Robot Map Building by Kohonen's Self-Organizing Neural Networks

Article · November 1998

Source: CiteSeer

CITATIONS
11

READS
208

3 authors, including:

Nikos Vlassis
University of Luxembourg
162 PUBLICATIONS **5,967** CITATIONS
SEE PROFILE

Panayiotis Tsanakas
National Technical University of Athens
143 PUBLICATIONS **1,782** CITATIONS
SEE PROFILE

Robot Map Building by Kohonen's Self-Organizing Neural Networks

Nikos A. Vlassis George Papakonstantinou
Panayiotis Tsanakas

National Technical University of Athens
Department of Electrical & Computer Engineering
Zographou Campus, 15773 Athens, Greece

E-mail: nvlassis@dsclab.ece.ntua.gr

Abstract

A fundamental issue in mobile robotics is map building. The term refers to cases where a robot is forced to move in an unknown environment and has to build a map entirely based on its sensory information. In this paper we present a method for building robot maps by using a Kohonen's self-organizing artificial neural network, and describe how path planning can be subsequently performed on such a map. We show that our method can also be applicable in cases of a pre-existent CAD of the environment.

1 Introduction

For the problem of building and maintaining a robot map to be used as a basis for robot's other tasks, like path planning, two are the main lines in the literature, namely, the *grid-based* approach and the *topological* approach. The first one [3] uses an accurate description of the environment that is based on a grid tessellation where each grid cell denotes a small and precisely pre-determined region of the total space.

On the other hand, topological approaches [10, 12, 8] build maps as graphs where nodes depict distinctive places in the environment and vertices correspond to paths between places. A recent approach that combines the above two approaches was proposed in [16], where a topological map was built on top of a grid-based one.

A relatively new area in the field has emerged by the introduction of neural networks in robot map building and navigation [17, 2, 1, 11, 19, 9]. Here, neural networks have been proven handy tools for mapping the robot environment to a dense graph-representation using a combination of external sensor information and dead-reckoning.

Our approach uses a *self-organizing neural network* [7] to build an internal representation of the robot free space and then perform path planning on it. By exploiting the odometry information of the robot a topological graph of the environment is automatically built where nodes correspond to neurons and paths to lateral connections between neurons. Our simulation results demonstrate the potential of the method to handle both cases of known or unknown environments.

2 Self-Organizing Maps

A self-organizing map (SOM) [7] is an artificial neural network the cells of which are placed at the nodes of a lattice and become selectively tuned to various classes of input signal patterns through an unsupervised learning process. The cells on this lattice tend to become ordered with respect to each other, thus generating actually a mapping from a high-dimensional continuous signal space to a lower-dimensional topological structure.

A SOM is usually one or two-dimensional. Each cell in the map is connected to the input signals and the weight of the respective connection denotes the input vector the cell is maximally tuned to. After learning of the SOM, the weight vector of a particular cell defines the center of a certain class of the input data. In Fig. 1 a typical two-dimensional hexagonal SOM is shown, where each cell is connected to its 6 equal-distant neighbors. The input vector $\vec{\xi} = (\xi_1, \xi_2, \ldots, \xi_n) \in \mathbf{R}^n$ is connected

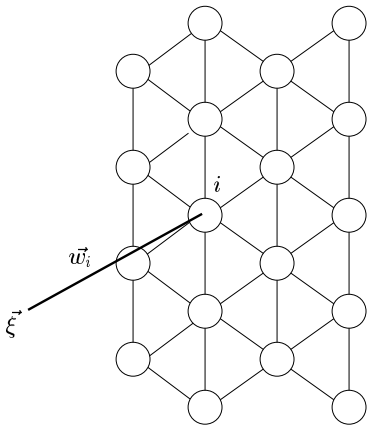


Figure 1: The hexagonal SOM.

to cell i and the respective connection has weight $\vec{w}_i = (w_{i1}, w_{i2}, \ldots, w_{in}) \in \mathbf{R}^n$.

Let us assume that we have a SOM of M total cells. The learning of the SOM is performed in the following steps:

1. Initially, we assign to the weight vectors $\vec{w}_i$ either random values, or if we have an a-priori knowledge of the inputs distribution we give them appropriate initial values so as to meet the needs of the specific problem. We normalize the values of $\vec{w}_i$ in $[0, 1]$.

