31

Growing Topology Learning Self-Organizing Map

Vilson L. Dalle Mole and Aluizio F. R. Araújo
Federal Technologic University of Paraná, Federal University of Pernambuco
Brasil

1. Introduction

This chapter presents a neural model capable of learning the input space topology producing a true representation map. Such a model is an extension of GSOSM (Growing Self-Organizing Surface Map) (DalleMole & Araújo, 2008, 2010a) which was firstly developed considering the surface reconstruction problem. The GSOSM algorithm starts with an empty map and learns the surface topology from a dense cloud of points. With GSOSM, a folded surface is reproduced by a mesh of approximately equal and almost equilateral triangles whose granularity is controlled by a simple parameter. Because of its initial purpose of Surface Reconstruction, the GSOSM algorithm uses a cross product operation which is not applicable for $\mathbb{R}^n$, $n > 3$ spaces. Then, we have modified GSOSM replacing the cross product with another approach. Also, we have added an activation function to produce an unglued activation value. This makes possible some inferences without knows the value of parameter e_{max}. Then, we called the modified model as Growing Topology Learning Self-Organizing Map (GTLSOM) (DalleMole & Araújo, 2010b). Moreover, here we give a proof on GTLSOM capability to produce a topology preservation map. Thereafter, we used GTLSOM as a basic memory of perceptions in our work with mobile robots. We also described our proposed model to give to a robot the capability of learn about feasibility of navigation actions starting from contextual perceptions.
Section 2 presents a brief discussion about topology preservation meaning. Section 3 presents de GSOSM model. Section 4 presents the changes maid into GSOSM to produce the GTLSOM model. Section 5 presents our model for robot perception of obstacles and also presents some of obtained results. Our conclusions are given in Section 6.

2. Why topology learning

Consider a folded surface immersed into $\mathbb{R}^3$ space. The surface defines the topology of space comprising all surface points. This is very distant of classical math definition for vector space. However, a map describing the surface topology makes possible some inferences i.e. about the relative position of two surfaces. Pattern classification is another great area in which topology preserving maps are useful. Because a topology preserving map represents the patterns distribution subdividing the input space while representing the regions where a specific pattern has a positive probability density. Then, given a map or a set of maps describing the space topology for a set of patterns classes, the correct class of an input

sample is determined by inspecting which stored pair (pattern, class) better represent it. Therefore, topology preserving maps is useful for computational intelligence acting as a memory where patterns are stored and classified.

3. The GSOSM Model

The main goal of Growing Self-Organizing Surface Map (GSOSM) is to provide a learning approach for surface reconstruction from an unstructured point cloud producing a mesh of approximately equilateral triangles with similar areas. Its design can be characterized by: (i) an incremental mapping process in which the neighbourhood nodes are defined by a learning process; (ii) an algorithm without any dependence on error accumulators or cycle counters; (iii) a learning rule which handle input sequences presentation of similar or very distinct patterns; (iv) the capacity of produce a suitable representation for any shape; and (v) a mesh construction process which is eventually free of internal, overlapping, or false faces (DalleMole & Araújo, 2010a). The GSOSM algorithm have nine simple steps: (i) parameter setup; (ii) sample presentation; (iii) determination of closest neighbouring nodes starting from current sample; (iv) insertion of a new node; (v) weight vector update; (vi) node collapse; (vii) determination of winner connection with the current sample; (viii) insertion or substitution of a connection; and (ix) removal or swap of connections.

3.1 Initial map and parameters

The initial state of GSOSM map is an empty graph $G(S, C)$, where S is a set of nodes and C is the set of connections (edges) linking each pair of neighboring nodes. And, four parameter values needs to be specified: (i) e_{max} - The maximum local error, which is directly related to the length of connections between nodes (triangle edges); (ii) α - The learning rate; (iii) θ_{min} – specify the smallest value accepted for the maximum internal angle of a triangle; and (iv) β_{min} - The minimal correlation rate required to insert a new connection between a selected pair of nodes. This parameter constrains the insertion of a wrong connection because GSOSM employs it to identify two distinct parallel sheets of a surface.

3.2 The nodes insertion operator

The insertion of a new node aims to represent the samples belonging to non-mapped areas. GSOSM nodes distribution aims to represent the whole input space using the lowest possible number of nodes. Each node has a spherical receptive field with radius e_{max}, hence any input signal $\boldsymbol{\xi}$ within it is considered as mapped. Then a new node is inserted even an input signal $\boldsymbol{\xi}$ is determined as unmapped. The weight vector $\boldsymbol{\omega}_{s_{new}}$ of a newly node s_{new} is set equal to input signal $\boldsymbol{\xi}$. Moreover, a newly node has no initial connections because the neighbourhood relationships are established during connection learning process.

3.3 The adaptation operator

This step aims to produce fine adjustments in the map considering the topology of input space. The GSOSM adaptation schema has two self excluding adaptation steps. Given an input signal $\boldsymbol{\xi}$, find its three closest nodes $\{s_0, s_1, s_2\}$. Then, consider the plan P spanning the triangle $\mathcal{T}$ defined by the weigh vectors of these nodes, and a prism $\mathcal{P}$ defined by the three orthogonal planes intersecting P at the edges of $\mathcal{T}$. The first adaptation step is as in SOM (Kohonen, 1989) and is employed if $\boldsymbol{\xi}$ is not within $\mathcal{P}$ or if the nodes s_0, s_1 and s_2 are not interconnected.

$$\boldsymbol{\omega}_{s_0} = \boldsymbol{\omega}_{s_0} + \alpha[\boldsymbol{\xi} - \boldsymbol{\omega}_{s_0}] \tag{1}$$

Otherwise if $\boldsymbol{\xi}$ is inside of $\mathcal{P}$, the second adaptation step is employed. This step rotates the triangle $\mathcal{T}$ around an imaginary axis containing $\boldsymbol{\omega}_{s_1}, \boldsymbol{\omega}_{s_2}$ and towards $\boldsymbol{\xi}$, approximating the triangle face to the correct surface reconstruction.

$$\boldsymbol{\omega}_{s_0} = \boldsymbol{\omega}_{s_0} + \alpha\, \mathbf{P}_n\, d(P, \boldsymbol{\xi}) \tag{2}$$

where, $\mathbf{P}_n$ is the normal vector of plane P, and $d(P, \boldsymbol{\xi})$ is the distance between $\boldsymbol{\xi}$ and P.
This two step operator enables GSOSM to handle correlated samples while avoiding node overfitting**.** Whereas, if only the original SOM adaptation operator is applied, it will pulls the winner node towards the second closest node undoing the configuration of the equilateral triangles.

