed direction α_j and a variable distance d_j of the obstacle relative to the robot's heading and location. Each sensor's input $\mathbf{u}_j = (\alpha_j, d_j)^T$ induces an obstacle localization EKM. Note that each distance sensor (e.g., laser) can only reflect the nearest obstacle in its sensing direction. Hence, the number of obstacle localization EKMs that are activated does not depend on the number of obstacles but, rather, on the number of distance sensors. The obstacle localization EKMs have the same number of neurons and input weight values as the target localization EKMs; that is, each neuron i in the obstacle localization EKM has the same input weight vector $\mathbf{w}_i$ as the neuron i in the target localization EKM. The EKM's output inhibitory signals to the motor control EKM in the neural integration module (see section 3.4).

3.3.1 Obstacle Localization. The obstacle localization EKMs are activated as follows: For each sensory input $\mathbf{u}_j$, $j = 1, \ldots, h$ (i.e., h distance sensors):

1. Determine the winning neuron s in the jth obstacle localization EKM. The obstacle localization EKM is activated in the same manner as step 1 of target localization (see section 3.2).
2. Compute output activity b_i of neuron i in the jth obstacle localization EKM:

$$b_i = G_b(\mathbf{w}_s, \mathbf{w}_i), \tag{3.7}$$

where

$$\begin{aligned} G_b(\mathbf{w}_s, \mathbf{w}_i) &= \exp\left(-\frac{(\alpha_s - \alpha_i)^2}{2\sigma_{b\alpha}^2} - \frac{(d_s - d_i)^2}{2\sigma_{bd}^2(d_s, d_i)}\right) \\ \sigma_{bd}(d_s, d_i) &= \begin{cases} 2.475 & \text{if } d_i \geq d_s \\ 0.02475 & \text{otherwise.} \end{cases} \end{aligned} \tag{3.8}$$

The function G_b is a gaussian stretched along the obstacle direction α_s so that motor control EKM neurons beyond the obstacle locations are also inhibited to indicate inaccessibility (see Figure 3C). If no obstacle is detected, $G_b = 0$. In the presence of an obstacle, the neurons in the obstacle localization EKMs at and near the obstacle locations will be activated to produce obstacle fields. The neurons nearest to the obstacle locations have the strongest activities. The effects of the parameters σ_{bd} and $\sigma_{b\alpha}$ on the robot's obstacle avoidance capabilities will be investigated in section 4.2.

3.4 Neural Integration and Motor Control. The neural integration module uses a motor control EKM to integrate the activities from the neurons in the target and obstacle localization EKMs. The motor control EKM has the same number of neurons and input weight values as the target and robot localization EKMs.

3.4.1 Neural Integration. The neural integration is performed as follows:

1. Compute activity e_i of neuron i in the motor control EKM,

$$e_i = a_i - \sum_{j=1}^{h} b_{ji}, \tag{3.9}$$

   where a_i is the excitatory input from neuron i of the target localization EKM (see section 3.2) and b_{ji} is the inhibitory input from neuron i of the jth obstacle localization EKM (see section 3.3).

2. Determine the winning neuron k in the motor control EKM. Neuron k is the one with the largest activity:

$$e_k = \max_i e_i. \tag{3.10}$$

3.4.2 Motor Control. The motor control EKM also has a set of output weights, which encode the outputs produced by the neuron. However, unlike existing direct-mapping methods (Cameron et al., 1998; Heikkonen & Koikkalainen, 1997; Rao & Fuentes, 1998; Ritter & Schulten; 1986; Smith, 2002; Touzet, 1997; Versino & Gambardella, 1995), the output weights of neuron i of the motor control EKM represent control parameters $\mathbf{M}_i$ in the parameter space $\mathcal{M}$ instead of the actual motor control vector (see Figure 1). The control parameter matrix $\mathbf{M}_i$ is mapped to the actual motor control vector $\mathbf{c}$ by a linear model (see equation 3.11).

With indirect-mapping EKM, motor control is performed as follows: Compute motor control vector $\mathbf{c}$,

$$\mathbf{c} = \begin{cases} \mathbf{M}_k \mathbf{u} & \text{if } |\mathbf{M}_k \mathbf{u}| \leq \mathbf{c}^* \text{ and } k = s \\ \mathbf{M}_k \mathbf{w}_k & \text{otherwise,} \end{cases} \tag{3.11}$$

where s is the winning neuron in the target localization EKM, and $\mathbf{M}_k$ and $\mathbf{w}_k$ are, respectively, the control parameter matrix and sensory weight vector of the winning neuron k in the motor control EKM (step 2 of Neural Integration). The constant vector $\mathbf{c}^*$ denotes the upper limit of physically realizable motor control signal. For instance, for the Khepera robots, $\mathbf{c}$ consists of the motor speeds v_l and v_r of the robot's left and right wheels. In this case, we define $\mathbf{c} \leq \mathbf{c}^*$ if $v_l \leq v_l^*$ and $v_r \leq v_r^*$. Note that if $\mathbf{c}$ is beyond $\mathbf{c}^*$, simply saturating the wheel speeds does not work. For example, if the target is far away and not aligned with the robot's heading, then saturating both wheel speeds only moves the robot forward. Without correcting the robot's heading, the robot will not be able to reach the target. Hence, the winning neuron's input weights $\mathbf{w}_k$ are used to generate the physically realizable motor control output. This motor control would be the best substitution for the sensory input $\mathbf{u}$ because $\mathbf{w}_k$ is closest to $\mathbf{u}$ compared to other weights $\mathbf{w}_i$, $i \neq k$.

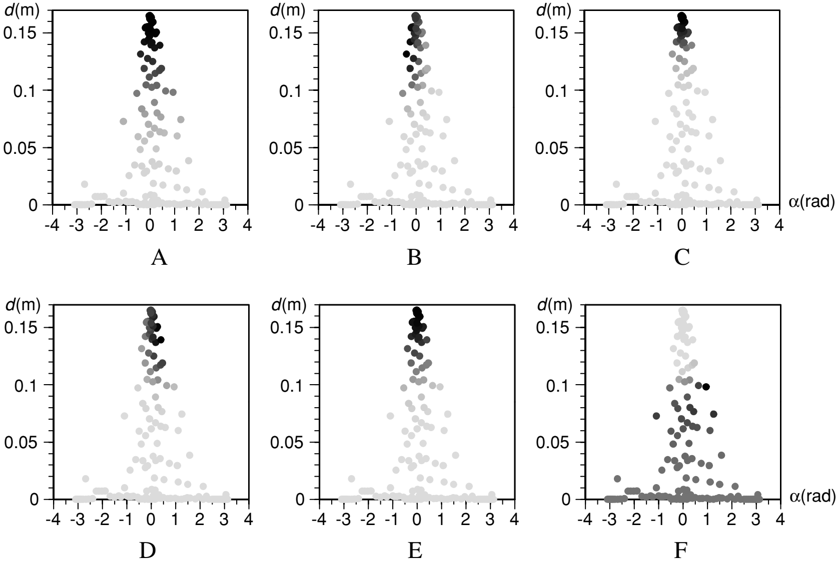

Figure 4: Output activities of neurons in (A) target localization EKM, (B) obstacle localization EKM activated by distance sensor at $-\pi/6$ radian, (C) obstacle localization EKM activated by distance sensor at 0 radian, (D) obstacle localization EKM activated by distance sensor at $\pi/6$ radian, (E) obstacle localization EKMs combined, and (F) localization EKMs combined during neural integration. Each dot denotes the sensory weights $\mathbf{w}_i = (\alpha_i, d_i)^T$ of a neuron. A darker dot implies that the neuron has a stronger output activity. A lighter dot implies the opposite.

In activating the motor control EKM (see Figure 3D), the obstacle fields are subtracted from the target field (see equation 3.9). If the target lies within the obstacle fields, the activation of the motor control EKM neurons close to the target location will be suppressed. Consequently, another neuron at a location that is not inhibited by the obstacle fields becomes most highly activated (see Figure 3D). This neuron produces a control parameter that moves the robot away from the obstacle. While the robot moves around the obstacle, the target and obstacle localization EKMs are continuously updated with the current locations and directions of the target and obstacles. Their interactions with the motor control EKM produce fine, smooth, and accurate motion control of the robot to negotiate the obstacle and move toward the target until it reaches the goal state $\mathbf{u}(T)$ at time step T.

