Unknown Environment Representation for Mobile Robot Using Spiking Neural Networks

Amir Reza Saffari Azar Alamdari

***Abstract*—** In this paper, a model of self-organizing spiking neural networks is introduced and applied to mobile robot environment representation and path planning problem. A network of spike-response-model neurons with a recurrent architecture is used to create robot's internal representation from surrounding environment. The overall activity of network simulates a self-organizing system with unsupervised learning. A modified A* algorithm is used to find the best path using this internal representation between starting and goal points. This method can be used with good performance for both known and unknown environments.

***Keywords*—**Mobile Robot, Path Planning, Self-organization, Spiking Neural Networks.

I. Introduction

RECENTLY, there is a great effort by many researchers to create models of human brain with computational capabilities to solve some real world problems. Since neurons in central nervous system (CNS) communicate using short duration electrical impulses called spikes most of these models at neuronal levels use biologically more realistic neuron models called spiking neuron models. We will call these types of models spiking neural networks (SNNs) [1]. Several results show that feed-forward SNNs have at least the same computational power of multi-layer neural networks with sigmoidal neurons [2]. There are a few applications of SNNs to real world problems, and this is because of lack of powerful and efficient learning algorithms.

In this paper, a self-organizing SNN with especial configuration and learning algorithm is introduced and used to build a mobile robot's internal environment representation (also known as internal map) from surrounding world. There are strong evidences showing that dynamic behavior and information processing in many parts of the brain is self-organized [4]. Also there exist some SNN models showing self-organized behavior, see [4] and references there.

Environment representation and navigating is a cognitive task solved in a self-organized manner in humans and animals. Also these tasks play very important role in mobile robot navigation; especially in situations that environment is not fully understood or known. Here we address these problems using a brain-like computational system together with a modified version of A* algorithm (pronounced and known also as A-star) [3] for path planning. This model is intended to solve these problems in a totally unsupervised manner for unknown environments.

Manuscript received May 20, 2005. This work was supported by Research Administration of Sahand University of Technology, Tabriz, Iran.

Amir R. Saffari A. A. was with Electrical Eng. Dept. of Sahand University of Technology, Tabriz, Iran. He is now with the Institute for Theoretical Computer Science, Graz University of Technology, Austria (e-mail: amir@ymer.org).

II. Self-Organizing Spiking Neural Network

A. Network Structure and Model

The general structure of self-organizing SNN is shown in Fig.1. Neurons are organized in two layers: the first is sensory stage with only feedforward connections, and the second stage is processing layer with lateral or feedback connections. The sensory stage converts each external analog stimulations, x_j, into exact spike times based on first-spike-time encoding approach [1] using:

$$t_j = T - \alpha_j x_j \quad , \quad j = 1,2,...,N \tag{1}$$

which t_j is the associated spike time, T is the reference time, and α_j is a scaling factor. This scaling factor can be used to convert any physical quantity into a suitable value in time.

All neurons in the processing stage of the network are Spike Response Model (SRM) neurons [1], arranged in a 2D lattice, which equations describing the behavior for *i*th neuron are given by:

$$u_i = \eta(t - t_i) + \sum_{j=1}^{N+1} \varepsilon_s(t - t_j - d_{ij}) + \sum_{k=1}^{M} w_{ik} \varepsilon_f(t - t_k) \tag{2}$$

$$if\ (u_i > \theta_i)\ then\ t_i = t \tag{3}$$

which (2) is describing membrane potential, u_i, due to internal state and external stimulations while (3) stands for spike generation rule when membrane potential exceeds threshold value, θ_i. In (2), first term shows refractoriness effect of recent generated spike t_i which $\eta(s)$ represents refractory kernel. Second term is describing post-synaptic potentials (PSPs) generated by incoming sensory spikes with $\varepsilon_s(s)$ used for response kernel. t_j is input spike time from *j*th sensory neuron, and d_{ij} represents synaptic transmission delay between pre-synaptic neuron *j* and post-synaptic neuron *i*. The last term is PSPs associated with lateral feedback connections where w_{ik} represents synaptic weight for

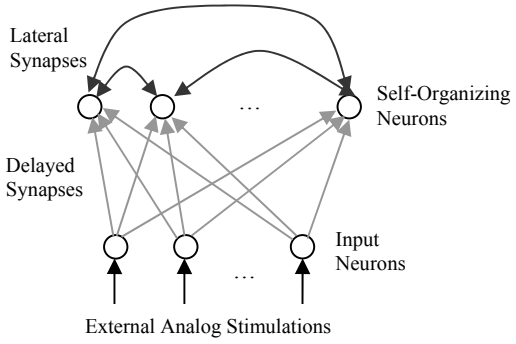

Fig.1 Network architecture

connection from *k* to *i*, t_k is feedback spike time from *k*th neuron, and $\varepsilon_f(s)$ is again response kernel. Kernel equations are described by:

$$\varepsilon_s(s) = \varepsilon_s(s) = \frac{1}{1-\tau_s/\tau_m}\left[\exp(-\frac{s}{\tau_m}) - \exp(-\frac{s}{\tau_s})\right]\Theta(s) \quad (4)$$

$$\eta(s) = \eta_0 \exp(-\frac{s}{\tau_m})\Theta(s)$$

which τ_m and τ_s are membrane and PSP current time constants, respectively, $\Theta(s)$ is a step function, and η_0 is a constant indicating strength of refractory effect on membrane potential. Note that the total number of input channels are *N* and total number of processing neurons are *M*. Also there exists a reference input channel which spikes exactly at time *T*. This reference spike will be used for normalization purposes, discussed later.

Before proceeding into dynamic analysis of network, initial values of delays and lateral weights should be specified. In this work delays values are drawn randomly from a uniform distribution in a specified range, except the delay value for the reference spike which is assumed to be zero. The initial values of lateral weights are chosen to be dependent on spatial distance between neurons and are given by:

$$w_{ij} = \begin{cases} 2\exp(-\frac{z_{ij}^2}{2\sigma_1^2}) - \exp(-\frac{z_{ij}^2}{2\sigma_2^2}) & i \neq j \\ 0 & i = j \end{cases} \quad (5)$$

which z_{ij} is the spatial distance between two neurons and can be calculated using several metrics. The σ_1 and σ_2 are constants specifying shape of weight distribution. Note that neurons do not have any self-feedback. Using this initial weight function neurons with low spatial distances would have excitatory lateral connections and those with high distances would receive feedback through inhibitory synapses. Also note that for simplicity it has been assumed that individual neurons can have both excitatory and inhibitory connections with each other. For more biologically realistic model, one can use columns of inhibitory neurons instead of negative weights.

B. Network Behavior

Activity of the neurons (i.e. spike firing times) can be described in this model in terms of synchronization of incoming spikes: the more input spikes are synchronized the more higher membrane potential can reach which means higher possibility to generate output spike. In other words, if the input spikes are close enough to each other, their response kernels are more aligned and this gives to higher values of activity in (2). There exist some evidences from biological research about synchronous spiking patterns in brain [5].

Since the model is input driven, synchrony between incoming spikes are achieved and controlled using sensory delay channels. Arrival time of input spike from *j*th sensory channel to *i*th neuron is, according to second right hand term in (2):

$$a_{ij} = t_j + d_{ij} = T - \alpha_j x_j + d_{ij}. \quad (6)$$

We can rewrite this equation in vector format for all sensory channels of *i*th neuron as:

$$\vec{A}_i = \vec{T} + (\vec{D}_i - \vec{X}) \quad (7)$$

which $\vec{A}_i$ is the vector of input spike arrival time, $\vec{T}$ is the vector of reference time, $\vec{D}_i$ is the vector of delays, and $\vec{X}$ is the scaled input vector. One interpretation of this equation is an Euclidian type of metric for comparison between scaled input vector with delay vectors of individual neurons in terms of synchrony of incoming spikes. So those neurons with less distance has more aligned input spikes and as a result are more likely to generate output spike. This gives the idea of winner neuron for a competitive learning mechanism: the first neuron which fires has less distance between its delay vector and scaled input vector.

To implement this idea we also need to take account one other problem: neurons with equally spaced distance (but not close to zero) between their delay vector and scaled input vector would also receive synchronous input spikes. This is the same normalization problem when we want to compute the winner in traditional competitive ANNs only by computing dot multiplication of their weights and input vectors. To solve this problem we introduce a reference spike that spikes at time *T*, so those neurons with distance close enough to zero will have chance to fire. From biological modeling approach, this reference spike can be related to brain wave oscillations in gamma band and coherent activities in the brain [6].

To complete behavior analysis, we have to mention the value of threshold chosen for this model. Threshold value is chosen according to:

$$\theta = N\max(\varepsilon_s(s), \varepsilon_f(s)) + \delta \ , \ s > 0 \ , \delta > 0 \quad (8)$$

which δ is a positive constant. This means that at least *N*+1 synchronized or partially synchronized input spikes at time *T* are needed to excite efficiently a processing neuron to produce an output spike. Note that *N* is the number of sensory input channels.

C. Synaptic Plasticity

There are two adjustable parameters in this network model: input transmission delays and lateral weights. Some recent

findings show that transmission delays in biological neural networks can be adaptable [7]. A delay shift mechanism is used as adaptation rule for transmission delays in this work which is a Hebbian adaptation system in nature. For lateral synaptic connections, also a Hebbian mechanism is used to adjust the weights, as described below.