3.4 The merging operator

The GSOSM nodes insertion operator ensures a suitable distance between a newly node and any other node in the map. However, all nodes are subject to being repositioned by the adaptation operator. As a consequence, a repositioned node can invade the receptive field of node. Then, given an input signal $\boldsymbol{\xi}$, if the distance between the two closest nodes s_0 and s_1 is smaller than e_{max}, the nodes are merged. The resultant node s_{new} is positioned between the original positions of merged nodes.

$$\boldsymbol{\omega}_{s_{new}} = 0.5\left[\boldsymbol{\omega}_{s_0} + \boldsymbol{\omega}_{s_1}\right] \tag{3}$$

where $\boldsymbol{\omega}_{s_{new}}$ is the weight vector of resulting node.

3.5 The competitive connection hebbian learning – CCHL

GSOSM model introduced a new learning rule called Competitive Connection Hebbian Learning (CCHL). This is a substitute for CHL (Martinetz & Schulten, 1991) to learn the node-neighborhood relationships producing a triangulation. It is because surface reconstruction procedures need to reach a complete triangulation that can not be obtained using CHL (DalleMole & Araújo, 2008). This new learning rule is based on the concept of second order Voronoi diagram (Martinetz & Schulten, 1994), and it can be formally stated as: given an input signal $\boldsymbol{\xi}$ and its three closest nodes $\{s_0, s_1, s_2\}$, the pair to be connected is that forming the closest edge to $\boldsymbol{\xi}$.
Employing the CCHL rule, the winner connection could be determined considering the triangle $\mathcal{T}$ formed by the three closest nodes to the current sample $\boldsymbol{\xi}$. The line segments connecting each vertex to its incenter[1] determine the regions where the sample $\boldsymbol{\xi}$ is closest to one of its edges. Therefore, each possible connection has a Voronoi region where a sample induces its creation.

3.6 The connection insertion operator

In GSOSM, the main objective of insertion of connections is to obtain a complete triangulation producing a mesh representation with approximately equilateral triangles with similar size. Hence, given a sample $\boldsymbol{\xi}$, the CCHL rule decides the connection $c_{new}(s_i, s_j)$ to be inserted. However, the GSOSM imposes three other conditions that need to be satisfied before creating a connection: (i) The coefficient of correlation β (Equation 4)

[1] The point that is equidistant from all triangle edges.

satisfies (Equation 5); (ii) the coefficient of similarity θ (Equation 6) between s_i and s_j with respect to node s_k , $k \in [0,1,2]$, $k \neq i,j$ is greater than or equal to parameter θ_{min}; and (iii) c_{win} does not cross the Voronoi region of any other connection. The first condition verify the continuity of the input space between the nodes s_i and s_j; the second assures the formation of triangles that are approximately equilateral; and the third condition identifies crossing connections that could create overlapping triangle faces.

$$\beta = \frac{\left(\boldsymbol{\xi} - \boldsymbol{\omega}_{s_i}\right).\left(\boldsymbol{\omega}_{s_j} - \boldsymbol{\omega}_{s_i}\right)}{\left\|\boldsymbol{\omega}_{s_j} - \boldsymbol{\omega}_{s_i}\right\|^2} \quad , \quad \left\|\boldsymbol{\xi} - \boldsymbol{\omega}_{s_i}\right\| \leq \left\|\boldsymbol{\xi} - \boldsymbol{\omega}_{s_j}\right\| \tag{4}$$

$$\beta_{min} \leq \beta \leq 0.5 \qquad \beta_{min} \in [0, 0.5] \tag{5}$$

The coefficient β is the projection of input signal $\boldsymbol{\xi}$ into the connection that is to be created. As the projection of $\boldsymbol{\xi}$ approximates to the connection midpoint the probability of inserting an erroneous connection is decreased.

The parameter β_{min} specifies a minimum value for the coefficient β and controls the GSOSM capability to differentiate between two parallel surface parts with a small gap between them. The coefficient θ is the cosine of the angle between vectors $\mathbf{v} = \boldsymbol{\omega}_{s_i} - \boldsymbol{\omega}_{s_k}$ and $\mathbf{u} = \boldsymbol{\omega}_{s_j} - \boldsymbol{\omega}_{s_k}$, therefore $\theta \in [-1,1]$. The insertion of a new connection inside of a regular pentagon has to satisfy $\theta_{min} \lesssim \cos(108°) \cong -0.309$ because this is the angle between two consecutive pentagon edges. However, as θ_{min} goes to -1 the algorithm becomes inefficient allowing insertion of connections producing irregular triangles with a large variation of edge lengths which will be removed rapidly. The coefficient θ is determined by:

$$\theta = \frac{\boldsymbol{\omega}_{s_i} - \boldsymbol{\omega}_{s_k}}{\left\|\boldsymbol{\omega}_{s_i} - \boldsymbol{\omega}_{s_k}\right\|} \cdot \frac{\boldsymbol{\omega}_{s_j} - \boldsymbol{\omega}_{s_k}}{\left\|\boldsymbol{\omega}_{s_j} - \boldsymbol{\omega}_{s_k}\right\|} \tag{6}$$

The third constraint stated that c_{win} does not cross any Voronoi region of other connections. This constraint is verified using the CCHL to verify if the midpoint of c_{win} is within a Voronoi region of another connection $c_{s_k s_t}$, $t \neq i,j$. Moreover, the search is restricted to the connections emanating from s_k and going towards the mid point between $\boldsymbol{\omega}_{s_i}$ and $\boldsymbol{\omega}_{s_j}$.

3.7 The fidelity coefficient

The coefficient ϕ (Equation 7) is a measure about the fidelity representation of a connection and GSOSM employs it to determine if the connection insertion process should be aborted; or if a existent connection should be replaced by c_{win}. To select the action to be carried out, GSOSM choose the action that preserves the connection with the highest ϕ value.

$$\phi = \left(1 + |\beta - 0.5| + \frac{d(c_{s_i s_j,}, \boldsymbol{\xi})}{\left\|c_{s_i s_j,}\right\|}\right)^{-1} \tag{7}$$

with $$d\left(c_{s_i s_j,}, \boldsymbol{\xi}\right) = \left\|\left[\boldsymbol{\omega}_{s_i} + \beta\left[\boldsymbol{\omega}_{s_j} - \boldsymbol{\omega}_{s_i}\right]\right] - \boldsymbol{\xi}\right\| \text{ and } \left\|c_{s_i s_j}\right\| = \left\|\boldsymbol{\omega}_{s_i} - \boldsymbol{\omega}_{s_j}\right\|$$

where $\boldsymbol{\omega}_{s_i}$ and $\boldsymbol{\omega}_{s_j}$ are the weight vectors of nodes at the extremities of connection $c_{s_i s_j}$.
The last component of Equation 7 is the distance between the sample $\boldsymbol{\xi}$ and the connection c_{ij}, whereas the middle component ensures that samples closest to the connection midpoint have a ϕ value greater than other ones. The constant first term is used to avoid numeric ill condition only. Thereby, the coefficient ϕ represents the connection fidelity as a direct function of samples that are captured by the Voronoi region of the connection. Moreover, the stored ϕ value is updated as the connection is a winner, therefore maintaining even the highest value.