Figure 4 shows the output activities of the neurons in different EKMs produced in response to the environment setup depicted in Figure 3. In Figure 4A, the output activities of the neurons in the target localization EKM form the target field (see Figure 3B). Since the neuron at $d = 0.16$ m and

$\alpha = 0.1$ radian (darkest dot in Figure 4A) is closest to the target location, it is most strongly activated and thus produces the highest output activity. This neuron corresponds to the black dot in Figure 3B. Its neighboring neurons also produce relatively strong output activities to form the target field used in overcoming the concave obstacle. The obstacle localization EKMs shown in Figures 4B, 4C, and 4D are activated by distance sensors positioned at $-\pi/6$, 0, and $\pi/6$ radian, respectively. For each EKM induced by a sensor, the neuron that is closest to its sensed obstacle becomes most strongly activated. These activated neurons are at $d = 0.132$ m and $\alpha = -0.4$ radian (darkest dot in Figure 4B), $d = 0.165$ m and $\alpha = 0$ radian (darkest dot in Figure 4C), and $d = 0.139$ m and $\alpha = 0.4$ radian (darkest dot in Figure 4D). They correspond to the three crosses in Figure 3C. Figure 4E shows the combined output activities of the neurons in the obstacle localization EKMs, which form the obstacle fields (see Figure 3C). Since the target lies within the obstacle fields, the strong excitatory activities from the target localization EKM neurons that are close to the target location will be suppressed. As a result, another neuron at $d = 0.098$ m and $\alpha = 0.9$ radian (darkest dot in Figure 4F) that is not inhibited by the obstacle fields becomes most strongly activated in the motor control EKM. This neuron corresponds to $\triangle$ in Figure 3D. It produces a control parameter that enables the robot to negotiate the concave obstacle.

Recall that the various modules run asynchronously at different rates (see section 3.1). In particular, the obstacle avoidance module runs at a faster rate than the target-reaching module. During neural integration, the localization EKMs remain activated until they are updated asynchronously at the next sensing cycle. So, the motor control EKM can receive continuous inputs from the localization EKMs and is always able to produce a motor signal as and when new inputs are sensed.

3.5 Self-Organization of EKMs.

In contrast to most existing off-line learning methods (Bruske & Sommer, 1995; Gorinevsky & Connolly, 1994; Karayiannis & Mi, 1997; Moody & Darken, 1989), online learning is adopted for the EKMs. Initially, the EKMs have not been trained, and the motor control vectors $\mathbf{c}$ generated are inaccurate. Nevertheless, the EKMs self-organize, using these control vectors $\mathbf{c}$ and the corresponding robot displacements $\mathbf{v}$ produced by $\mathbf{c}$, to map $\mathbf{v}$ to $\mathbf{c}$ indirectly. Note that $\mathbf{v}$ is used as the training input rather than sensory input $\mathbf{u}$. Since the untrained EKMs produce inaccurate motor control vectors $\mathbf{c}$ in response to $\mathbf{u}$ (i.e., $\mathbf{c}$ does not move the robot to the target location specified by $\mathbf{u}$), the robot will learn the wrong sensorimotor mapping if $\mathbf{u}$ is used as the corresponding training input. On the other hand, $\mathbf{v}$ is the actual displacement that corresponds to $\mathbf{c}$. Using $\mathbf{v}$ as the training input will enable the robot to learn the correct mapping as it moves around. Hence, its sensorimotor control becomes more accurate. At this stage, the online learning just fine-tunes the indirect mapping. The self-organized learning algorithm (in an obstacle-free environment) is as follows:

Self-Organized Learning

Repeat

1. Get sensory input $\mathbf{u}$.
2. Execute target-reaching procedure, and move robot.
3. Get new sensory input $\mathbf{u}'$ and compute actual displacement $\mathbf{v}$ as a difference between $\mathbf{u}'$ and $\mathbf{u}$.
4. Use $\mathbf{v}$ as the training input to determine the winning neuron k (same as step 1 of Target Localization except that $\mathbf{u}$ is replaced by $\mathbf{v}$).
5. Adjust the weights $\mathbf{w}_i$ of neurons i in the neighborhood $\mathcal{N}_k$ of the winning neuron k toward $\mathbf{v}$,

   $$\Delta \mathbf{w}_i = \eta\, G(k,i)(\mathbf{v} - \mathbf{w}_i), \tag{3.12}$$

   where $G(k,i)$ is a gaussian function of the distance between the positions of neurons k and i in the EKM and η is a constant learning rate. This step is similar to the self-organization of Kohonen's self-organizing map.
6. Update the weights $\mathbf{M}_i$ of neurons i in the neighborhood $\mathcal{N}_k$ to minimize the error e:

   $$e = \frac{1}{2} G(k,i) \|\mathbf{c} - \mathbf{M}_i \mathbf{v}\|^2 . \tag{3.13}$$

   That is, apply a recursive stochastic approximation algorithm, which can be cast into this general form,

   $$\Delta \mathbf{M}_i = -\eta \frac{\partial e}{\partial \mathbf{M}_i} \mathbf{H}_i, \tag{3.14}$$

   where $\mathbf{H}_i$ is a weighting matrix. If $\mathbf{H}_i = \mathbf{I}$, a first-order learning method, gradient descent, is obtained from equation 3.14:

   $$\Delta \mathbf{M}_i = -\eta \frac{\partial e}{\partial \mathbf{M}_i} \mathbf{I} = \eta\, G(k,i)(\mathbf{c} - \mathbf{M}_i \mathbf{v})\mathbf{v}^T . \tag{3.15}$$

   In the case of the quadratic error function e (see equation 3.13), learning can be accelerated by a second-order learning method (Battiti, 1992). This can be achieved by setting $\mathbf{H}_i$ to $\mathbf{R}_i^{-1}$ where $\mathbf{R}_i$ is a Gauss-Newton approximation of the Hessian $\partial^2 e / \partial \mathbf{M}_i^2$. A second-order learning method, recursive least squares (Glentis, Berberidis,

& Theodoridis, 1999), is thus derived from equation 3.14 with $\eta = 1$ (optimum step size),

$$\Delta \mathbf{R}_i^{-1} = \frac{1}{\lambda}\left((1-\lambda)\mathbf{R}_i^{-1} - \frac{\mathbf{R}_i^{-1}\mathbf{v}\mathbf{v}^T\mathbf{R}_i^{-1}}{\dfrac{\lambda}{G(k,i)} + \mathbf{v}^T\mathbf{R}_i^{-1}\mathbf{v}}\right) \tag{3.16}$$

$$\Delta \mathbf{M}_i = -\frac{\partial e}{\partial \mathbf{M}_i}\mathbf{R}_i^{-1} = G(k,i)(\mathbf{c} - \mathbf{M}_i\mathbf{v})\mathbf{v}^T\mathbf{R}_i^{-1}, \tag{3.17}$$

where λ is a constant forgetting rate and $\mathbf{R}_i^{-1}$ is initialized to $\mathbf{I}$. Note that the recursive online update of $\mathbf{R}_i^{-1}$ (see equation 3.16) is obtained using matrix inversion lemma to avoid the costly matrix inversion operation (Haykin, 2002). Each update of $\mathbf{M}_i$ requires $O(n^2)$ computations and $O(n^2)$ additional memory to store $\mathbf{R}_i^{-1}$ where n is the number of dimensions in $\mathbf{v}$. In contrast, gradient descent requires $O(n)$ computations and no additional memory. The performance of these two learning methods is compared in section 4.1.

The target and obstacle localization EKMs self-organize in the same manner as the motor control EKM except that step 6 is omitted. At each training cycle, the weights of the winning neuron k and its neighboring neurons i are modified. The amount of modification is proportional to the distance $G(k,i)$ between the neurons in the EKM. The input weights $\mathbf{w}_i$ are updated toward the actual displacement $\mathbf{v}$, and the control parameters $\mathbf{M}_i$ are updated so that they map the displacement $\mathbf{v}$ to the corresponding motor control $\mathbf{c}$. After self-organization has converged, the neurons will stabilize in a state such that $\mathbf{v} = \mathbf{w}_i$ and $\mathbf{c} = \mathbf{M}_i\mathbf{v} = \mathbf{M}_i\mathbf{w}_i$. For any winning neuron k, given that $\mathbf{u} = \mathbf{w}_k$, the neuron will produce a motor control output $\mathbf{c} = \mathbf{M}_k\mathbf{w}_k$, which yields a desired displacement of $\mathbf{v} = \mathbf{w}_k$. If $\mathbf{u} \neq \mathbf{w}_k$ but close to $\mathbf{w}_k$, the motor output $\mathbf{c} = \mathbf{M}_k\mathbf{u}$ produced by neuron k will still yield the correct displacement if linearity holds within the input region that activates neuron k. Thus, given enough neurons to produce an approximate linearization of the sensory input space $\mathcal{U}$, indirect-mapping EKM can produce finer and smoother motion control than direct-mapping EKM, as shown in section 4.1.

