Enhancing the Reactive Capabilities of Integrated Planning and Control with Cooperative Extended Kohonen Maps

Kian Hsiang Low†, Wee Kheng Leow†, and Marcelo H. Ang Jr.‡

†Dept. of Computer Science, and ‡Dept. of Mechanical Engineering, National University of Singapore, Singapore

Email: †ieslkh, leowwk@comp.nus.edu.sg, ‡mpeangh@nus.edu.sg

***Abstract*— Despite the many significant advances made in robot motion research, few works have focused on the tight integration of high-level deliberative planning with reactive control at the lowest level. In particular, the real-time performance of existing integrated planning and control architectures is still not optimal because the reactive control capabilities have not been fully realized. This paper aims to enhance the low-level reactive capabilities of integrated planning and control with Cooperative Extended Kohonen Maps for handling complex, unpredictable environments so that the workload of the high-level planner can be consequently eased. The enhancements include fine, smooth motion control, execution of more complex motion tasks such as overcoming unforeseen concave obstacles and traversing between closely spaced obstacles, asynchronous execution of behaviors, and removal of redundant overhead in behavioral modules.**

I. Introduction

Robot motion research has proceeded along two dichotomous streams: high-level deliberative planning and low-level reactive control. Deliberative planning uses a world model to generate an optimal sequence of collision-free actions that can achieve a globally specified goal in a complex static environment. However, in a dynamic environment, unforeseen obstacles may obstruct the action sequence, and replanning to react to these situations can be too computationally expensive. Reactive control directly couples sensed data to appropriate actions. It allows the robot to respond robustly and timely to unexpected obstacles and environmental changes but may be trapped by them. These two paradigms have their own strengths and weaknesses, which are rather complementary. Their union into one coherent integrated framework can potentially mitigate their respective drawbacks and yield the best of both approaches.

Nevertheless, developing such a unification methodology is non-trivial and recent proposals lack the capacity for real world use. In particular, the real-time performance of existing integrated planning and control architectures is still not optimal because the reactive control capabilities have not been fully realized. Often, the workload of the high-level planner far exceeds that of the low-level reactive controller [5], [7], [10], [12]. The planner produces the exact motion path with detailed sequence of actions to be executed by the actuators. The reactive controller performs only a single task, i.e., simple local obstacle avoidance, by correcting the course of action.

Hence, the work presented in this paper aims to augment the low-level reactive capabilities of integrated planning and control for handling complex, unpredictable environments so that the workload of the high-level planner can be consequently eased. The following key enhancements distinguish our framework from existing architectures:

A. *Perform fine, smooth and efficient motion control*

A high degree of smoothness and precision in motion control is essential to the efficient execution of sophisticated tasks and the social interaction with humans. This can only be achieved with *continuous response encoding* (i.e., infinite set of responses) of very low-level velocity/torque control of motor/joint actuators. To do so, an alternative would be to encode all possible action combinations [20], [23] but it becomes computationally intractable with higher degrees of freedom. Our proposed architecture trains a self-organizing neural network to continuously sample the low-level configuration space. Other integrated architectures [2], [6], [9], [12], [14] utilize potential fields [11] in their reactive controllers to encode continuous responses, which are subject to local minima problems [13]. In contrast, integrated architectures [24], [25] that employ *discrete response encoding* (i.e., finite, enumerated set of responses) encode high-level motion commands (e.g., {forward,left,right,…,etc.}), which may not be physically realizable due to negligence of kinematic constraints (e.g., non-holonomy). Interpolation of these discrete commands may incur problems similar to potential fields.