3.8 The connection removal operator

The removal operator aims to eliminate long connections which combined with suitable connections produce non-equilateral triangles. Another objective is to eliminate cross connections which produce overlapping triangles in the mesh. Therefore, this is a two steps operator. The first step, is based on the connection removal operator of the ITM model (Jockusch & Ritter, 1999) and removes long connections. GSOSM considers the nodes s_i and s_j at extremities of connection c_{win} and removes the connections $c_{s_i s_t}$ satisfying the Equation 8 (except $c_{s_i s_j}$). In GSOSM, the parameter θ_{min} replaces the fixed value $\frac{\pi}{2}$ of the ITM model.

$$\frac{\boldsymbol{\omega}_{s_i} - \boldsymbol{\omega}_{s_j}}{\left\|\boldsymbol{\omega}_{s_i} - \boldsymbol{\omega}_{s_j}\right\|} \cdot \frac{\boldsymbol{\omega}_{s_k} - \boldsymbol{\omega}_{s_j}}{\left\|\boldsymbol{\omega}_{s_k} - \boldsymbol{\omega}_{s_j}\right\|} < \theta_{min} \ , \tag{8}$$

If the removal of a long connection results in a lozenge without any diagonal, GSOSM uses a swap operator ensuring a complete local triangulation. To couple with the surface folds, the whole process is aborted if the swap results in a connection whose value of coefficient ϕ is smaller than $\phi_{c_{s_i s_k}}$.
The second step employ CCHL rule to find and remove a cross connection, eliminating overlapping triangle faces. It is carried out using the CCHL to verify if the midpoint of c_{win} is within another connection Voronoi region (DalleMole & Araújo, 2010a). If an intersection is detected, the connection with the smallest ϕ value is removed.

3.9 GSOSM algorithm

The algorithm of the GSOSM is as follow.

1. Give the values to the parameter set and initialize an empty map;
2. While the terminate condition is not satisfied
   2.1. Sample a new stimulus $\boldsymbol{\xi}$;
   2.2. Run the node insertion operator as in Subsection 3.2;
   - If a new node is inserted or the map size is lower than three nodes, go to Endwhile;

   2.3. Construct the neighborhood set $N(\boldsymbol{\xi})$, the three closest nodes of $\boldsymbol{\xi}$;
   2.4. Run the adaptation operator as in Subsection 3.3;
   2.5. Run the merging operator as in Subsection 3.4;
   - If the pair s_0, s_1 is merged, go to Endwhile;

   2.6. Run CCHL to determine the winner connection c_{s_i,s_j} to current input signal $\boldsymbol{\xi}$ as in Subsection 3.5;

2.7. If a winner connection c_{s_i,s_j} is identified
- Determine or actualize the value of coefficient ϕ as in Subsection 3.7;
- Run the connection insertion operator as in Subsection 3.6;
- If a new connection was inserted, go to Endwhile;
- Run the connection removal operator as in Subsection 3.8;

End_while;
3. End.

4. The GTLSOM model

Basically, the GTLSOM model is an extension of GSOSM model which was initially conceived for surface reconstruction. GTLSOM introduces two little changes turning the model suitable to work with any entry space. The first one includes an adjustment of Equation 2, because the cross product used in GSOSM is not defined for $\mathbb{R}^n$, $n > 3$. The second includes an activation function to represent the activation of nodes as a value ungrounded to parameter e_{max}. These changes are detailed bellow.

4.1 Adjusting the second adaptation atep

The second step of adaptation of GSOSM algorithm (Equation 2) performs a triangle rotation. This step requests the computation of a normal vector considering the plane formed by the three closest nodes to input signal $\boldsymbol{\xi}$. Commonly, a normal vector is obtained employing the cross product operator. However, this is not defined for a space $\mathbb{R}^n$, $n > 3$, then we used the method proposed by Sloughter (2001) and so the following text is based on its work.

In order to get an suitable orthogonal vector, given the input signal $\boldsymbol{\xi}$ and its three closest nodes s_0, s_1 and s_2 and the respective model vectors $\boldsymbol{\omega}_{s_i}$, $i = 0,1,2$. If vectors $\boldsymbol{\omega}_{s_i}$ are linearly independent they determine a plane P. Then determine the vectors $\mathbf{v}$ and $\mathbf{w}$ lien in the plane P as

$$\mathbf{v} = \boldsymbol{\omega}_{s_1} - \boldsymbol{\omega}_{s_0} \text{ , } \mathbf{w} = \boldsymbol{\omega}_{s_2} - \boldsymbol{\omega}_{s_0} \tag{9}$$

and the parametric equation of plane P as

$$\mathbf{y} = t\mathbf{v} + s\mathbf{w} + \boldsymbol{\omega}_{s_0}. \tag{10}$$

In a general sense, given a plane P with equation $\mathbf{y} = t\mathbf{v} + s\mathbf{w} + \mathbf{p}$, if we find two vectors $\mathbf{a}$ and $\mathbf{b}$ lying in P such that $\mathbf{a} \perp \mathbf{b}$ with $\|\mathbf{a}\| = 1$ and $\|\mathbf{b}\| = 1$, becomes that equation $\mathbf{y} = t\mathbf{a} + s\mathbf{b} + \mathbf{p}$ describes the same plane P. Therefore, taking $\mathbf{c}$ as the projection of $\mathbf{w}$ into $\mathbf{v}$ the vectors $\mathbf{a}$ and $\mathbf{b}$ can be wrote as

$$\mathbf{a} = \frac{1}{\|\mathbf{v}\|}\mathbf{v}, \tag{11}$$

$$\mathbf{b} = \frac{1}{\|\mathbf{w} - \mathbf{c}\|}(\mathbf{w} - \mathbf{c}). \tag{12}$$

Moreover, given a vector $\mathbf{q}$ not in P, let $\mathbf{r}$ be the sum of the projections of $\mathbf{q}$ onto $\mathbf{a}$ and $\mathbf{b}$,

$$\mathbf{r} = (\mathbf{q}.\mathbf{a})\mathbf{a} + (\mathbf{q}.\mathbf{b})\mathbf{b}. \tag{13}$$