4 Experiments and Discussion

4.1 Online Learning of Target-Reaching Motion. This section presents a quantitative evaluation of the indirect-mapping EKM in online sensorimotor learning of the robot's target-reaching motion. For the purpose of evaluating performance, the following network architectures were compared:

1. B15: BFN with 15×15 neurons
2. D15: direct-mapping EKM with 15 × 15 neurons
3. G9: indirect-mapping EKM with 9 × 9 neurons trained by gradient descent
4. G12: indirect-mapping EKM with 12 × 12 neurons trained by gradient descent
5. G15: indirect-mapping EKM with 15 × 15 neurons trained by gradient descent
6. R15: indirect-mapping EKM with 15 × 15 neurons trained by recursive least squares

Our implementation of BFN was similar to those proposed by Bruske and Sommer (1995), Hartman and Keeler (1991), Karayiannis and Mi (1997), and Moody and Darken (1989), except that it was trained online rather than offline. The basis function centers were trained in a similar manner as the input weights of indirect-mapping EKM (see equation 3.12). Each basis function width was updated to approach the Euclidean distance between itself and its nearest neighbor (Hartman & Keeler, 1991; Moody & Darken, 1989; Platt, 1991). The output weights were trained by gradient descent.

We also attempted to train the basis function centers with gradient descent (Ghosh & Nag, 2001; Karayiannis, 1999; Platt, 1991; Poggio & Girosi, 1990; Wettschereck & Dietterich, 1992), but learning was unsuccessful despite extensive tuning of parameters. Although the robot learned to move toward the target locations successfully, it was not able to come to a stop at these locations, even after prolonged training. One possible explanation, as detailed by Moody and Darken (1989), is that gradient descent training of the basis function centers may lead to unpredictable target-reaching motions because the centers are sometimes squeezed out of the region of input space that contain data. Furthermore, learning converges slowly due to nonlinear optimization. In contrast, the self-organization of the input space in our implemented BFN is datacentric. More neurons are committed to input regions with dense sampling of data during online learning, which improves the resolution in these regions (see section 1). Faster convergence in learning has also been reported in this case (Moody & Darken, 1989).

The tests were performed using Webots (http://www.cyberbotics.com), a 3D, kinematic, sensor-based simulator for Khepera mobile robots, which incorporates 10% white noise in its sensors and actuators. The simulator computes the trajectories and sensory inputs of a robot situated in an environment corresponding to a given physical setup. The resulting simulation allows the controller to be transferred to a real robot without changes (Michel, 2004). The simulated behaviors are very close to those of a real robot, as demonstrated in these works (Hayes, Martinoli, & Goodman, 2002; Ijspeert, Martinoli, Billard, & Gambardella, 2001; Martinoli, Ijspeert, & Mondada, 1999).

In the experiments, the neural networks were trained in a 5 m by 5 m obstacle-free environment. Each training-testing trial took 100,000 time steps, and each time step for target-reaching motion lasted 1.024 sec. During training, the input weights were initialized to correspond to regularly spaced locations in the sensory input space $\mathcal{U}$. The robot began its network training at the center of the environment, and a randomly selected sequence of targets was presented. The robot's task was to move to the targets, one at a time, and weight modification was performed at each time step after the robot had made a move. At each time interval of 10,000 steps during training, a fixed testing procedure was conducted. In each test, the robot began at the center of the environment and was presented with 50 random target locations in sequence. The robot's task was to move to each of the target locations. No training was performed during this testing phase. The training-testing trial was repeated five times and, testing performance was averaged over the five trials.

Three testing performance indices are measured in the training-testing trials. The first index is the mean positioning error E, which measures the average distance ε_i between the center of the robot and the ith target location after it has come to a stop (i.e., motor control $\mathbf{c} = \mathbf{0}$):

$$E = \frac{1}{RN} \sum_i \varepsilon_i, \tag{4.1}$$

where R is the number of trials and N is the number of testing target locations. The second index normalized time-to-target T measures how long it takes the robot to reach the target locations:

$$T = \frac{1}{RN} \sum_i \tilde{t}_i, \ \tilde{t}_i = \frac{t_i}{l_i}, \tag{4.2}$$

where t_i is the time it takes the robot to reach the ith target, l_i is the straight-line distance between targets $i-1$ and i, and $\tilde{t}_i$ is the normalized time taken to reach target i. That is, normalized time to target measures the average amount of time the robot takes to travel a distance of 1 m toward a target. The third index, mean deviation from straight-line trajectory D, measures how straight or wavy the robot's trajectory is,

$$D = \frac{1}{RN} \sum_i \tilde{\delta}_i, \ \tilde{\delta}_i = \frac{|d_i - l_i|}{l_i}, \tag{4.3}$$

where d_i is the distance traveled to reach the target location i and $\tilde{\delta}_i$ is the deviation from straight-line trajectory for target i.

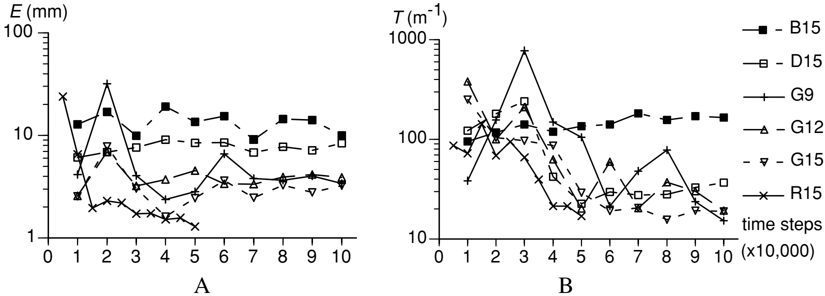

Figure 5: Performance comparison between various network architectures in (A) mean positioning error and (B) normalized time to target.

Figures 5A and 5B show, respectively, how the mean positioning error and normalized time to target decreased during the self-organized learning of various network architectures. In Figure 5A, both B15 and D15 stabilized as early as 10,000 time steps but achieved much poorer *E* performance compared to the other networks. G9, G12, G15, and R15 stabilized more gradually at about 70,000, 60,000, 50,000, and 15,000 time steps, respectively, but they could all achieve lower *E*. Notice that the larger indirect-mapping EKMs stabilized faster. To explain this counter intuitive result, note that the standard deviations of the gaussian functions in equations 3.12 and 3.15 (see section 3.5) were the same for all EKMs. That is, the proportion of neurons requiring weight updates at each time step was greater in smaller EKMs than in larger EKMs. As such, the neurons in smaller EKMs updated their weights more frequently, thus stabilizing more slowly.

While the *E* performance shows the quality of the robot positioning at the target location, the *T* and *D* performance demonstrate the quality of the robot's trajectory. We will illustrate only *T* in Figure 5B; the convergence of *D* is similar. The self-organization of D15 stabilized at about 50,000 time steps. G9, G12, and G15 stabilized at about 90,000, 70,000, and 50,000 time steps, respectively, which supported the observation that larger EKMs stabilized more quickly. R15 stabilized at 40,000 time steps, which was faster than that trained by gradient descent. Therefore, its training-testing process was stopped at 50,000 time steps, which was sufficient for its self-organized learning to stabilize. Although B15 stabilized as early as 10,000 time steps, its poorer performance, relative to the other networks, became obvious with increasing training time.

Table 1 shows the test results after training. All indirect-mapping EKMs achieved lower mean positioning errors, normalized time to target, and mean deviation from straight-line trajectory than D15 and B15. R15 achieved much lower *E* and *D* than G9, G12, and G15. Among the indirect-mapping EKMs trained by gradient descent, G12 enabled the robot

Table 1: Performance Comparison Between Networks After Training.