B. *Scale to more sophisticated motion tasks*

Extensive simplification of the action selection mechanism can unnecessarily deflate the reactive capabilities to achieving only simple tasks; information useful to selecting actions has not been fully exploited. Arbitration strategies [8], [19] are severely victimized by their policy of one winning behavior among a group of competing ones to assume full control of the robot until the next selection cycle. This precludes the execution of several, possibly conflicting motion tasks (e.g., target reaching and obstacle avoidance) in parallel. Superposition techniques [1], [4], [11],

[15] perform a vector sum of action commands, each optimal to its respective behavior, to produce a combined output that may not guarantee the satisfaction of any active behaviors. Problems of local minima and no passage between narrowly spaced obstacles may arise [13]. Orientation selection models [3], [25], [26] face similar problems due to a lack of distance perspective. This class of methods allow a robot with long range sensors to detect and avoid complex obstacles but the distant presence of a narrow doorway may be missed due to poor resolution at long range. It also cannot slow down while approaching obstacles. Our architecture presents a versatile action selection mechanism that utilizes both distance and angle information to execute complex motion tasks (e.g., avoiding complex obstacles) requiring a multitude of concurrently active behaviors.

C. Achieve asynchronous execution of behavioral modules

The asynchronous execution of behaviors, each at its fastest rate possible, is crucial to the preservation of reactive capabilities. Integrated architectures (e.g., [24], [25]) that coordinate their behaviors to operate directly on the action representation require them to be synchronized to produce a meaningful action. On the other hand, our architecture utilizes a time independent action selection module to learn a neural map representation of the local workspace that can interface with asynchronized behaviors.

D. Remove redundant overhead in behavioral modules

If every behavior accounts for the non-holonomic constraints and operates directly with the configuration space (e.g., [20], [21], [23]), a considerable duplication of effort would result, incurring a huge computational burden especially with higher degrees of freedom. Besides, such an architecture would be tedious to reconfigure for other robotic platforms. In contrast, our architecture performs all these within a single interchangeable action selection module. It hosts a low-dimensional neural map representation of the local workspace to interface seamlessly with the behaviors and planner. It also effectuates the mapping from this local workspace to the configuration space. The tight integration between the behaviors and the action selection module improves the overall computational efficiency. The behaviors can thus be easily reused for other robotic platforms.

These enhancements to the reactive competences enable the deliberative planner to be simplified; it produces a sequence of checkpoints in the global workspace instead of a complete motion path in the configuration space. The constraint of adhering strictly to a generated path is removed.

II. Integrated Planning & Control Framework

A. Overview

Our integrated planning and control framework is illustrated in Fig. 1. At the highest level, the *planning* module

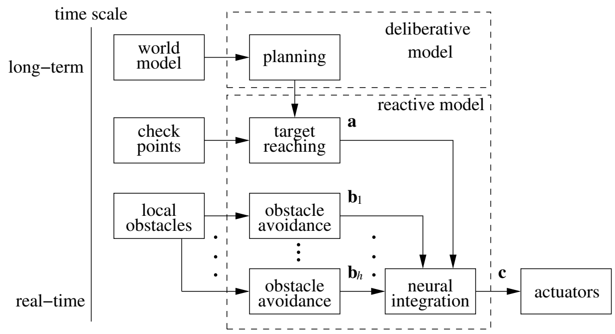


Fig. 1. A framework for the tight integration of planning, target reaching, and obstacle avoidance.

produces a sequence of checkpoints from the start point to the goal using an approximate cell decomposition method in [21]. However, our algorithm operates in the robot's workspace instead of the configuration space. In essence, the free workspace is decomposed into much fewer cells than other decomposition techniques to reduce the search time but any two points in the cell can still be traversed by reactive motion. This method is elaborated in a separate paper [17]. The planning module is unlike conventional planning algorithms [16] that plot a detailed path.

The reactive model is constituted by *Cooperative Extended Kohonen Maps* (EKMs), which are trained to produce a collision-free sequence of low-level (motor velocity) control commands that move the non-holonomic mobile robot from one checkpoint to the next. The *target reaching* module contains a *target localization EKM* that provides excitatory inputs to the *motor control EKM* in the *neural integration* module at and around locations of the sensed checkpoint. The *obstacle avoidance* modules host *obstacle avoidance EKMs*, which provide inhibitory inputs to the motor control EKM at and around locations where obstacles are detected. The motor control EKM in the neural integration module combines these inputs from the EKMs and uses them to select appropriate actions that can negotiate unforeseen obstacles while reaching targets.