Without lost of generallity, consider the plane P through the origin. It follows that for any $\mathbf{y} = t\mathbf{a} + s\mathbf{b} + \mathbf{p}$ in plane P, becomes

$$(\mathbf{q} - \mathbf{r}).\mathbf{y} = (\mathbf{q} - \mathbf{r}).(t\mathbf{a} + s\mathbf{b}) = t(\mathbf{q} - \mathbf{r}).\mathbf{a} + s(\mathbf{q} - \mathbf{r}).\mathbf{b} = 0, \tag{14}$$

that is, $\mathbf{q} - \mathbf{r}$ is orthogonal to every vector in the plane P. Returning to the initial problem, all we need is to find $\mathbf{r}$ and then doing $\mathbf{P}_n = \frac{\boldsymbol{\xi} - \mathbf{r}}{\|\boldsymbol{\xi} - \mathbf{r}\|}$ and $d(P, \boldsymbol{\xi}) = \|\boldsymbol{\xi} - \mathbf{r}\|$, the Equation 2 becomes

$$\boldsymbol{\omega}_{s_0} = \boldsymbol{\omega}_{s_0} + \alpha(\boldsymbol{\xi} - \mathbf{r}). \tag{15}$$

4.2 Setting up an activation function

We have considered a Gaussian function as a suitable function to express the node activation in terms of its similarity to the input signal $\boldsymbol{\xi}$. It is because the output of this function type is constrained into interval [0;1]. Moreover, we can control its radial base enforcing the value of σ as need. Then, setting $\sigma = e_{max}$, the equation of activation level of a node s_i becomes.

$$a_{s_i} = exp\left(-\frac{\left\|\boldsymbol{\xi} - \boldsymbol{\omega}_{s_i}\right\|^2}{2.{e_{max}}^2}\right), \tag{16}$$

where a_{s_i} is the activation of node s_i, $\|\cdot\|$ is the Euclidean distance.
Note that using Equation 16, a value of $a_{s_i} \cong 0.6$ is obtained when the input signal is located near the boundaries of receptive field of node s_i. Moreover, the relative position of $\boldsymbol{\xi}$ and its three closest neighbouring nodes can be estimated using the information provided by a_{s_i}.

4.3 Topology preservation

In the common sense, the topology preservation is understood as the degree to which a map preserves the information about proximity between points and about the continuity of entry space. The topology preservation implicates into the equivalence between the generated map and the entry space for all represented points and its neighbourhoods (Vilmann, 1999).
Definition 1: A graph G together its nodes i associated to points $\mathbf{w}_i$ forms a topology preserving map of a entry space (*manifold*) Ω, if exists a mapping function $\varphi: \Omega \longrightarrow G$ leading neighbour signals in Ω to neighbour nodes in G; and its inverse mapping $\varphi^{-1}: G \longrightarrow \Omega$ leading nodes that are neighbour in G to neighbour places in Ω (Martinetz, 1994).
From insertion operator (Subsection 3.2) each input signal $\boldsymbol{\xi}$ is classified to a node $s_i \in S$ with a receptive field, or produces the insertion of a new node s_{new} with a weight vector $\boldsymbol{\omega}_{s_{new}} = \boldsymbol{\xi}$. Therefore, the first condition is directly satisfied by the GTLSOM mapping process:

$$\varphi: \Omega \longrightarrow S \mid \varphi(\boldsymbol{\xi}_k) = s_i,\ s_i \in S,\ \forall \boldsymbol{\xi}_k \in \Omega. \tag{17}$$

The inverse mapping φ^{-1} is also directly verified because a new node s_{new} is created setting $\boldsymbol{\omega}_{s_{new}} = \boldsymbol{\xi}$:

$$\varphi^{-1}: S \longrightarrow \Omega \mid \varphi^{-1}(s_i) = \boldsymbol{\xi}_k, \qquad \forall s_i \in S \rightarrow \boldsymbol{\xi}_k \in \Omega. \tag{18}$$

Moreover, using CCHL the connection insertion operator is fired only if an input signal falls between two nodes. Therefore, the connection represents the input space continuity. Then, the existence of mapping functions φ and φ^{-1} plus CCHL connection learning rule ensure the topology preservation.

5. Robot obstacle perception and safe navigation learning

We have used GTLSOM to enables a robot to learn about its surrounding environment while identifying the presence of obstacles or empty space. We used a GTLSOM map as a basic memory for robot perceptions because its capabilities of storing and clustering similar inputs. Whereas a learning schema was employed to produce synapses linking the robot perceptions to navigation actions. The main objective was learning about feasibility of each possible navigation action, starting from a remembering process of context similarities. The cognitive process recovers the knowledge stored into synapses producing a value judgment to subsidize the robot behavior. In other words, the robot learns about the feasibility of each action into the perceived surrounding environment context. Once a local environment has been explored the assimilated knowledge remains available and the robot makes use of it to navigate into unknown environments. Moreover, the stored knowledge is continuously refined and updated. Lastly, we developed a layered schema (Fig. 1) to accommodate the subsystems in a modular way.

The first layer is responsible for sense the surrounding environment producing perceptions. The second layer is a GTLSOM map grouping an storing perceptions producing a map, a basic memory, $\mathcal{M}^p$, for perceptions. At layer 3 a learning process conjugates information about recollected memories, selected actions and motor feedbacks, producing and modifying synapses between memories and actions (Fig. 2). Another process uses the knowledge stored into the synapses to produce a value judgment about action feasibility. The process at layer 4 considers the produced judgments to select the action to be carried out and signal it to motor subsystem. The motor subsystem accepts, as entry, a vector specifying the action command which is executed producing a feedback signal for Synapse Learning module.

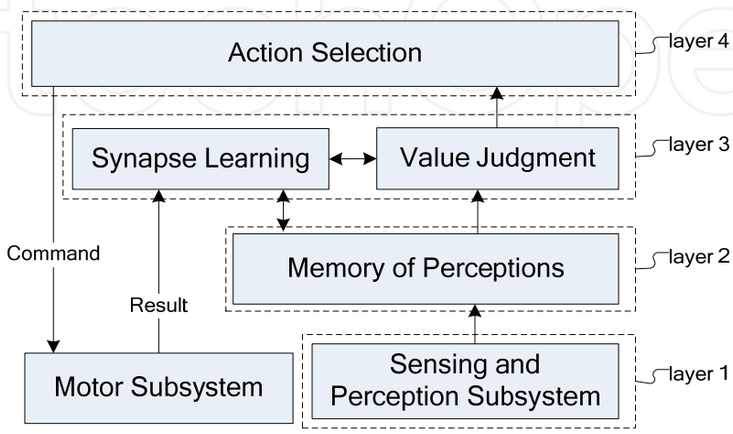


Fig. 1. Diagram of blocks of layered schema for robot navigation system

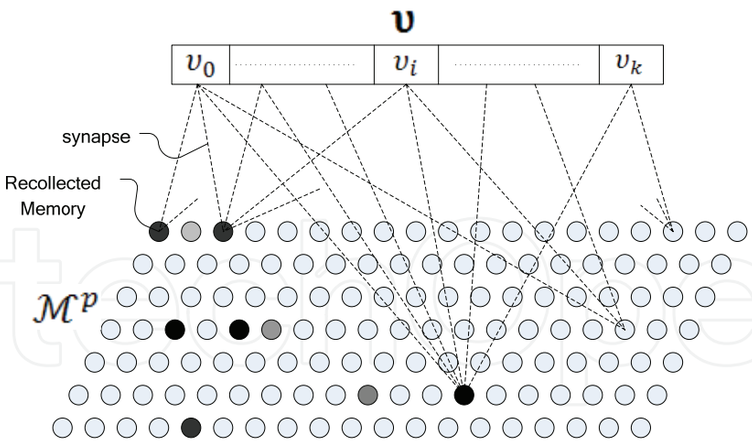


Fig. 2. Memory of perceptions and the synapses between actions $\boldsymbol{v_i}$ and memory units