2. We draw a random sample $\vec{\xi}$ from the probability distribution of the input samples and normalize it in [0,1]. Then we compute the cell i that is closest to it by means of the Euclidean distance, i.e.,

$$\|\vec{\xi} - \vec{w}_i\| = \min, \quad i = 1, 2, \ldots, M$$

3. In its simplest form, the adaptation process adjusts the weight vector of cell i and the weight vectors of its neighboring cells in N_i by

$$\vec{w}_j = \vec{w}_j + \alpha(\vec{\xi} - \vec{w}_j), \quad j = i, \quad j \in N_i$$

where α is a time decreasing adaptation gain with $0 < \alpha < 1$, and N_i may initially extend over many neighboring cells and after a certain number of steps (ordering phase) degenerates to the winning cell i only. If a pre-defined number of steps is not exceeded we repeat step 2.

Although difficult to prove mathematically, extensive simulations over numerous applications have shown that the above algorithm produces maps *topologically ordered*, i.e., maps in which neighboring cells have near weights. Variations to this basic algorithm have been shown, e.g., in [5, 15], and reported satisfactory results.

3 Self-Organizing Maps in Robot Navigation

3.1 Building the map

The learning capabilities of the self-organizing maps make them attractive for the problem of building robot maps. Usually a robot has limited or no knowledge about its environment. In most cases it has to build a good map from scratch by exploring the free space. Moreover, this map must be convenient for other tasks of the robot, like path planning.

The notion of building self-organizing maps for robot navigation stems from the observation that the environment the robot moves in can be viewed as a two-dimensional input space, with high probability in open regions and low probability over obstacle regions. In mathematical terms this can be expressed as

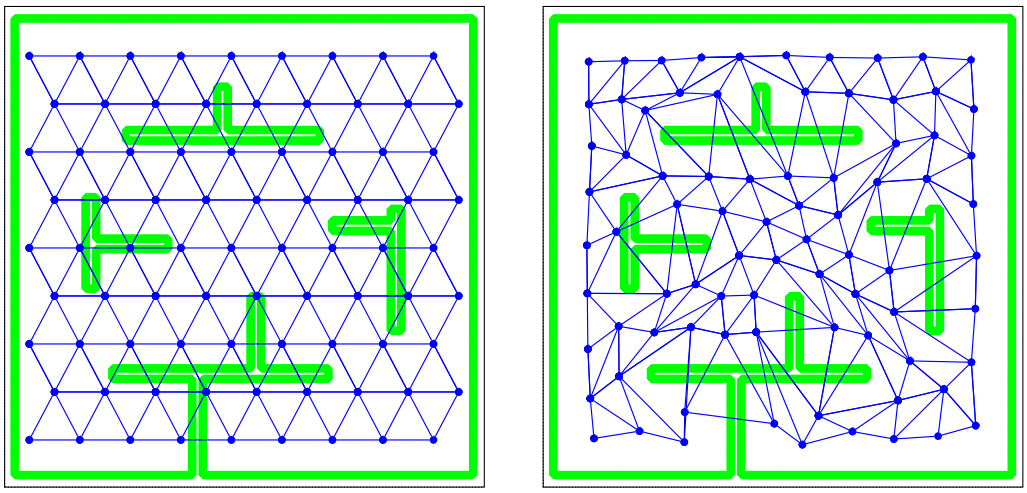
$$E = \{\vec{\xi} | \vec{\xi} = (x, y), \quad x, y \in \mathbf{R}\},$$

where E the robot environment and $E \in \mathbf{R}^2$.

Based on the above description, we build a two-dimensional self-organizing map with a pre-defined number of cells along each dimension, arranged in a hexagonal fashion. The number of cells is not constant, but depends on the structure of the environment. For typical indoor configurations our simulations showed that values between 8 and 11 in each dimension give good results.

The position of the robot (x, y) at every instance becomes the input vector $\vec{\xi}$ to the network. We initialize the weight vectors $\vec{w} = (w_x, w_y)$ of each cell so that they get uniformly distributed in the total space. In Fig. 2a we show the initial configuration of a 9×9 map before the learning procedure over a typical indoor arrangement of obstacles. Cells are plotted according to their weight vector $\vec{w}$, and the connections are established between neighboring cells of the map.

In order to adjust the cells of the map the robot starts exploring the environment and collects the (x, y) points of its trajectories. Assuming that the robot is equipped with an elementary obstacle avoidance mechanism and the (x, y) samples are correct (e.g., odometer measures are continuously adjusted) these values may appropriately train the SOM. After the end of the learning phase the weight vectors of the cells must depict classes of input samples of high



a. initial configuration b. after learning

Figure 2: The SOM before and after learning.

probability, hence be uniformly distributed among the free regions of the total space.