All the modules operate asynchronously at different rates. The planning module typically operates at the time scale of several seconds or minutes depending on task complexity. The target reaching module operates at about 256 ms between servo ticks while the obstacle avoidance module operates at intervals of 128 ms. The neural integration module is activated as and when neural activities are received. The asynchronous execution of modules is the key to preserving reactive capabilities while allowing improvement of performance by the deliberative planner. In fact, the planner can be removed and the resulting decapitated architecture degrades to a purely reactive system capable of less complex motion tasks.

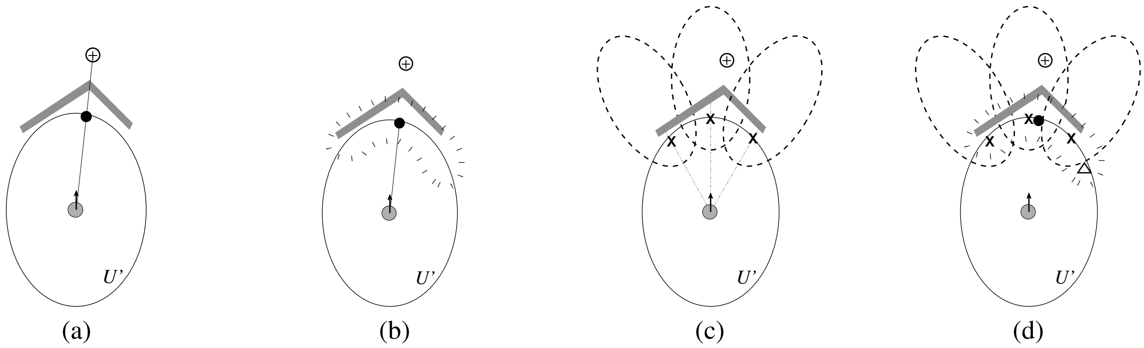


Fig. 2. Cooperative EKMs. (a) In response to the target $\oplus$, the nearest neuron (black dot) in the target localization EKM (ellipse) of the robot (gray circle) is activated. (b) The activated neuron produces a target field (dotted ellipse) in the motor control EKM. (c) Three of the robot's sensors 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 $\triangle$ to become the winner in the motor control EKM, which moves the robot away from the obstacle.

B. Target Reaching

The target reaching module adopts an egocentric representation of the sensory input vector $\mathbf{u}_p = (\alpha, d)^T$ where α and d are the direction and the distance of a checkpoint relative to the robot's current location and heading. At the goal state at time T, $\mathbf{u}_p(T) = (\alpha, 0)^T$ for any α. It 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 $\mathbf{u}_p$, the corresponding neuronal activities are sent to the motor control EKM in the neural integration module (Section II-D).

Target Localization

Given a sensory input vector $\mathbf{u}_p$ of a target location,

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

$$D(\mathbf{u}_p, \mathbf{w}_s) = \min_{i \in \mathcal{A}(\alpha)} D(\mathbf{u}_p, \mathbf{w}_i). \tag{1}$$

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

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

where β_α and β_d are constant parameters. The minimum in Eq. 1 is taken over the set $\mathcal{A}(\alpha)$ of neurons encoding very similar angles as α:

$$\begin{gathered} |\alpha - \alpha_i| \leq |\alpha - \alpha_j|, \\ \text{for each pair } i \in \mathcal{A}(\alpha), j \notin \mathcal{A}(\alpha) \ . \end{gathered} \tag{3}$$

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 towards it [28]. In the EKM, each neuron encodes a location $\mathbf{w}_i$ in the sensory input space $\mathcal{U}$. The region of $\mathcal{U}$ that encloses all the neurons is called the *local workspace* $\mathcal{U}'$. Even if the target falls outside $\mathcal{U}'$, the nearest neuron can still be activated (Fig. 2a).