5.1 Sensing and perception subsystem

We have considered an internal and structured environment composed of rooms, corridors and passages between them. And a robot with a body shaped like a box having six laser based sensor devices one on each lateral face one on top and other on bottom face. Each sensor device s_i produces a flow $\mathcal{F}^{s_i}$ of reads $\mathbf{x}_t^{s_i}$ sweeping its laser spot by 180º stepping no more that 10º between reads. The perception module uses the vectors of sensor reads recovering a environment semi contour (Fig. 3) surrounding the robot for each instant t.

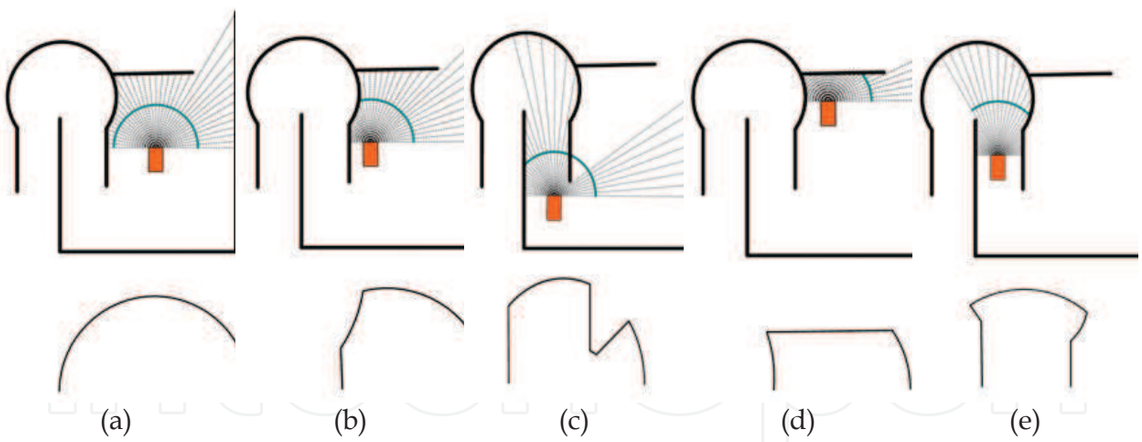


Fig. 3. Samples of surrounding contour perception produced using reads from frontal sensor device. The circular line crossing the lines of reads indicates the prunning distance

The sensor reads are pruned (Equation 19) so that only the immediate surrounding environment is considered as specified by parameter $d_{\max}$. Then, each perception p is codified into a vector $\mathbf{w}_i \in \mathbb{R}^{\hbar}$ using $\hbar$ first geometric moments of Zernick (1934). Firstly the surrounding contour is plotted into a circular bitmap image with radius equals to value of parameter r and then the values of geometric moments are determined. Therefore, a perception represents a contour image as illustrated in Fig. 3 and is the basic unit of perceptive flows.

$$\mathbf{x'}_t^{s_i} = [x'_0, \dots, x'_i, \dots], x'_i = (\mathbf{u}_i, d'_i),\ d'_i = \begin{cases} d_i & if\ d_i \le d_{max} \\ d_{max} & if\ d_i > d_{max} \end{cases}, \tag{19}$$

where $\mathbf{u}_i$ is the direction vector of read; d_i and d'_i are, respectively, the measured and pruned distance; and $\mathbf{x'}_t^{s_i}$ is the vector of pruned reads.
This representation schema preserves the image discretionary capability while enables direct comparisons employing Euclidean distance. Moreover, it was proved effective and under determined conditions could reduce the space of configurations to a finite number. Such reduction could be reached because the pruning step and the use of a finite number of geometric moments and because GTLSOM groups similar inputs with a receptive field.

5.2 Synapse learning

This module knows which units of memory were activated by perceptions coming from each sensor device, as well as actions taken based on them and the feedback received from motor subsystem. Such information is employed for synapse establishment or adjustment. The actions are represented by a vector $\boldsymbol{\upsilon} \in \mathbb{R}^k$ with k being the degrees of freedom of robot. For each possible action (*go_ahead, go_back, turn_left, turn_right and soon)* a set of synapses is constructed and the set of all synapses and its stored values forms a base of knowledge about actions feasibility for each local environment context. Each synapse is valued into an interval [0,1] with initial value being given by parameter μ, which specify the minimum value to an action be considered as feasible. Therefore, learning occurs by adjusting synapse value through a schema of punishment and reward (Kaebling *et al*., 1996); (Sutton & Barto, 1998).
The synapse adjustment consider the coefficients represented by parameters α_r and α_p and affect only the synapses $s_{l_i v_j}$ between the recollected memory l_i and the associated action υ_j. The adjustments are determined using Equation (20) for successful actions and Equation (21) for those producing unwanted results i.e. a collision with an obstacle.

$$s'_{l_i v_j}(t') = \begin{cases} s_{l_i v_j}(t') + \alpha_r^{t'}\left(1 - s_{l_i v_j}(t')\right)\ if\ sucess \\ s_{l_i v_j}(t') + \alpha_p^{t'}\left(0 - s_{l_i v_j}(t')\right)\ if\ failure \end{cases},\ \forall t' \in]t - \tau, t], \tag{20}$$

with $\alpha_r^t = \alpha_r,\ \alpha_r^{t-1} = 0.5\alpha_r^t$
$\alpha_p^t = \alpha_p,\ \alpha_p^{t-1} = 0.5\alpha_p^t$,

$$s'_{l_i v_j}(t) = s_{l_i v_j}(t) + \alpha_p^t\left(0 - s_{l_i v_j}(t)\right),\ \alpha_p^t = \alpha_p, \tag{21}$$

where $s_{l_i v_j}$ is the synapse value at time t and $s'_{l_i v_j}$ is the synapse value after adaptation step; υ_j is the $j-$ésima component of $\boldsymbol{\upsilon}$ and represents an possible action; α_r and α_p are paramters representing the rates of reward and punishment respectively, $\alpha_r, \alpha_p \in [0; 1]$; t is the current time instant and t' is the time instant in which the action was executed; and τ is a parameter controlling the number of time instants to back propagation of adjustments for a selected action executed successfully.
Therefore, the adjustment affects positively the η stronger synapses, if the action is executed successfully. And negatively the synapses that subsidized the selection of an action with unwanted results.