Network	Total Parameters	Performance Indices E (mm)	T (m^{-1})	D
B15	1350	10.02 ± 3.81	166.40 ± 28.49	0.11 ± 0.06
D15	900	8.37 ± 2.25	36.78 ± 11.11	0.18 ± 0.15
G9	486	3.40 ± 1.83	15.31 ± 4.64	0.10 ± 0.04
G12	864	3.89 ± 1.61	19.31 ± 8.47	0.06 ± 0.02
G15	1350	3.23 ± 0.66	18.96 ± 4.37	0.07 ± 0.02
R15	1350	1.29 ± 0.14	17.00 ± 2.48	0.04 ± 0.01

to travel the straightest path to stop at the target location. Reducing the number of neurons to 9×9 caused the path to be more convoluted. Increasing the number of neurons to 15×15 increased, instead of decreased, D slightly. This phenomenon could be explained by how the neurons self-organized in the sensory input space $\mathcal{U}$, which is elaborated in the next paragraph.

Neurons in G15 were self-organized into four clusters: $d = 0$ m and $\alpha = -3, 0, +3$ radian (see Figure 6C). Neurons in G9 and G12 were self-organized into two clusters only: $d = 0$ m and $\alpha = 0$ radian (see Figures. 6A and 6B). With more neurons, G15 gained the flexibility of backward motion ($\alpha = -3, +3$ radian). However, these two regions of input space were less well sampled by the neurons than the region at $\alpha = 0$ radian. As such, if a distant target appeared behind the robot with G15, its backward motion would produce a wavier path. The robot with G12 would instead turn around to face the target via the cluster at $d = 0$ m before moving forward in a much straighter path. As for G9 (see Figure 6A), since its neurons sampled the input space at $\alpha = 0$ radian more sparsely than those in G15 at $\alpha = 0, +3, -3$ radian (i.e., both forward and backward motion), it would inevitably produce a more convoluted path than G15 regardless of whether the target is in front or behind.

Table 1 also shows that smaller mean deviation did not necessarily imply shorter normalized time to target. During learning, direction had priority over distance in the competition between EKM neurons (see equation 3.4). So a larger EKM had more neurons allocated for adjusting orientation without moving long distances (i.e., $d = 0$ m cluster in Figure 6). Consequently, the robot might move short motion steps to adjust its orientation first before moving straight to the target. Therefore, its trajectory deviated less from the straight-line path.

The advantages of indirect-mapping EKMs over D15 and B15 can also be assessed from the self-organization results (see Figure 6). The neurons in the indirect-mapping EKMs cover larger areas in the sensory input space than those in D15 or B15. Moreover, they sample distances up to 0.16 m, whereas D15 and B15 neurons sample distances only up to 0.12 m. Note

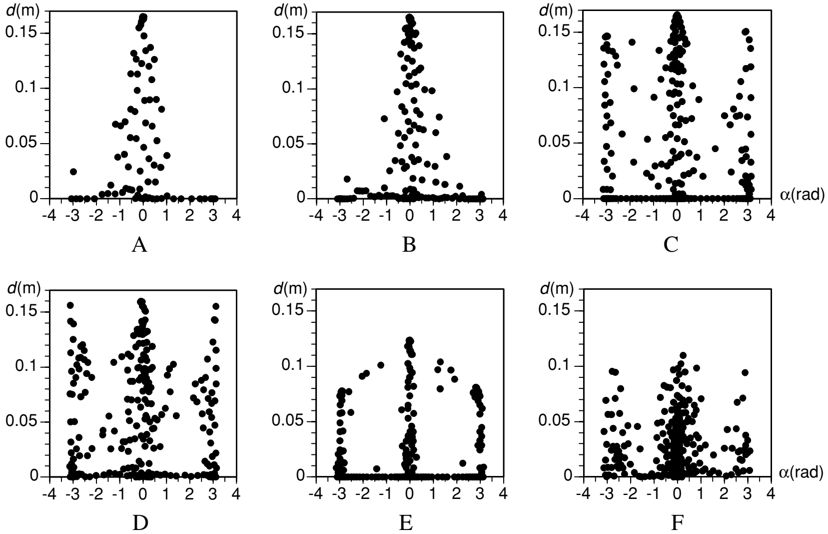

Figure 6: Self-organization results of (A) G9, (B) G12, (C) G15, (D) R15, (E) D15, and (F) B15 taken after one of the training trials. Each dot denotes the weights $\mathbf{w}_i = (\alpha_i, d_i)^T$ of a neuron.

that 0.16 m is the farthest that a Khepera robot can move in a single time step of 1 second. That is, indirect-mapping EKMs sample the sensory input space more completely than do D15 and B15 and thus produce finer, smoother, and more efficient motor control.

To determine whether there is statistically significant difference between the test results of different networks, *t*-tests were performed. In Table 2, a large value indicates that the test results between two networks are similar, that is, not significantly different (Mendenhall & Sincich, 1994). The *E* and *T* of indirect-mapping EKMs are significantly different from those of D15 and B15 because the *t*-test values are less than 0.1. However, the differences in *E* and *T* of G9, G12, and G15 are not significant. This means that G9 is sufficient for the robot to stop very close to the targets at a rate that is as fast as G12 or G15. While the difference in *E* and *D* between R15 and indirect-mapping EKMs trained by gradient descent is significant, the difference in *T* is not. The low *D* of G15 is not significantly different from that of B15. The difference in *D* of G9 from D15 and B15 is also not significant. This means that G9 achieves similar *D* performance as D15 and B15 even though it uses fewer network weights.

Often a robot is required to move through several checkpoints in a complex environment before stopping at the goal. Given that the radius of the Khepera robot is 30 mm, it is reasonable to regard the robot to have reached

Table 2: Significance Levels from *t*-Tests on Similarity in Performance Between Networks After Training.

	B15	D15	G15	G12	G9
E (mm)					
R15	0.00	0.00	0.00	0.00	0.02
G9	0.00	0.00	0.42	0.33	
G12	0.01	0.00	0.21		
G15	0.00	0.00			
D15	0.22				
T (m^{-1})					
R15	0.00	0.00	0.20	0.29	0.25
G9	0.00	0.00	0.12	0.19	
G12	0.00	0.01	0.47		
G15	0.00	0.01			
D15	0.00				
D					
R15	0.01	0.04	0.01	0.04	0.00
G9	0.43	0.16	0.09	0.02	
G12	0.04	0.06	0.09		
G15	0.12	0.09			
D15	0.19				

(and touched) a target checkpoint if the distance-to-target ε is less than 30 mm. Figure 7 illustrates the performance comparison that evaluates this target-reaching criterion after the robot has been trained.

The target-reaching probability $P(\varepsilon)$ measures the probability of the robot's reaching closer than a distance of ε (with or without stopping) from the target locations. The normalized time-to-target $T(\varepsilon)$ measures how long it takes the robot to reach closer than a distance of ε (with or without stopping) from the target locations. The mean deviation from straight-line trajectory $D(\varepsilon)$ measures how straight or wavy the robot's trajectory is.

Test results show that with indirect-mapping EKMs, the robot could get much closer to the targets with higher probability (see Figure 7A) and reach the targets much faster (see Figure 7B) than with D15 or B15. Moreover, it could travel in straighter paths (see Figure 7C) than with D15.

Table 3 shows the test results for $\varepsilon = 5$ mm. R15 outperformed G12 and G15 in $P(5)$ and G9 and G15 in $D(5)$. Among the indirect-mapping EKMs trained by gradient descent, G9 enabled a robot to reach closer than 5 mm from target locations with higher probability than G15. This could be because there were more neurons in G9 than in G15 at very small, nonzero d and $|\alpha| < 1.57$ radian in the input space. This set of neurons was responsible for moving the robot, at less than 10 mm away from the target location, forward to closer than 5 mm. As a result, a higher $P(5)$ could be achieved. It was also observed that R15, which achieved the highest $P(5)$, had more

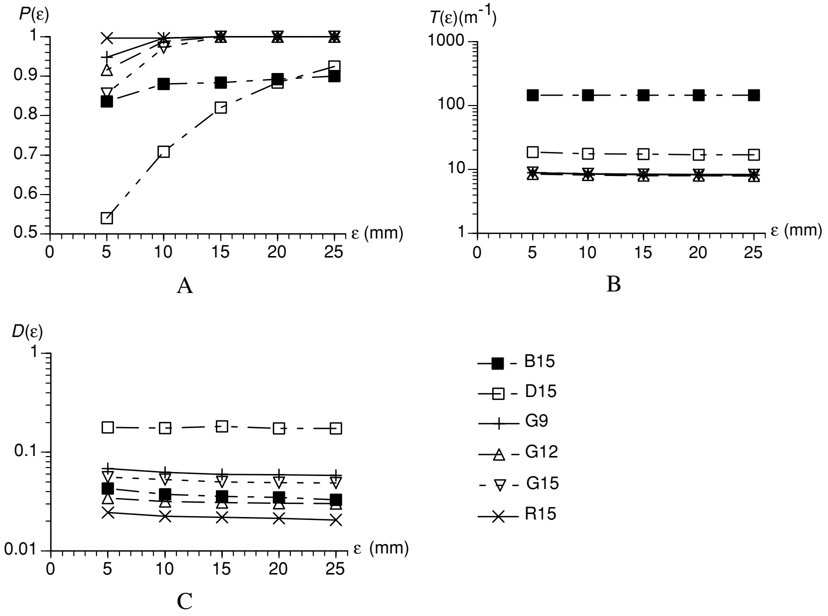

Figure 7: Performance comparison between various network architectures in (A) target-reaching probability, (B) normalized time to target, and (C) mean deviation from straight-line trajectory after training.