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

$$a_i = G_a(\mathbf{w}_s, \mathbf{w}_i) \tag{4}$$

The function G_a is an elongated Gaussian:

$$G_a(\mathbf{w}_s, \mathbf{w}_i) = \exp(-(\frac{\alpha_s - \alpha_i}{\sigma_{a,\alpha}})^2 - (\frac{d_s - d_i}{\sigma_{a,d}})^2). \tag{5}$$

Parameter $\sigma_{a,d}$ 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 (Fig. 2b). This elongated Gaussian is the *target field*, which plays an important role in avoiding local minima during obstacle avoidance.

C. Obstacle Avoidance

Each obstacle avoidance module contains an obstacle localization EKM that is self-organized in the same way as the target localization EKM (Section II-E); each neuron i in the obstacle localization EKMs has the same input weight vector $\mathbf{w}_i$ as the neuron i in the target localization EKM. 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 sensed input $\mathbf{u}_j = (\alpha_j, d_j)^T$ induces neuronal activities, which are sent to the motor control EKM in the neural integration module (Section II-D).

Obstacle Localization

For each sensed input $\mathbf{u}_j$, $j = 1, \ldots, h$,

1) Determine the winning neuron s' in the jth obstacle localization EKMs. Each sensor input $\mathbf{u}_j$ activates

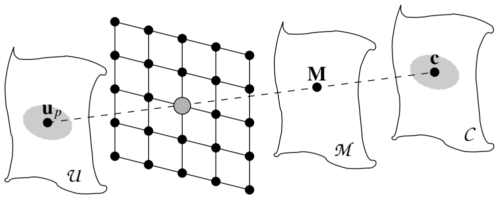


Fig. 3. Motor control EKM. The neurons map the sensory input space $\mathcal{U}$ indirectly to motor control space $\mathcal{C}$ through control parameter space $\mathcal{M}$.

a winning neuron s' in the jth obstacle localization EKM, which is activated in the same manner as Step 1 of Target Localization (Section II-B).

2) Compute output activity $b_{i,j}$ of neuron i in the jth obstacle localization EKMs:

$$b_{i,j} = G_b(\mathbf{w}_{s'}, \mathbf{w}_i) \tag{6}$$

where

$$\begin{aligned} G_b(\mathbf{w}_{s'}, \mathbf{w}_i) = &\ \exp(-(\tfrac{\alpha_{s'}-\alpha_i}{\sigma_{b,\alpha}})^2 - (\tfrac{d_{s'}-d_i}{\sigma_{b,d}(d_{s'},d_i)})^2) \\ \sigma_{b,d}(d_{s'}, d_i) = &\ \begin{cases} 3.5 & \text{if } d_i \geq d_{s'} \\ 0.035 & \text{otherwise.} \end{cases} \end{aligned} \tag{7}$$

The function G_b is a Gaussian stretched along the obstacle direction $\alpha_{s'}$ so that motor control EKM neurons beyond the obstacle locations are also inhibited to indicate inaccessibility (Fig. 2c). 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* (Eq. 6). The neurons nearest to the obstacle locations have the strongest activities.

D. Neural Integration

The neural integration module contains a motor control EKM, which integrates the excitatory and inhibitory inputs from the neurons in the target and obstacle localization EKMs respectively. It is trained to partition the sensory input space $\mathcal{U}$ into locally linear regions. Each neuron i in the motor control EKM is self-organized in the same way as the target and obstacle localization EKMs by encoding the same input weight vector $\mathbf{w}_i$ as the neuron i in those EKMs. It also has a set of output weights which encode the outputs produced by the neuron. However, unlike existing direct-mapping methods (e.g., [27], [22], [28]), the output weights $\mathbf{M}_i$ of neuron i of the motor control EKM represents control parameters in the parameter space $\mathcal{M}$ instead of the actual motor control vector (Fig. 3). The control parameter matrix $\mathbf{M}_i$ is mapped to the actual motor control vector $\mathbf{c}$ by a linear model (Eq. 10). Compared to direct-mapping EKM, indirect-mapping EKM can provide finer and smoother robot motion control. Detailed comparison and discussion have been reported in [18]. With indirect-mapping EKM, motor control is performed as follows:

Motor Control

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

$$e_i = a_i - \sum_{j=1}^{h} b_{i,j} \tag{8}$$

where a_i is the excitatory input from the neuron in the target localization EKM (Section II-B) and $b_{i,j}$ is the inhibitory input from the neuron in the jth obstacle localization EKM (Section II-C).