5.3 Action value judgment

For each time instant a value judgment is determined for all possible actions. Then, for each component v_j of $\mathbf{v}$ this process considers the average of values stored in the synapses between it and the first λ activated units of memory. Therefore, the output of this process is a k dimensional vector of values into interval [0; 1] representing the robot knowledge about the feasibility of each navigation action into the current context. As mentioned early, a parameter μ is used to signal a minimum value for an action to be considered as feasible.

5.4 Algorithm

1. Get the sets P^{s_i} of current perceptions from flows $\mathcal{F}^{s_i}$;
2. Present each set P^{s_i} to the basic memory $\mathcal{M}^p$ getting the respective list $\mathcal{L}^{s_i}$ of activated units;
3. For each component action $v_j \in \mathbf{v}$, produce a value judgment considering the respective list $\mathcal{L}^{s_i}$
   - 3.1. Sort $\mathcal{L}^{s_i}$ according synapses values;
   - 3.2. Determine the value judgment of v_j as the average of first λ synapses in $\mathcal{L}^{s_i}$;
   - 3.3. Registry the first λ synapses into the historic list ℓ of value subsidy;
4. Signal the value judgment for the Action Selection module;
5. For each navigation action taken
   - 5.1. Catch the feedback signal (**SUCESS** / **UNWANTED**);
   - 5.2. Update the historic list l of last τ navigation actions taken;
   - 5.3. If the status of action $l(t)$ indicates **SUCESS** employ the adjustment schedule defined by Equation (20).
     - 5.3.1 Initialize $t' = t$, $\alpha_r^{t'} = \alpha_r$ and $\alpha_p^{t'} = \alpha_p$;
     - 5.3.2 For each action in l, consider the synapses registered in $\ell(t')$
       - If **REWARD**, adjust the values of η less pessimistic synapses;
       - If **PUNISHMENT**, adjust the values of λ more pessimistic synapses;
       - Decrement t' updating $\alpha_r^{t'}$ and $\alpha_p^{t'}$
   - 5.4. If the status of action $l(t)$ indicates **UNWANTED**
     - 5.4.1 Adjust the values of λ more pessimistic synapses in $\ell(t')$ (Equation 21);
6. Return to step 1;

5.5 Results

This section presents three sets (A, B and C) of results obtained with our model for robot obstacle perception and safe navigation learning. In test sets A and C the robot is constrained to navigate in 2D whereas set B refers to robot navigating in 3D. In set C the robot was removed from a known environment part and introduced into another similar environment part. The selection of navigation actions was subject of a probabilistic algorithm. Then the robot remain wandering and swaps the selected navigation action if it collides with an obstacle or the value judgment of current action indicates that it is unsafe.

5.5.1 Parametrization

The basic memory of perception $\mathcal{M}^p$ is a GTLSOM map then its parameter set was established setting $\hbar = 10$, $e_{max} = 20$ and $e_{max} = 40$ defining two test subsets. The parameters set for the laser based sensors was established setting $sweep\_angle = 180°$, $sweep\_step = 10°$ and $rotation\_angle = 180°$, $rotation\_step = 15°$ for the case 3D. The robot velocity was fixed as two units of measurement per time unit and two degrees per time unit for rotation velocity. The association between sensors and navigation actions is given in the Table 1.

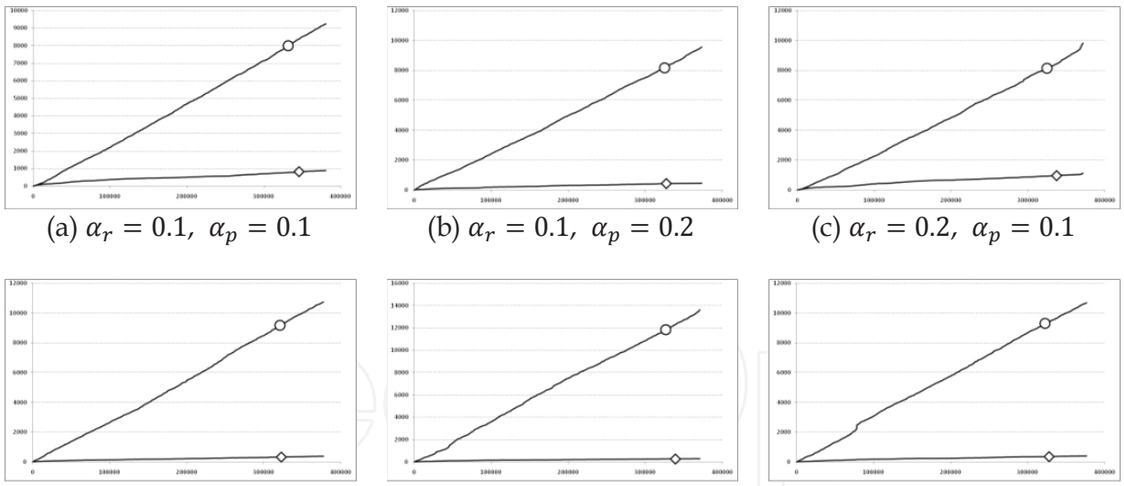
Sensor	Front	Left	Right	Back	Top	Botton
Action	go ahead	turn left	turn right	go back	float up	float down

Table 1.Sensors and Navigation Actions Association

The parameter set of our model was established setting $d_{max} = 100$, $r = 200$, $\mu = 0.6$, $\tau = 3$, $\lambda = 3$, $\eta = 1$. The coeficientes α_r and α_p were set according the test case.

5.5.2 Test set A – navigating in 2D

The graphs of Fig. 4 and Fig. 5 report the robot behaviour evolution when positioned towards an obstacle. Each graph show the relation between the number of navigation actions aborted in function of its judgment value (circular marker) and the number of navigation actions with unwanted results (lozenge marker).