Table 3: Performance Comparison Between Networks.

Network	Total Parameters	Performance Indices		
		$P(5)$	$T(5)$ (m^{-1})	$D(5)$
B15	1350	0.84 ± 0.05	144.70 ± 20.16	0.04 ± 0.03
D15	900	0.54 ± 0.07	18.61 ± 1.66	0.18 ± 0.08
G9	486	0.95 ± 0.08	8.83 ± 0.96	0.07 ± 0.02
G12	864	0.92 ± 0.09	8.48 ± 0.74	0.03 ± 0.02
G15	1350	0.86 ± 0.04	8.96 ± 0.47	0.06 ± 0.01
R15	1350	1.00 ± 0.01	8.85 ± 0.35	0.03 ± 0.01

neurons than G9 in this region of the input space. G15 had many more neurons at approximately zero d than at very small, nonzero d. G12 achieved lower $D(5)$ than G9 and G15. This outcome could be justified, in a similar manner, by the explanation provided for the previous test results on D. By comparing the differences in normalized time-to-target and mean deviation between Tables 1 and 3, we could notice a greater amount of time and distance required for the robot to come to a stop.

Table 4: Significance Levels from t-Tests on Similarity in Performance at $\varepsilon = 5$ mm Between Networks.

	B15	D15	G15	G12	G9
$P(5)$					
R15	0.00	0.00	0.00	0.04	0.12
G9	0.02	0.00	0.03	0.29	
G12	0.06	0.00	0.11		
G15	0.26	0.00			
D15	0.00				
$D(5)$					
R15	0.12	0.00	0.00	0.19	0.00
G9	0.09	0.01	0.14	0.02	
G12	0.31	0.00	0.03		
G15	0.20	0.01			
D15	0.00				
$T(5)$ (m^{-1})					
R15	0.00	0.00	0.35	0.17	0.48
G9	0.00	0.00	0.40	0.27	
G12	0.00	0.00	0.13		
G15	0.00	0.00			
D15	0.00				

One other interesting comparison is the total number of parameters or weights utilized by the various networks (see Tables 1 and 3). G12 uses fewer weights than both D15 and B15 but still performs better comparatively.

Table 4 shows the t-test values at $\varepsilon = 5$ mm. While the difference in $P(5)$ and $D(5)$ between R15 and indirect-mapping EKMs trained by gradient descent is significant, the difference in $T(5)$ is not. Among the indirect-mapping EKMs trained by gradient descent, the t-tests for $P(5)$ show no significant difference between G9 and G12 and between G12 and G15. The differences in $T(5)$ between G9, G12, and G15 are also not significant.

To summarize, R15 achieved the best overall performance among the various networks, in particular, its performance in E, D, $P(5)$, and $D(5)$. Its T and $T(5)$ performance were not significantly different from those of the other indirect-mapping EKMs. Among the indirect-mapping EKMs, it stabilized most quickly. Although B15 stabilized much faster than the other networks, it produced the poorest performance in E, T, and $T(5)$. D15 offered the poorest performance in D, $D(5)$, and $P(5)$.

4.2 Neural Network Ensemble for Target-Reaching Motion with Obstacle Avoidance. This section evaluates qualitatively and quantitatively the performance of cooperative EKMs in goal-directed, collision-free robot motion in complex, unpredictable environments. The experiments

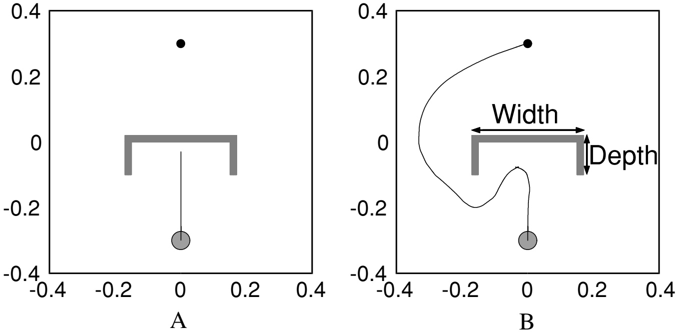

Figure 8: Negotiating unforeseen concave obstacle that was 34 cm wide and 12 cm deep. (A) The robot using command fusion was trapped, but (B) the one adopting cooperative EKMs successfully moved around the obstacle.

were also performed using Webots. Twelve directed long-range sensors were modeled around its body of radius 3 cm. Each sensor had a range of 17 cm, enabling the detection of obstacles at 20 cm or nearer from the robot's center and a resolution of 0.5 cm to simulate noise.

Two tests were performed to compare cooperative EKMs with another ensemble method (Low, Leow, & Ang, 2002). The latter approach, termed *command fusion*, linearly combines the motion control outputs, using weighted sum, of different neural networks implementing different behaviors. This is a widely used technique to integrate the motion control outputs produced by different neural networks (Hashem, 1997; Haykin, 1999; Jacobs, 1995). For our case, the target-reaching motion is produced by an indirect-mapping EKM while obstacle avoidance is performed using the method of Braitenberg's type-3C vehicle (Braitenberg, 1984). To elaborate, when the robot senses the presence of an obstacle, say, in front and on the left, the right motor will rotate backward faster than the left motor's rotation forward, thus turning the robot away from the obstacle.

For both ensemble methods, the target-reaching and obstacle avoidance modules ran at intervals of 256 ms and 128 ms, respectively. The robot's performance was assessed in an environment under two unforeseen conditions: (1) concave obstacle and (2) narrow doorway between closely spaced obstacles.

In the first test (see Figure 8), the robot fitted with command fusion got trapped by the concave obstacle (see Figure 8A). The target-reaching behavior tried to move the robot forward to reach the target while the obstacle avoidance behavior moved it backward to avoid the obstacle. The combined output cancelled each other, causing the robot to be trapped by the obstacle. The robot with cooperative EKMs could overcome the obstacle to reach the goal successfully (see Figure 8B).

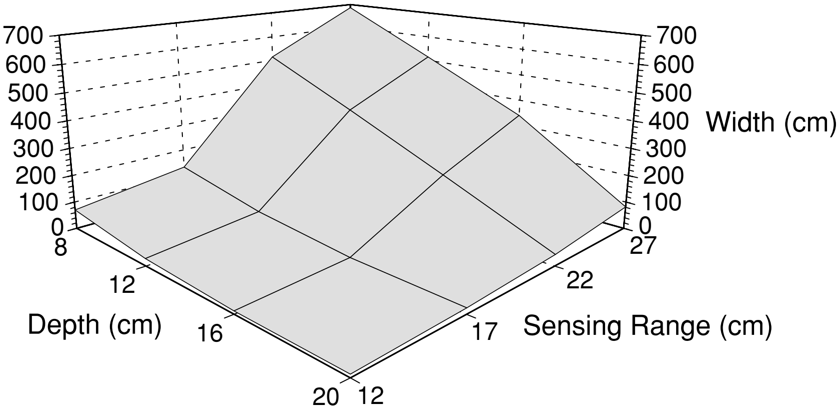

Figure 9: Maximum width of concave obstacle (see Figure 8B) that a robot with cooperative EKMs can overcome with different combinations of obstacle depths and robot-sensing ranges.

It is noted that a robot with cooperative EKMs can still get trapped if the obstacle is so concave that the obstacle fields cannot completely inhibit the neurons at or near the target location. Figure 9 shows the maximum obstacle width that a robot with cooperative EKMs can overcome with varying obstacle depths and robot-sensing ranges. Given a fixed sensing range, the maximum negotiable obstacle depth decreases with increasing width. When the sensing range increases, the robot with cooperative EKMs can negotiate an extremely wide concave obstacle if it is not too deep. Conversely, to be able to overcome a fairly deep obstacle, its width cannot be too large.