Applying the SOM learning procedure to the previous map leads to the configuration shown in Fig. 2b. The cells have been 'moved' so as to statistically follow the inputs and have been distributed in the free space.

In cases where a CAD map of the environment is known, the SOM can be trained by (x, y) samples drawn from the free area of the environment. For achieving better initial configuration of the cells, fake obstacles are built half the robot's diameter away from the real ones. In this way there is less probability that after training there are some cells in locations unreachable by the robot.

In Tab. 1 one can see the values of the various parameters used in the learning process. Time t corresponds to the number of steps.

3.2 Planning on the map

As shown in Fig. 2b, the produced map actually constitutes a graph with vertices the cells of the map and edges the connections between neighboring cells. The ordering properties of the SOM algorithm make this graph appropriate for path planning of the robot, since neighboring cells correspond to adjacent areas of the environment.

Our path planning method is based on a modified $A^\star$ algorithm [18] that applies over the learned SOM. The basic idea is that we transform the initial graph to a modified one with points and links, and then apply the classical $A^\star$ algorithm [14] on it. In the new graph a point denotes a pair of adjacent nodes of the initial graph together with their edge, whereas a link connects two neighboring points, i.e., two consecutive edges of the initial graph. By including the angle between two consecutive edges of the initial graph into the length (cost) of a link, the modified algorithm may penalize the paths that contain very steep curves and thus force the robot to move along smoother trajectories.

In order to find a path between the current robot position $\vec{s}$ and goal position $\vec{g}$ two dummy cells are inserted in the map, having weight vectors equal to $\vec{s}$ and $\vec{g}$, respectively. A linear search among the weight vectors of all map cells is performed in order to find their closest neighbors and respective connections are established. Then the modified $A^\star$ algorithm is applied to the extended graph and a path to goal is found.

As shown in Fig. 2b, after the SOM learning procedure there are certain edges that pass over obstacles, which of course are illegal. Initially the map contains no information about these illegal edges. Thus it is possible that the planner chooses a path through such an edge. The robot starts moving along the selected path and when its sensing mechanism perceives a blocking situation, e.g., an obstacle, the robot stops, appropriately updates the respective map connection, and searches for a new path to the goal position.

Input dimension	2
Number of Cells	9×9
Cells' configuration	Hexagonal
Ordering phase steps	1000
Total number of steps	200000
Adaptation gain α	$0.7 - 0.69t/1000, \quad t < 1000$ $0.01, \quad t \geq 1000$
Neighborhood N_i	$i + 6$ neighbors of $i, \quad t < 1000$ $i, \quad t \geq 1000$

Table 1: Parameters for the SOM learning.

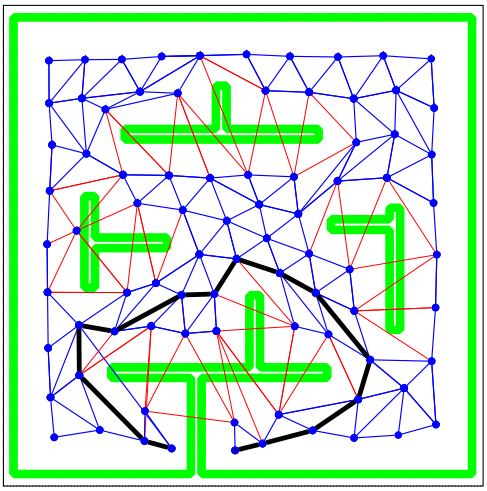


Figure 3: A final configuration.

In the current version a map connection is assumed to be either legal or illegal. The on-line update of a connection implies changing its state from legal to illegal, or the opposite. This information is taken into consideration by subsequent $A^\star$ runs and eventually only paths containing legal connections are selected. Fig. 3 shows the final configuration of the map after a successful planning. One can see the starting and goal positions captured by dummy cells and the chosen path.

4 Conclusions—Future work

In this paper we presented an algorithm for robot map building and navigation by means of a self-organizing neural network. We implemented and tested our method in the Khepera simulator [13] and our results showed that our approach works well in both cases of a known or unknown environment.

However, if our method is to be of practical use the hypothesis that the odometry measures are correct during map exploring and building must be relaxed. We are currently implementing a method of self-localization by correlating the sensoring data to the robot coordinates at each point, similar to the ones proposed in [19, 9].