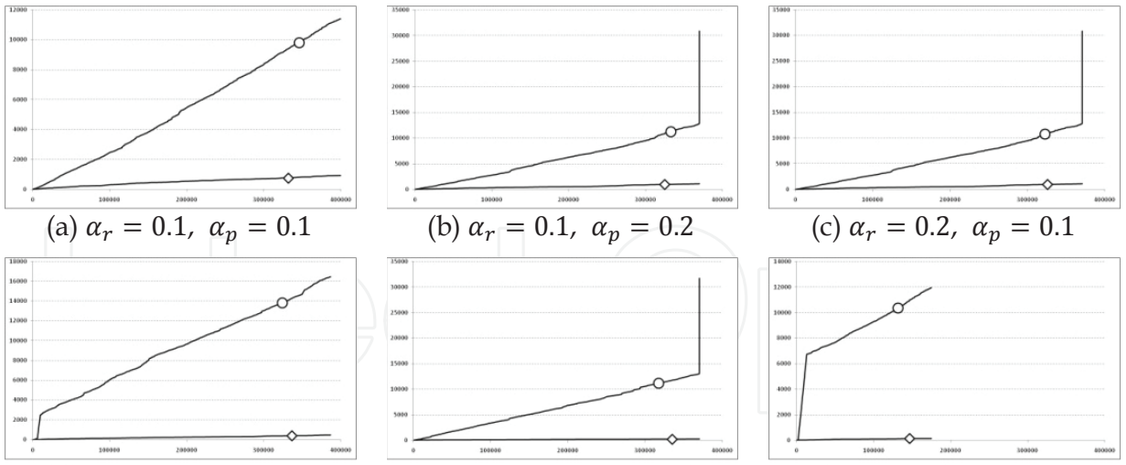


(a) $\alpha_r = 0.1,\ \alpha_p = 0.1$ (b) $\alpha_r = 0.1,\ \alpha_p = 0.2$ (c) $\alpha_r = 0.2,\ \alpha_p = 0.1$

(d) $\alpha_r = 0.3,\ \alpha_p = 0.3$ (e) $\alpha_r = 0.3,\ \alpha_p = 0.5$ (f) $\alpha_r = 0.5, \alpha_p = 0.3$

Fig. 4. Robot navigation behavior if positioned towards an obstacle and the parameter of perception memory granularity set to $e_{max}^{\mathcal{M}^p} = 20$

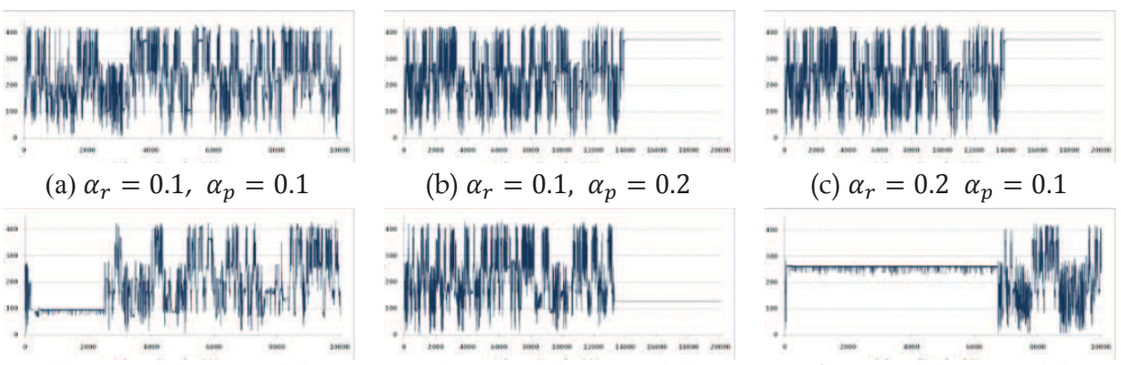
The Fig. 4 shows that the robot is learning to avoid the obstacles while maintaining a wandering behaviour. However, note the abnormalities into graphs (b, c, and e) of Fig. 5, this is because the robot stuck in a local minima where all selected navigation actions were judged as not feasible. While, the graphs (d) and (f) of Fig. 5 reveals a local minima where the robot stuck however not totally immobile, it remained alternating between antonym actions as *turn_left* and *turn_right.* This behaviour could be better visualized in Fig. 6.



(a) $\alpha_r = 0.1,\ \alpha_p = 0.1$ (b) $\alpha_r = 0.1,\ \alpha_p = 0.2$ (c) $\alpha_r = 0.2,\ \alpha_p = 0.1$

(d) $\alpha_r = 0.3,\ \alpha_p = 0.3$ (e) $\alpha_r = 0.3,\ \alpha_p = 0.5$ (f) $\alpha_r = 0.5,\ \alpha_p = 0.3$

Fig. 5. Robot navigation behavior if positioned frontwards an obstacle and the parameter of perception memory granularity set to $e_{max}^{\mathcal{M}^p} = 40$



(a) $\alpha_r = 0.1,\ \alpha_p = 0.1$ (b) $\alpha_r = 0.1,\ \alpha_p = 0.2$ (c) $\alpha_r = 0.2\ \ \alpha_p = 0.1$

(d) $\alpha_r = 0.3,\ \alpha_p = 0.3$ (e) $\alpha_r = 0.3,\ \alpha_p = 0.5$ (f) $\alpha_r = 0.5,\ \alpha_p = 0.3$

Fig. 6. Robot navigation behavior for $e_{max}^{\mathcal{M}^p} = 40$

Moreover, the wandering behaviour was re-established (Fig. 6 (d) and (f).) after a human mediation suspending the autonomous navigation and teaching some safe navigation actions and then returning to autonomous mode. Notably, the cases where the robot stuck occurred only when we set the memory granularity to a large value. It is because the robot capacity to distinguish between similar contexts was biased by memory granularity as shown Table 2.

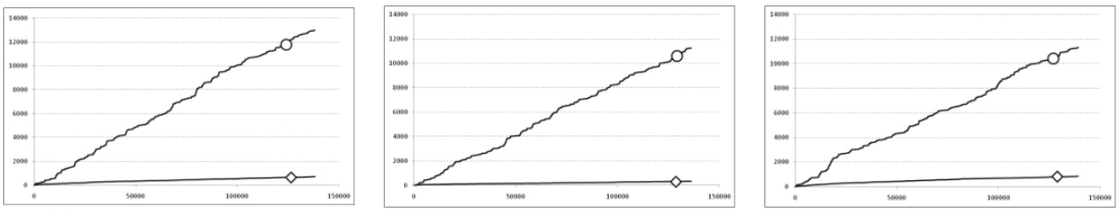
Units into Perception Memory $\mathcal{M}^p$						
$e_{max}^{\mathcal{M}^p}$	$\alpha_r = 0.1$ $\alpha_p = 0.1$	$\alpha_r = 0.1$ $\alpha_p = 0.2$	$\alpha_r = 0.2$ $\alpha_p = 0.1$	$\alpha_r = 0.3$ $\alpha_p = 0.3$	$\alpha_r = 0.3$ $\alpha_p = 0.5$	$\alpha_r = 0.5$ $\alpha_p = 0.3$
20	73,210	78,204	80,779	80,347	82,495	79,219
40	6,798	6,921	6,835	6,977	6,798	6,870

Table 2. Number of memory units used to represent the robot perceptions

The relationship between the memory granularity and robot capacity to learn to distinguish safe navigation actions from unsafe, becomes clear confronting the data reported in Table 2 with the graphs in Fig. 4 and Fig. 5. Then found a suitable value for parameter e_{max} of GTLSOM is a main job to obtain a suitable memory of perceptions. Moreover, the presented results indicates high values of α_p weaken the synapses so that the algorithm is unable to recover them and the learning process stuck in a local minima. Whereas setting up α_p values lightly greater than α_r seem to produce better results (Fig. 4 (b) and (e)).