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

$$e_k = \max_i e_i \ . \tag{9}$$

3) Compute motor control vector $\mathbf{c}$ for target reaching:

$$\mathbf{c} = \begin{cases} \mathbf{M}_k\mathbf{u}_p & \text{if } -\mathbf{c}^* \leq \mathbf{M}_k\mathbf{u}_p \leq \mathbf{c}^* \text{ and } k = s \\ \mathbf{M}_k\mathbf{w}_k & \text{otherwise.} \end{cases} \tag{10}$$

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 $\mathbf{u}_p$ because $\mathbf{w}_k$ is closest to $\mathbf{u}_p$ compared to other weights $\mathbf{w}_i, i \neq k$.

In activating the motor control EKM (Fig. 2d), the obstacle fields are subtracted from the target field (Eq. 8). 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 field becomes most highly activated (Fig. 2d). 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 towards the target until it reaches the goal state $\mathbf{u}_p(T)$ at time step T.

Recall that the target reaching and obstacle avoidance modules run asynchronously at different rates. Since the neural integration module receives inhibitory inputs from the obstacle avoidance modules at a faster rate, excitatory inputs from the target reaching module may not be generated in time for neural integration. To resolve this synchronization issue, the neural integration module can feedback the most recent $\mathbf{u}_p$ or $\mathbf{w}_k$ computed from Eq. 10 to the target reaching module as an updated sensory input for generaton of new excitatory activities. Such a sensory input would be valid only if the EKMs are trained at time intervals longer than the servo ticks of the behavioral modules.

E. Self-Organization of EKMs

In contrast to most existing methods, online training 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. As the robot moves around and learns the correct mapping, its sensorimotor control becomes more accurate. At this stage, the same online training mainly fine tunes the indirect mapping. The self-organized training algorithm (in obstacle-free environment) is as follows:

Self-Organized Training

Repeat

1) Get sensory input $\mathbf{u}_p$.
2) Execute target reaching procedure and move robot.
3) Get new sensory input $\mathbf{u}'_p$ and compute actual displacement $\mathbf{v}$ as a difference between $\mathbf{u}'_p$ and $\mathbf{u}_p$.
4) Use $\mathbf{v}$ as the training input to determine the winning neuron k (same as Step 1 of Target Reaching).
5) Adjust the weights $\mathbf{w}_i$ of neurons i in the neighborhood $\mathcal{N}_k$ of the winning neuron k towards $\mathbf{v}$:

$$\Delta\mathbf{w}_i = \eta\, G(k,i)(\mathbf{v} - \mathbf{w}_i) \tag{11}$$

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.

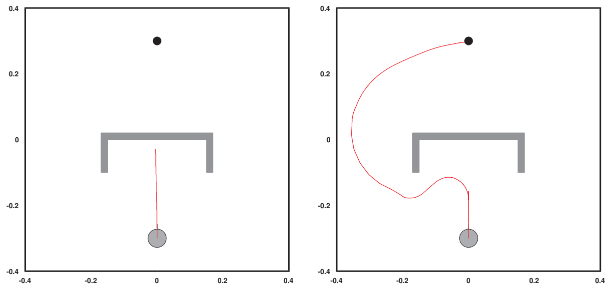
6) Update the weights $\mathbf{M}_i$ of neurons i in the neighborhood $\mathcal{N}_k$ to minimize the error e:

$$e = \frac{1}{2} G(k,i)\|\mathbf{c} - \mathbf{M}_i\mathbf{v}\|^2 \,. \tag{12}$$

That is, apply gradient descent to obtain

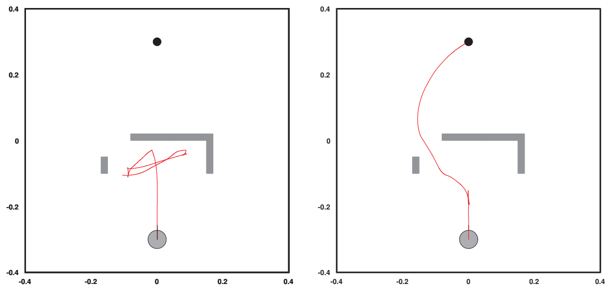
$$\Delta\mathbf{M}_i = -\eta\frac{\partial e}{\partial \mathbf{M}_i} = \eta\, G(k,i)(\mathbf{c} - \mathbf{M}_i\mathbf{v})\mathbf{v}^T \,. \tag{13}$$

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 towards 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 the sensory input $\mathbf{u}_p = \mathbf{w}_k$, the neuron will produce a motor control output $\mathbf{c} = \mathbf{M}_k\mathbf{w}_k$ which yields a desired displacement of $\mathbf{v} = \mathbf{w}_k$. For a sensory input $\mathbf{u}_p \neq \mathbf{w}_k$ but close to $\mathbf{w}_k$, the motor control output $\mathbf{c} = \mathbf{M}_k\mathbf{u}_p$ 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 $\mathcal{U}$, the indirect-mapping EKM can produce finer and smoother motion control than the direct-mapping EKM.



(a) (b)

Fig. 4. Negotiating unforeseen concave obstacle. (a) The robot using Braitenberg obstacle avoidance and superposition based action selection was trapped but (b) the one adopting the integrated architecture with cooperative EKMs successfully overcame the obstacle.



(a) (b)

Fig. 5. Passing through unforeseen narrow doorway between closely spaced obstacles. (a) The robot using Braitenberg obstacle avoidance and superposition based action selection was trapped but (b) the one adopting the integrated architecture with cooperative EKMs successfully passed through the narrow doorway to the goal.

III. Experiments and Discussions

Quantitative evaluation of the performance of indirect-mapping EKM for motor control has already been presented in [18]. This section presents a qualitative evaluation of the enhanced reactive capabilities of the integrated architecture with cooperative EKMs. The experiments were performed using Webots, Khepera mobile robot simulator (http://www.cyberbotics.com), which incorporated 10% noise in its sensors and actuators. 12 directed long-range sensors were also modelled around its body of radius 2.5cm. Each sensor had a range of 17.5cm, enabling the detection of obstacles at 20cm or nearer from robot center. To simulate noise, the sensors have a resolution of 0.5cm.

Two tests were performed to compare the integrated architecture using cooperative EKMs with that presented in [18]; the latter used Braitenberg's Type-3C vehicle [4] for obstacle avoidance and superposition (or vector sum) technique for action selection. For both architectures, the target reaching and obstacle avoidance modules ran at intervals of 256ms 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 (Fig. 4), the robot fitted with the Braitenberg scheme and superposition technique got trapped by the concave obstacle (Fig. 4a). The target reaching module tried to move the robot forward to reach the target while the obstacle avoidance module moved it backward to avoid the obstacle. The combined output cancelled each other's efforts. On the other hand, the robot that adopted the integrated architecture with cooperative EKMs successfully overcame the obstacle to reach the goal (Fig. 4b).

In the second test (Fig. 5), the robot endowed with the Braitenberg scheme and superposition technique could not pass through the narrow doorway between closely spaced obstacles (Fig. 5a). The repulsive forces from the walls counteracted the attractive force to the designated goal. In contrast, the robot that adopted the integrated architecture with cooperative EKMs successfully traversed through the narrow doorway to the goal (Fig. 5b).

These two tests demonstrate that for the architecture with superposition based action selection mechanism, though each behavioral module proposes an action optimal to itself, the vector sum of these action commands produces a combined output that may not guarantee the satisfaction of any behaviors. The neural integration method used in the integrated architecture with cooperative EKMs, however, considers the suboptimal preferences of each behavioral module via excitatory or inhibitory inputs and integrates them to determine an action that can satisfy each behavior to a certain degree. Such tightly coupled interaction between the behaviors and the action selection mechanism enable more complex motion tasks to be achieved.