This limitation, however, does not diminish the significance of our method as it is simpler than many existing reactive robot motion methods for overcoming unforeseen concave obstacles (Lagoudakis & Maida 1999; Liu, Ang, Krishnan, & Lim, 2000; Zelek & Levine, 1996). In particular, it utilizes only local information of the target location and the unforeseen obstacles, as opposed to motion planners (Latombe, 1999) that require global knowledge of the environment to operate.

In the second test (see Figure 10), the robot endowed with command fusion could not pass through the narrow doorway between closely spaced obstacles (see Figure 10A) because its obstacle avoidance behavior counteracted the target-reaching behavior. In contrast, the robot with cooperative EKMs could always traverse through the narrow doorway to the goal successfully (see Figure 10B).

These two simple tests show that for command fusion, though each neural network proposes an action that is optimal by itself, the weighted sum of these action commands produces a combined action that may not satisfy the overall task. Cooperative EKMs, however, consider the activity signals

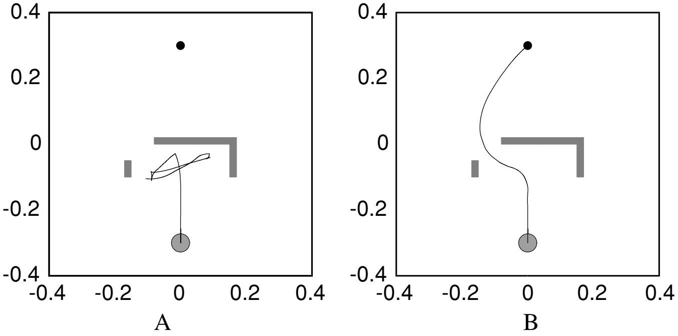

Figure 10: Passing through an unforeseen narrow doorway between closely spaced obstacles that was 86 mm wide. (A) The robot using command fusion was trapped, but (B) the one adopting cooperative EKMs successfully passed through the narrow doorway to the goal.

of each localization EKM and integrate them to determine an action that can satisfy each localization EKM to a certain degree. Such tightly coupled interaction between the localization EKMs and the motor control EKM in the cooperative EKMs framework enables the robot to achieve more complex tasks.

Recall that the standard deviations σ of the gaussian functions for the target and obstacle fields play an important role in the robot motion capabilities of cooperative EKMs (see sections 3.2 and 3.3). For the target field (see Figure 3A), $\sigma_{a\alpha}$ and σ_{ad} control the elongation of the target field perpendicular to and along the target direction, respectively. Parameters $\sigma_{b\alpha}$ and σ_{bd} achieve a similar effect for the obstacle field. For the negotiation of concave obstacles (e.g., see Figure 8B), the target field has to be considerably elongated perpendicular to the target direction. This requires a large enough $\sigma_{a\alpha}$ parameter value. However, as this value increases, the tendency of the robot moving along the shorter path to the goal via the narrow doorway (see Figure 10B) decreases and the longer detour featured in Figure 8B is increasingly preferred in the situation of Figure 10B. When $\sigma_{a\alpha}$ is large, subtraction of the obstacle field from the highly elongated target field (see Figure 3D) in the motor control EKM, rather than the neuron at the narrow doorway, may result in the neuron at the edge of the concave obstacle to be more highly activated. Nonetheless, the above two tests and the subsequent ones can be achieved by a single $\sigma_{a\alpha}$ value of 2.475. If σ_{ad} is too small, the target field may be totally suppressed by the obstacle field depending on $\sigma_{a\alpha}$. This may or may not cause the robot to be trapped in the concave obstacle since any neuron not inhibited by the obstacle field can be potentially activated. If σ_{ad} is too large, the robot may get trapped in the concave obstacle. Superposition of the fields may cause the neuron in the cavity of

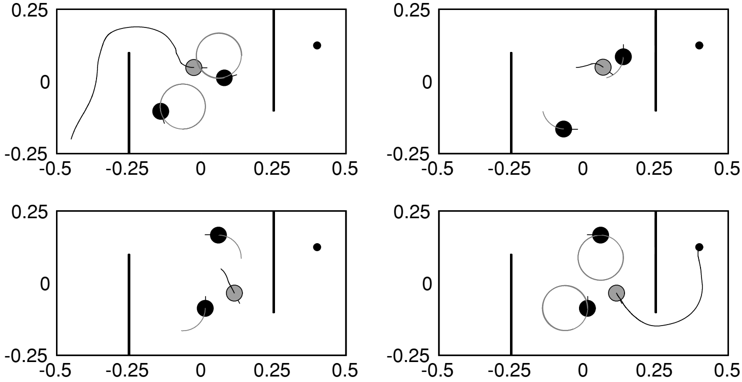

Figure 11: Motion of robot (gray) in an environment with two unforeseen obstacles (black) moving in anticlockwise circular paths. The robot could successfully negotiate past the extended walls and the dynamic obstacles to reach the goal (small black dot).

the concave obstacle, rather than the neuron at the edge of the obstacle, to be more highly activated. In all the tests, σ_{ad} is set to 0.0495.

The parameter values of $\sigma_{b\alpha}$ and σ_{bd} have to be large enough for the robot to avoid collision with obstacles as well as discriminate whether a doorway is wide enough to pass through. However, if these values are too large, the robot cannot move to target locations near the obstacles or detect the presence of narrow but traversable doorways. In all the tests, $\sigma_{b\alpha}$ is set to 0.495, while σ_{bd} uses the values given in equation 3.8. The above evaluation of the target and obstacle field parameters highlights their significance to the robot motion capabilities of cooperative EKMs. In our future work, we will consider using reinforcement learning to train the appropriate parameter values for negotiating different obstacles when the robot encounters them during motion.

The next two tests aim to demonstrate the capabilities of cooperative EKMs in performing more complex motion tasks. The environment for the first test consisted of three rooms connected by two doorways (see Figure 11). The middle room contained two obstacles moving in anticlockwise circular paths. The robot began in the left-most room and was tasked to move to the right-most room. Test results show that the robot was able to negotiate past the extended walls and the dynamic obstacles to reach the goal. Note that this target-reaching motion was completely determined by the cooperation and competition between the EKMs, and no global planning was used.

The environment for the second test consisted of three rooms connected by two doorways and some unforeseen static obstacles (see Figure 12). The robot began in the top corner of the left-most room and was tasked to move

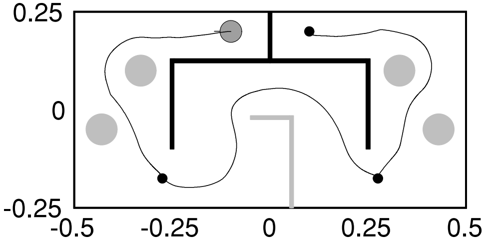

Figure 12: Motion of robot (dark gray) in a complex environment. The checkpoints (small black dots) were located at the doorways and the goal position. The robot could successfully navigate through the checkpoints to the goal by traversing between unforeseen narrowly spaced convex obstacles (light gray) in the first and the last room and overcoming an unforeseen concave obstacle (light gray) in the middle room.

into the narrow corner of the right-most room via checkpoints plotted by a planner (Low, Leow, & Ang, 2002). The robot was able to move through the checkpoints to the goal by traversing between narrowly spaced convex obstacles in the first and the last room, and overcoming an unforeseen concave obstacle in the middle room. The results of these last two tests further confirm the effectiveness of cooperative EKMs in handling complex tasks in complex, unpredictable environments.