In the adaptation process, both weights and delays of those neurons which were able to produce an output spike are adjusted using (9), which is a special type of spike time dependent plasticity (STDP). Note that in (9) it is assumed that the neuron *i* is the winner (fired first spike), *j* is the index of input sensory neuron, *m* and *n* are the indexes of the other spiking neurons in the processing layer with t_m and t_n as output spike times, respectively, w_{mn} is the synaptic weight connecting neuron *n* to *m*, and z_{mn} is the spatial distance between two neurons *m* and *n*.

$$\Delta d_{mj} = \eta_d h(m,i)(x_j - d_{mj})$$

$$\Delta w_{mn} = \begin{cases} +\eta_w h(m,n)(1-w_{mn})\exp(t_m - t_n/\tau_1), & t_m \geq t_n \\ -\eta_w h(m,n)(1+w_{mn})\exp(t_n - t_m/\tau_2), & t_m < t_n \end{cases} \quad (9)$$

$$h(m,n) = \exp(-z_{mn}^2/2\sigma^2)$$

Note that there is a general neighborhood function, $h(m,n)$ which limits the adjustments applied to the neurons with high distances, using parameter σ. In lateral weight adaptation mechanism there exists a limiting factor, $(1 \mp w_{mn})$, which saturates the weight on ± 1 values when weight tends to grow unbounded. Note that when pre-synaptic neuron fires before post-synaptic neuron, there is an increase in weight value with a time constant of τ_1, and synaptic weight decreases when post-synaptic neuron fires before pre-synaptic neuron with a time constant of τ_2. Learning steps can be controlled through η_d and η_w.

Using these adaptation mechanisms, this network exhibits an especial dynamic behavior, like what is seen in self-organizing maps (SOMs). In Fig.2 an example of applying this network over a random distribution of two dimensional input data is shown. After several epochs of simulation, delay vectors of network would be organized to represent input space cluster distributions. Lateral weights adaptation helps to improve the clustering process and faster movement of delay vector to desired locations in input space.

III. Application to Mobile Robot Environment Representation and Path Planning

A. Environment Representation

In previous sections it was shown that after several training epochs the presented self-organizing spiking neural network can adapt its delay vectors to input space distribution. In a path planning problem for a given map, all free non-obstacle places can be viewed as input clusters and the network task as internal representation builder is to detect and classify these areas as walk-able locations of map. In this way, any delay vector is named as a node in the map, and has two variables related to coordinates of input location (*x*,*y*). This definition can be extended to 3-dimensional spaces easily. It is supposed that mobile robot is equipped with location recording systems like GPS or odometer. Also it is assumed that robot has elementary obstacle detection and anti-collusion devices.

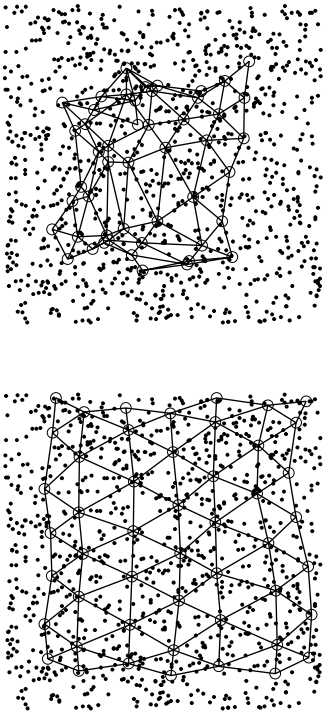

Fig.2 Different steps of self-organizing behavior of network over random inputs to the network (black dots). Up: nodes configuration after several steps of training. Bottom: nodes structure after finishing training phase

To specify links or tracks for robot movement, nodes of six neighbor processing neurons are connected to form a topology of neurons representing internal map. For a given start and goal positions, robot has to plan a path to navigate through obstacles to the goal point (path planning algorithm is discussed in next section). In each movement step, the coordinates of robot's position are used as input values to the network to run a simulation and adaptation process. Fig.3 is an example of a movement and its associated effect on node positions. The overall effect is the movement of network nodes to the locations which robot were there, i.e. free locations. For known maps, the internal representation formation can be done in offline manner, without forcing robot to move to every free location. For unknown maps, a search and exploration phase can be applied to robot with manually or randomly chosen starting and goal points. In this case closer start and finish points are desirable. Fig.3 shows an example of an unknown environment for robot and the effect of movement in this space on internal representation.

B. Path Planning

One of the famous and efficient path planning algorithms is the A* [3]. An A* path finding system is an optimization algorithm with modified best-first-search (BFS) strategy which uses a heuristic travel cost estimation. For an A* algorithm, environment is represented with cells or grids where robot can or can not move on. Algorithm begins searching with neighbor cells of start point and calculates cost of movement to each next cell (excluding blocked cells and previous moved ones) associated with general heuristic cost approximation to travel from next point to goal. The algorithm continues with selecting best next cell with lower cost and then doing the same steps again for this new one. A cost estimation correction phase is also implemented in this level for shared neighbors of new and previous cell. For more details on this powerful method refer to [3].