The environment for the next test with unforeseen static obstacles consisted of three rooms connected by two doorways (Fig. 6). The robot began in the top corner of the left-most room and was tasked to move into the narrow corner

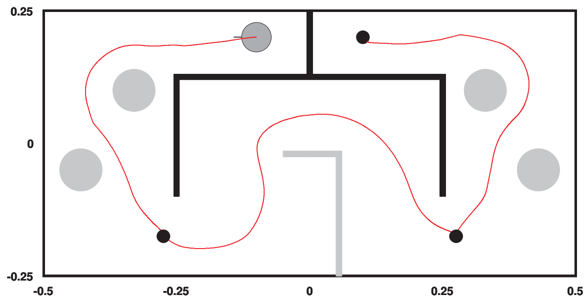


Fig. 6. Motion of robot (dark gray) in an environment with unforeseen static obstacles (light gray). The checkpoints (small black dots) were located at the doorways and the goal position. The robot can successfully navigate through the checkpoints to the goal by traversing between narrowly spaced convex obstacles in the first and the last room and overcoming a concave obstacle in the middle room.

of the right-most room via three checkpoints plotted by the planner. It was regarded to have reached a checkpoint if it was less than 5mm from the checkpoint and was required to stop at the goal. The robot is able to navigate through the checkpoints to the goal by traversing between narrowly spaced convex obstacles in the first and the last room and overcoming a concave obstacle in the middle room. The results of this test demonstrate the efficacy of the integrated planning and control architecture with cooperative EKMs in handling complex, unpredictable environments.

The environment for the last test also consisted of three rooms connected by two doorways (Fig. 7). The middle room contains two Khepera robots moving in anticlockwise circular paths. The robot began in the left-most room and was tasked to move to the right-most room without the help of the planner. Nevertheless, the robot is able to negotiate past the extended walls and the dynamic obstacles to reach the goal. This experiment further corroborates the enhanced reactive capabilities of the integrated architecture with cooperative EKMs to be capable of performing more complex motion tasks.

IV. Conclusion

By tightly integrating the target reaching, obstacle avoidance and neural integration modules with cooperative EKMs, the reactive capabilities of an integrated planning and control mobile robot architecture can be enhanced. This is extremely useful in complex, unpredictable environments where a robot controlled by this method can perform more complex motion tasks like negotiating unexpected concave and extended obstacles, and traversing between narrowly spaced obstacles. These characteristics distinguish our architecture from those integrated frameworks (e.g., [2], [6], [18]) that employ superposition based action selection mechanisms [1], [4], [11], which can be easily trapped by unforeseen concave obstacles and narrowly spaced obstacles unless global replanning is executed. Our

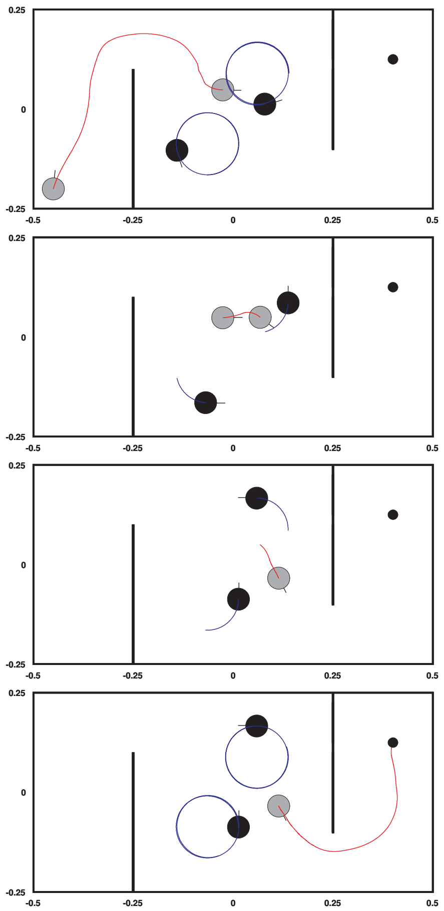


Fig. 7. Time sequence of the motion of robot (dark gray) in an environment with two unforeseen Khepera robots (black) moving in anti-clockwise circular paths. The robot can successfully negotiate past the extended walls and the dynamic obstacles to reach the goal without the help of the planner.

continuing research goal is to apply the integrated framework to the planning and control of robot manipulators.