5 Conclusion

This article presents a new approach of learning sensorimotor control for complex robot motion tasks using cooperative EKMs. Quantitative evaluation reveals that indirect-mapping EKM can produce finer, smoother, and more efficient robot motion control than other local learning methods such as direct-mapping EKM and BFN. Furthermore, training the control parameters of the indirect-mapping EKM with recursive least squares allows faster convergence and better performance than with gradient descent. The cooperation and competition of multiple EKMs enable the nonholonomic mobile robot to negotiate unforeseen concave, closely spaced, and dynamic obstacles. These tasks can easily trap robots that are controlled by neural network ensembles employing command fusion techniques. Cooperative EKMs can thus augment the reactive capabilities of an autonomous mobile robot significantly. Recently, we have enhanced cooperative EKMs further to achieve multirobot motion tasks such that multiple robots fitted with cooperative EKMs can coordinate their tracking of moving targets (see Figure 13). Qualitative and quantitative test results of the improved cooperative EKMs for multirobot tasks are presented in Low, Leow, and Ang

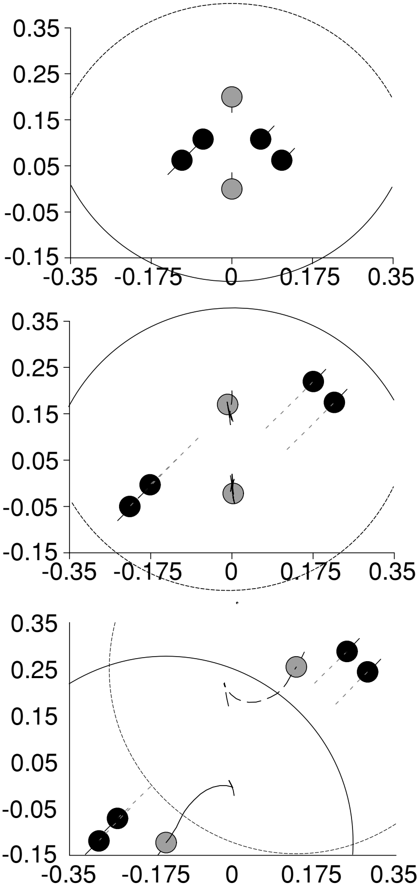

Figure 13: Cooperative tracking of moving targets. When the targets were moving out of the robots' sensory range, the two robots moved in opposite directions to track the targets. In this way, all targets could still be observed by the robots.

(2003) and Low, Leow, and Ang (2004). Our continuing research goal is to generalize this approach to other sensorimotor control problems such as those of static and mobile robot manipulators.

References

Arkin, R. C. (1998). *Behavior-based robotics*. Cambridge, MA: MIT Press.

Atkeson, C. G., Moore, A. W., & Schaal, S. (1997). Locally weighted learning. *Artificial Intelligence Review*, *11*(1–5), 11–73.

Baird, L. C. (1995). Residual algorithms: Reinforcement learning with function approximation. In *Proc. 12th International Conference on Machine Learning (ICML-95)* (pp. 30–37). San Francisco: Morgan Kaufmann.

Battiti, R. (1992). First- and second-order methods for learning: Between steepest descent and Newton's method. *Neural Comput.*, *4*(2), 141–166.

Battiti, R., & Colla, A. (1994). Democracy in neural nets: Voting schemes for classification. *Neural Networks*, *7*(4), 691–707.

Beer, R. D. (1995). A dynamical systems perspective on agent-environment interaction. *Artificial Intelligence*, *72*(1–2), 173–215.

Braitenberg, V. (1984). *Vehicles: Experiments in synthetic psychology*. Cambridge, MA: MIT Press.

Bruske, J., & Sommer, G. (1995). Dynamic cell structure learns perfectly topology preserving map. *Neural Comput.*, *7*(4), 845–865.

Burgard, W., Cremers, A. B., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W., & Thrun, S. (1999). Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, *114*(1–2), 3–55.

Cameron, S., Grossberg, S., & Guenther, F. H. (1998). A self-organizing neural network architecture for navigation using optic flow. *Neural Comput.*, *10*(2), 313–352.

Davids, A. (2002). Urban search and rescue robots: From tragedy to technology. *IEEE Intelligent Systems*, *17*(2), 81–83.

Decugis, V., & Ferber, J. (1998). Action selection in an autonomous agent with a hierarchical distributed reactive planning architecture. In *Proc. 2nd International Conference on Autonomous Agents* (pp. 354–361). New York: ACM Press.

Dietterich, T. G. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. *J. Artificial Intelligence Res.*, *13*, 227–303.

Fiorini, P., Kawamura, K., & Prassler, E. (Eds.). (2000). *Autonomous Robots, Special Issue on Cleaning and Housekeeping Robots*, *9*(3).

Ghosh, J., & Nag, A. (2001). An overview of radial basis function networks. In R. J. Howlett & L. C. Jain (Eds.), *Radial basis function networks 2: New advances in design* (pp. 1–36). New York: Physica-Verlag.

Glentis, G.-O., Berberidis, K., & Theodoridis, S. (1999). Efficient least squares adaptive algorithms for FIR transversal filtering. *IEEE Signal Processing Mag.*, *16*(4), 13–41.

Gorinevsky, D., & Connolly, T. H. (1994). Comparison of some neural networks and scattered data approximations: The inverse manipulator kinematics example. *Neural Comput.*, *6*(3), 521–542.

Gross, H.-M., Stephan, V., & Krabbes, M. (1998). A neural field approach to topological reinforcement learning in continuous action spaces. In *Proc. International Joint Conference on Neural Networks* (Vol. 3, pp. 1992–1997). New York: IEEE Press.

Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Trans. Pattern Anal. Machine Intell.*, *12*(10), 993–1001.

Hartman, E., & Keeler, D. (1991). Predicting the future: Advantages of semilocal units. *Neural Comput.*, *3*(4), 566–578.

Hashem, S. (1997). Optimal linear combinations of neural networks. *Neural Networks*, *10*(4), 599–614.

Hayes, A. T., Martinoli, A., & Goodman, R. M. (2002). Distributed odor source localization. *IEEE Sensors*, *2*(3), 260–271.

Haykin, S. (1999). *Neural networks: A comprehensive foundation* (2nd ed.). Upper Saddle River, NJ: Prentice Hall.

Haykin, S. (2002). *Adaptive filter theory* (4th ed.). Upper Saddle River, NJ: Prentice Hall.

Heikkonen, J., & Koikkalainen, P. (1997). Self-organization and autonomous robots. In O. Omidvar & P. van der Smagt (Eds.), *Neural systems for robotics* (pp. 297–337). Orlando, FL: Academic Press.

Hertzberg, J., Christaller, T., Kirchner, F., Licht, U., & Rome, E. (1998). Sewer robotics. In R. Pfeifer, B. Blumberg, J.-A. Meyer, & S. W. Wilson (Eds.), *From animals to animats 5: Proc. International Conference on Simulation of Adaptive Behavior* (pp. 427–436). Cambridge, MA: MIT Press.

Howard, A., Matarić, M. J., & Sukhatme, G. S. (2002). Network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In H. Asama, T. Arai, T. Fukuda, & T. Hasegawa (Eds.), *Distributed autonomous robotic systems 5: Proc. 6th International Symposium on Distributed Autonomous Robotic Systems* (pp. 299–308). New York: Springer.

Huntsberger, T., & Rose, J. (1998). BISMARC: A biologically inspired system for map-based autonomous rover control. *Neural Networks*, *11*(7–8), 1497–1510.

Ijspeert, A. J., Martinoli, A., Billard, A., & Gambardella, L. M. (2001). Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. *Autonomous Robots*, *11*(2), 149–171.

Jacobs, R. A. (1995). Methods for combining experts' probability assessments. *Neural Comput.*, *7*(5), 867–888.

Jansen, A., van der Smagt, P., & Groen, F. C. A. (1995). Nested networks for robot control. In A. F. Murray (Ed.), *Applications of neural networks* (pp. 221–239). Dordrecht: Kluwer.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *J. Artificial Intelligence Res.*, *4*, 237–285.

Karayiannis, N. B. (1999). Reformulated radial basis neural networks trained by gradient descent. *IEEE Trans. Neural Networks*, *10*(3), 657–671.

Karayiannis, N. B., & Mi, G. W. (1997). Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques. *IEEE Trans. Neural Networks*, *8*(6), 1492–1506.

Kim, J.-O., & Khosla, P. (1992). Real-time obstacle avoidance using harmonic potential functions. *IEEE Trans. Robot. Automat.*, *8*(3), 338–349.

Kittler, J., Hatef, M., Duin, R. P. W., & Matas, J. (1998). On combining classifiers. *IEEE Trans. Pattern Anal. Machine Intell.*, *20*(3), 226–239.

Kohonen, T. (2000). *Self-organizing maps* (3rd ed.). New York: Springer.

Koren, Y., & Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. In *Proc. IEEE International Conference on Robotics and Automation (ICRA'91)* (pp. 1394–1404). New York: IEEE Computer Society Press.

Kuperstein, M. (1991). INFANT neural controller for adaptive sensory-motor coordination. *Neural Networks*, *4*(2), 131–146.