A second issue is related to the binary nature of the SOM connections, i.e., legal—illegal. A more sophisticated approach would be to attach a probabilistic, time variant strength to each connection denoting for example the probability the robot can pass through. This way, dynamic environments, i.e., environments where obstacles are inserted or removed on the fly can be handled by appropriately altering the connections strengths.

As mentioned above, the number of cells in the map largely depends on the initial configuration of the environment. Since also dynamic environments are to be handled a unified method for adding or removing cells in the map should be employed, one that should not violate the hexagonality of the lattice structure. To this direction we plan to incorporate the self-organizing neural network model presented in [6] into our framework.

Finally, we are studying a method for trajectory smoothing, i.e., finding time-optimal trajectories by forcing the robot to follow continuous curvature paths [4].

References

[1] Balkenius C.: *Natural Intelligence in Artificial Creatures*. Ph.D. thesis, Lund University Cog-

nitive Science, Lund, Sweden, (1995).

[2] Donnart J.Y., Meyer J.A.: Hierarchical-map building and self-positioning with MonaLysa. *Adaptive Behavior* **5**(1) (In press).

[3] Elfes A.: Using occupancy grids for mobile robot perception and navigation. *IEEE Computer Magazine, Special Issue on Autonomous Intelligent Machines* (Jun 1989).

[4] Fleury S., Souères P., Laumond J.P., Chatila R.: Primitives for smoothing mobile robot trajectories. *IEEE Trans. on Robotics and Automation* **11**(3) (Jun 1995) 441–448.

[5] Fritzke B.: Growing cell structures—a self-organizing network for unsupervised and supervised learning. *Neural Networks* **7**(9) (1994) 1441–1460.

[6] Fritzke B.: Growing grid—a self-organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letters* **2**(5) (1995) 9–13.

[7] Kohonen T.: The self-organizing map. *Proceedings of the IEEE* **78**(9) (Sep 1990) 1464–1480.

[8] Kortenkamp D.: *Cognitive Maps for Mobile Robots: A Representation for Mapping and Navigation.* Ph.D. thesis, Computer Science and Engineering, The University of Michigan, (1993).

[9] Kröse B.J.A., Eecen M.: A self-organizing representation of sensor space for mobile robot navigation. In: *Proc. IEEE/RSJ/GI Int. Conf. on Intelligent Robots and Systems*, vol. 1, Dept. of Math. & Comput. Sci., Amsterdam Univ., Netherlands. IEEE, New York, NY, USA, (1994) 9–14.

[10] Kuipers B.J., Byun Y.T.: A robust qualitative method for robot spatial learning. In: *Proc. AAAI*, (1988) 774–779.

[11] Kurz A.: Building maps for path-planning and navigation using learning classification of external sensor data. In: Aleksander I., Taylor J., eds., *Artificial Neural Networks, 2*, vol. I. North-Holland, Amsterdam, Netherlands, (1992) 587–590.

[12] Matarić M.J.: Integration of representation into goal-driven behavior-based robots. *IEEE Trans. on Robotics and Automation* **8**(3) (1992) 304–312.

[13] Michel O.: *Khepera Simulator version 2.0, User Manual*. University of Nice Sophia-Antipolis, (1996).

[14] Nilsson N.J.: *Principles of Artificial Intelligence*. Springer-Verlag, New York, (1980).

[15] Sirosh J., Miikkulainen R.: How lateral interaction develops in a self-organizing feature map. In: *Proc. IEEE Int. Conf. on Neural Networks*. San Fransisco, CA, (1993).

[16] Thrun S., Bücken A.: Integrating grid-based and topological maps for mobile robot navigation. In: *Proc. AAAI*. Portland, USA, (1996) 944–950.

[17] Verschure P.F.M.J., Kröse B.J.A., Pfeifer R.: Distributed adaptive control: The self-organization of structural behavior. *Robotics and Autonomous Systems* **9** (1992) 181–196.

[18] Vlassis N.A., Sgouros N.M., Efthivoulidis G., Papakonstantinou G., Tsanakas P.: Global path planning for autonomous qualitative navigation. In: *Proc. 8th IEEE Int. Conf. on Tools with AI*. Toulouse, France, (Nov 1996).

[19] Zimmer U.R., Fischer C., von Puttkamer E.: Navigation on topologic feature-maps. In: *Proc. 3rd Int. Conf. on Fuzzy Logic, Neural Nets and Soft Computing*. Fuzzy Logic Systems Institute, Iizuka, Japan, (1994) 131–132.