References

[1] R. C. Arkin. Motor schema based mobile robot navigation: An approach to programming by behavior. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 264–271, 1987.

[2] R. C. Arkin and T. Balch. AuRA: Principles and practices in review. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2-3):175–189, 1997.

[3] J. Borenstein and Y. Koren. The vector field histogram: Fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

[4] V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. MIT Press, Cambridge, MA, 1984.

[5] O. Brock and L. E. Kavraki. Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 2, pages 1469–1474, 2001.

[6] O. Brock and O. Khatib. Executing motion plans for robots with many degrees of freedom in dynamic environments. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 1–6, 1998.

[7] O. Brock and O. Khatib. High-speed navigation using the global dynamic window approach. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 341–346, 1999.

[8] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[9] W. Choi and J. C. Latombe. A reactive architecture for planning and executing robot motions with incomplete knowledge. In *Proceedings of IEEE/RSJ International Workshop on Intelligent Robots and Systems*, volume 1, pages 24–29, 1991.

[10] D. Fox, W. Burgard, S. Thrun, and A. B. Cremers. A hybrid collision avoidance method for mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 2, pages 1238–1243, 1998.

[11] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 500–505, 1985.

[12] O. Khatib, S. Quinlan, and D. Williams. Robot planning and control. *Robotics and Autonomous Systems*, 21:249–261, 1997.

[13] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1394–1404, 1991.

[14] B. H. Krogh and C. E. Thorpe. Integrated path planning and dynamic steering control for autonomous vehicles. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1664–1669, 1986.

[15] E. W. Large, H. I. Christensen, and R. Bajcsy. Dynamic robot planning: Cooperation through competition. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 3, pages 2306–2312, 1997.

[16] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.

[17] K. H. Low. Integrated robot planning and control with extended Kohonen maps. Master's thesis, Department of Computer Science, School of Computing, National University of Singapore, July 2002.

[18] K. H. Low, W. K. Leow, and M. H. Ang Jr. Integrated planning and control of mobile robot with self-organizing neural network. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 4, pages 3870–3875, 2002.

[19] P. Maes. Situated agents can have goals. In P. Maes, editor, *Designing Autonomous Agents: Theory and Practice From Biology to Engineering and Back*, pages 49–70. MIT Press, Cambridge, MA, 1990. Also appeared in *Special Issue of Robotics and Autonomous Systems*, **6**(1-2).

[20] P. Pirjanian. Multiple objective behavior-based control. *Robotics and Autonomous Systems*, 31(1-2):53–60, 2000.

[21] S. Quinlan and O. Khatib. Towards real-time execution of motion tasks. In R. Chatila and G. Hirzinger, editors, *Experimental Robotics 2*. Springer-Verlag, 1993.

[22] R. P. H. Rao and O. Fuentes. Hierarchical learning of navigational behaviors in an autonomous robot using a predictive sparse distributed memory. *Machine Learning*, 31(1-3):87–113, 1998. Also appeared in *Autonomous Robots*, **5**(3-4), 297–316.

[23] J. Riekki and J. Röning. Reactive task execution by combining action maps. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 224–230, 1997.

[24] J. K. Rosenblatt. DAMN: A distributed architecture for mobile navigation. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2-3):339–360, 1997.

[25] A. Saffiotti, K. Konolige, and E. Ruspini. A multi-valued logic approach to integrating planning and control. *Artificial Intelligence*, 76(1-2):481–526, 1995.

[26] R. Simmons. The curvature-velocity method for local obstacle avoidance. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3375–3382, 1996.

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

[28] C. Versino and L. M. Gambardella. Learning the visuomotor coordination of a mobile robot by using the invertible Kohonen map. In *Proceedings of International Workshop on Neural Networks*, 1995.