Lagoudakis, M. G., & Maida, A. S. (1999). Robot navigation with a polar neural map: Student abstract. In *Proc. 16th National Conference on Artificial Intelligence (AAAI-99)* (p. 965). Menlo Park, CA: AAAI Press.

Latombe, J.-C. (1999). Motion planning: A journey of robots, molecules, digital actors, and other artifacts. *International Journal of Robotics Research*, *18*(11), 1119–1128.

Littmann, E., & Ritter, H. (1996). Learning and generalization in cascade network architectures. *Neural Comput.*, *8*(7), 1521–1539.

Liu, C., Ang Jr., M. H., Krishnan, H., & Lim, S. Y. (2000). Virtual obstacle concept for local-minimum-recovery in potential-field based navigation. In *Proc. IEEE International Conference on Robotics and Automation (ICRA'00)* (Vol. 2, pp. 983–988). New York: ACM Press.

Low, K. H., Leow, W. K., & Ang, Jr., M. H. (2002). A hybrid mobile robot architecture with integrated planning and control. In *Proc. 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS-02)* (pp. 219–226). New York: ACM Press.

Low, K. H., Leow, W. K., & Ang, Jr., M. H. (2003). Action selection for single- and multi-robot tasks using cooperative extended Kohonen maps. In *Proc. 18th International Joint Conference on Artificial Intelligence (IJCAI-03)* (pp. 1505–1506). New York: IEEE Press.

Low, K. H., Leow, W. K., & Ang, Jr., M. H. (2004). Task allocation via self-organizing swarm coalitions in distributed mobile sensor network. In *Proc. 19th National Conference on Artificial Intelligence (AAAI-04)* (pp. 28–33). Menlo Park, CA: AAAI Press.

Maes, P. (1995). Modeling adaptive autonomous agents. In C. G. Langton (Ed.), *Artificial life:An overview* (pp. 135–162). Cambridge, MA: MIT Press.

Mahadevan, S., & Connell, J. (1992). Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, *55*(2–3), 311–365.

Martinetz, T. M., Ritter, H. J., & Schulten, K. J. (1990). Three-dimensional neural net for learning visuomotor coordination of a robot arm. *IEEE Trans. Neural Networks*, *1*(1), 131–136.

Martinoli, A., Ijspeert, A. J., & Mondada, F. (1999). Understanding collective aggregation mechanisms: From probabilistic modelling to experiments with real robots. *Robotics and Autonomous Systems*, *29*(1), 51–63.

Mendenhall, W., & Sincich, T. (1994). *Statistics for engineering and the sciences* (4th ed.). Upper Saddle River, NJ: Prentice Hall.

Michel, O. (2004). Cyberbotics Ltd. WebotsTM: Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, *1*(1), 39–42.

Millán, J. del R., Posenato, D., & Dedieu, E. (2002). Continuous-action *Q*-learning. *Machine Learning*, *49*(2–3), 249–265.

Moody, J., & Darken, C. J. (1989). Fast learning in networks of locally-tuned processing units. *Neural Comput.*, *1*(2), 281–294.

Platt, J. (1991). A resource-allocating network for function interpolation. *Neural Comput.*, *3*(2), 213–225.

Poggio, T., & Girosi, F. (1990). Networks for approximation and learning. *Proc. IEEE*, *78*(9), 1481–1497.

Pomerleau, D. A. (1991). Efficient training of artificial neural networks for autonomous navigation. *Neural Comput.*, *3*(1), 88–97.

Port, R. F., & van Gelder, T. (1995). *Mind as motion: Explorations in the dynamics of cognition*. Cambridge, MA: MIT Press.

Rao, R. P. H., & Fuentes, O. (1998). Hierarchical learning of navigational behaviors in an autonomous robot using a predictive sparse distributed memory. *Machine Learning*, *31*(1–3), 87–113.

Rimon, E., & Koditschek, D. E. (1992). Exact robot navigation using artificial potential functions. *IEEE Trans. Robot. Automat.*, *8*(5), 501–518.

Ritter, H., & Schulten, K. (1986). Topology conserving mappings for learning motor tasks. In J. S. Denker (Ed.), *Neural networks for computing* (pp. 376–380). Snowbird, UT: American Institute of Physics.

Rohanimanesh, K., & Mahadevan, S. (2003). Learning to take concurrent actions. In S. Becker, S. Thrun, & K. Obermayer (Eds.), *Advances in neural information processing systems, 15* (pp. 1619–1626). Cambridge, MA: MIT Press.

Rosenblatt, J. K. (1997). DAMN: A distributed architecture for mobile navigation. *J. Expt. Theor. Artif. Intell.*, *9*(2–3), 339–360.

Russell, S., & Norvig, P. (1995). *Artificial intelligence: A modern approach*. Upper Saddle River, NJ: Prentice Hall,

Rybski, P. E., Stoeter, S. A., Gini, M., Hougen, D. F., & Papanikolopoulos, N. P. (2002). Performance of a distributed robotic system using shared communications channels. *IEEE Trans. Robot. Automat.*, *18*(5), 713–727.

Santamária, J. C., Sutton, R., & Ram, A. (1998). Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive Behavior*, *6*(2), 163–218.

Schaal, S., & Atkeson, C. G. (1998). Constructive incremental learning from only local information. *Neural Comput.*, *10*(8), 2047–2084.

Schaal, S., Atkeson, C. G., & Vijayakumar, S. (2002). Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence*, *17*(1), 49–60.

Sharkey, A. J. C., & Sharkey, N. E. (1997). Combining diverse neural nets. *Knowledge Engineering Review*, *12*(3), 231–247.

Sharkey, N. E. (1998). Learning from innate behaviors: A quantitative evaluation of neural network controllers. *Machine Learning*, *31*(1–3), 115–139.

Shastri, S. V. (Ed.). (1999). Field and service robotics [Special issue]. *International Journal of Robotics Research*, *18*(7).

Smart, W. D., & Kaelbling, L. P. (2000). Practical reinforcement learning in continuous spaces. In *Proc. 17th International Conference on Machine Learning (ICML-00)* (pp. 903–910). San Francisco: Morgan Kaufmann.

Smith, A. J. (2002). Applications of the self-organising map to reinforcement learning. *Neural Networks*, *15*(8–9), 1107–1124.

Sutton, R. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Tani, J., & Fukumura, N. (1994). Learning goal-directed sensory-based navigation of a mobile robot. *Neural Networks*, *7*(3), 553–563.

Touzet, C. (1997). Neural reinforcement learning for behavior synthesis. *Robotics and Autonomous Systems*, *22*(3–4), 251–281.

van der Smagt, P., Groen, F. C. A., & van het Groenewoud, F. (1994). The locally linear nested network for robot manipulation. In *Proc. International Conference on Neural Networks* (Vol. 5, pp. 2787–2792). New York: IEEE Press.

Versino, C., & Gambardella, L. M. (1995). Learning the visuomotor coordination of a mobile robot by using the invertible Kohonen map. In J. Mira & F. Sandoval

(Eds.), *Proc. International Workshop on Artificial Neural Networks* (pp. 1084–1091). Berlin: Springer.

Walter, J., & Ritter, H. (1996). Investment learning with hierarchical PSOM. In D. Touretzky, M. Mozer, & M. Hasselmo (Eds.), *Advances in neural information processing systems, 8* (pp. 570–576). Cambridge, MA: MIT Press.

Walter, J. A., & Schulten, K. J. (1993). Implementation of self-organizing neural networks for visuo-motor control of an industrial robot. *IEEE Trans. Neural Networks*, *4*(1), 86–95.

Wettschereck, D., & Dietterich, T. (1992). Improving the performance of radial basis function networks by learning center locations. In J. E. Moody, S. J. Hanson, & R. P. Lipmann (Eds.), *Advances in neural information processing systems, 4* (pp. 1133–1140). San Mateo, CA: Morgan Kaufmann.

Zalama, E., Gaudiano, P., & Coronado, J. L. (1995). A real-time, unsupervised neural network for the low-level control of a mobile robot in a non-stationary environment. *Neural Networks*, *8*(1), 103–123.

Zelek, J. S., & Levine, M. D. (1996). SPOTT: A mobile robot control architecture for unknown or partially known environments. In I. Nourbakhsh (Ed.), *Planning with incomplete information for robot problems: Papers from the AAAI Spring Symposium* (pp. 129–140). Toronto: AAAI Press.

Received March 9, 2004; accepted November 19, 2004.