The modification applied to the original A* algorithm are the redefinitions of cells and links which are necessary to make it compatible with network internal map representation. The first modification is to replace the concept of cells with delay nodes. For a given start node, the algorithm is the same except that any neighbor node which has a blocked link or itself is on obstacle is discarded from search algorithm temporarily. The discarded node may be entered in search list again in next steps, in contrast to original A* which the blocked cells are completely removed from the search list. In this way, only the previously traveled links are added to closed list in addition with the links which end node is on the obstacle location. There is another issue to be noted about start and finish locations. Because this algorithm uses nodes instead of cells positions, it is possible to define start or finish point not on the internal map nodes. In this situation, a new representative node will be added to the network on this point with linking and altering the original map to a new one with a node on start or stop point.

IV. Discussion

It is obvious from example presented in Fig.3 that this method has a good performance even for complex maze type obstacle configurations. Also the modified A* algorithm is much faster than the original one, especially when the robot size is much smaller than the length and width of map. The main reason is that this new algorithm only needs to search a smaller number of nodes and links compared to original A*, making it very efficient for large maps. Another benefit of this algorithm is that it can be applied for unknown environments without any supervision, since robot gradually makes its own representation from interaction with environment (i.e. exploration).

The main drawback is that in some area of the map with very narrow passages, this algorithm may not be able to find a path comparing to A* which is guaranteed to find at least a path if any exists. This situation is compensated by adding additional random nodes to the complex areas. Also, adding turning angle penalty to overall cost results in smoother path planning between nodes.

From modeling view, this system is based purely on a biological computation basis, with a successful application of a brain-like processing model to a real world problem. Neuronal synchronization and coherency formation are one of the challenging problems in modeling brain activities. This model is also based on synchronization of different neurons in processing layer with a coherent oscillation, named reference spike. Also a biological plausible adaptation mechanism, which is a Hebbian-like learning system, is used to adjust network parameters to proper values in a self-organizing manner.

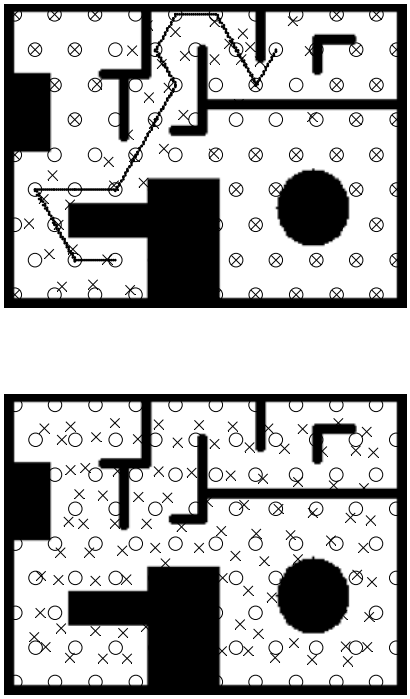

Fig.3 The effect of learning environment, circles represent initial internal map nodes, before learning environment, X represent current node positions. Up: changes in internal representation due to movement from start to goal points, the path is shown by dots. Bottom: internal representation after several movements in area

References

[1] W. Gerstner, W. M. Kistler, *Spiking Neuron Model: Single Neuron, Populations, and Plasticity.* Cambridge University Press 2002.

[2] W. Maass, "*Lower Bounds for the Computational Power of Spiking Neurons*" *Neural Computation*, vol. 8, pp 1-40, 1996.

[3] A. J. Patel "*Game Programming: Path Planning*", [Online], http://www-cs-students.stanford.edu/~amitp/gameprog.html#Paths.

[4] Y. Choe, "*Perceptual Grouping in a Self-Organizing Map of Spiking Neurons*" Ph.D. dissertation, University of Texas at Austin, 2001.

[5] R. Eckhorn, M. Arndt, P. Dike, "*Feature Linking via Synchronization Among Distributed Assemblies: Simulation Results from Cat Visual Cortex*" *Neural Computation*, vol. 2, pp 293-307, 1990.

[6] R. Eckhorn, R. Bauer, W. Jordan, M. Brosch, W. Kruse, M. Munk, H. J. Reitboeck, "*Coherent Oscillations: A Mechanism of Feature Linking in the Visual Cortex?*" *Biological Cybernetics*, vol. 60, pp 121-130, 1988.

[7] C. W. Eurich, K. Pawelzik, U. Ernst, A. Theil, J. D. Cowan, J. G. Milton "*Delay Adaptation in the Nervous System*" *Neurocomputing*, vol. 32-33, pp 741-748, 2000.