5.5.3 Test set B – navigating in 3D

For this test set, the robot was released to navigates in 3D employing actions to floating up and down besides that used to navigate in 2D. Then we have also considered the perceptions originated from sensors on Top and Bottom of robot. The parameter of memory granularity was set as $e_{max}^{\mathcal{M}^p} = 20$ because of the results reported in Subsection 5.5.2. The parameters α_r and α_p were set using the pairs (0.1; 0.1), (0.1; 0.2) and (0.2; 0.1). Other parameters were set as in Subsection 5.5.2 and the time of test was established as 150.000 attempts of navigation. The Fig. 7 shows that results are similar to those obtained with robot constrained to navigate in two dimensions (Fig. 4).



(a) $\alpha_r = 0.1,\ \alpha_p = 0.1$ (b) $\alpha_r = 0.1,\ \alpha_p = 0.2$ (c) $\alpha_r = 0.2,\ \alpha_p = 0.1$

Fig. 7. Robot navigation behavior if positioned frontwards an obstacle while navigating in 3D

5.5.4 Test set C – navigating into similar environments

In this test set, the objective was asset the knowledge reuse to navigate in another similar environment. Firstly, the robot was placed into an initial environment and left to acquire navigation skills learning about contexts and the feasibility of navigation actions. Further, the robot was removed and reinserted into a similar environment for two times without restart its memory. The parameter set was established with $e_{max}^{\mathcal{M}^p} = 20,\ \alpha_r = 0.1, \alpha_p = 0.1$ and others were set with values used in test set A (Subsection 5.5.2). The time of each phase was set to one hundred thousand navigation actions. The Fig. 8 shows percent of collisions for each phase. Note the spikes following the reinsertion indicating that the robot found unknown contexts which are learned rapidly. Moreover, the percent of collisions remains decreasing indicating the success of knowledge reuse.

6. Conclusion

We have discussed briefly the issue of topology learning and topology preserving maps and its applicability to surface reconstruction and to computational intelligence. Thereafter we have presented the GSOSM model which was developed considering the surface reconstruction problem. Then we have updated GSOSM algorithm to work with space $\mathbb{R}^n$,

$n \geq 3$. Specifically we have modified the adaptation step by introducing a method to obtain a suitable $\mathbb{R}^n$ normal vector. Moreover, we have added an activation function to report the nodes activation in a normalised way. Then we called the resulting model as Growing Topology Learning Self-Organizing Map (GTLSOM) and we attempt to give a proof of its topology preservation capability.

Further, we have used GTLSOM as a tool to produce a basic memory which was employed to enable a robot with a remembrance capability. Basically, in this aplication GTLSOM acts as memory by constructing a map grouping patterns into nodes with a receptive field (units of memory). In our model the memory inputs are representations of robot perceptions about its surrounding environment. The knowledge about contextual feasibility of each navigation action is stored into synapses linking units of memory and actions. Then, the incoming perceptions are classified to nodes firing a remembrance process that subsidizes a value judgment for each possible action to be selected. We also have reported the obtained results which are indicatives of success of our model. Moreover note that in our model, the basic memory does not know the origin of each perception as well as what happens out of its boundaries. Therefore, the basic memory could be shared with other cognitive process.

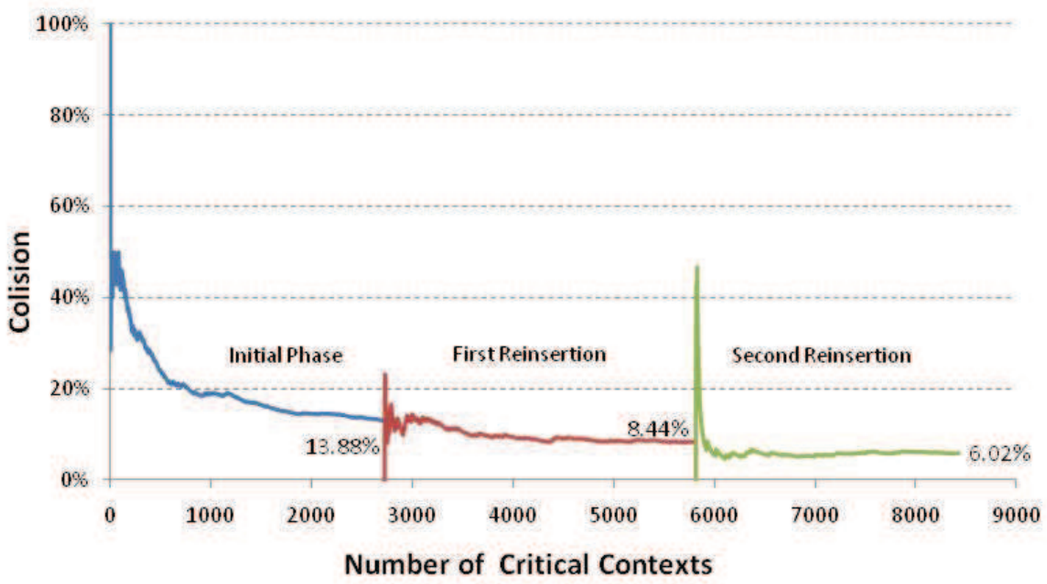


Fig. 8. Robot navigation behavior if moved from an environment to other similar one

7. References

DalleMole, V. L. & Araújo, A. F. R. (2010a) Growing Self-Organizing Surface Map: Learning a Surface Topology from a Point Cloud, Neural Computation, Vol. 22, No. 3, pp. 689-729.

DalleMole, V. L. & Araújo, A. F. R. (2010b) A Novel Topological Map of Place Cells for Autonomous Robots, To appear in: *Proceedings of 20th International Conference on Artificial Neural Networks*, Thessaloniki, Greece, September 15-18, 2010, Springer Academic Publishers.

DalleMole, V. L. & Araújo, A. F. R. (2008) The Growing Self-Organizing Surface Map, *Proceedings of International Joint Conference on Neural Networks IJCNN´08*, pp. 2061 – 2068, ISBN: 978-1-4244-1820-6, Hong Kong.

Jockusch, J. & Ritter, H. (1999). An instantaneous topological mapping model for correlated stimuli. In: *International Joint Conference on Neural Networks IJCNN'99*, Vol. 1, pp. 529-563, ISBN: 0-7803-5529-6, Washington.

Kaebling, L. P. Littman, M. L. & Moore, A. W. (1996) Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*,Vol 4, pp. 237-285.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. In: *Biological Cybernetic,* Vol. 43, pp. 59-69.

Martinetz, T., Schulten, K. (1994) Topology Representing Network. Neural Networks, Vol. 7, No. 3, pp. 507-522.

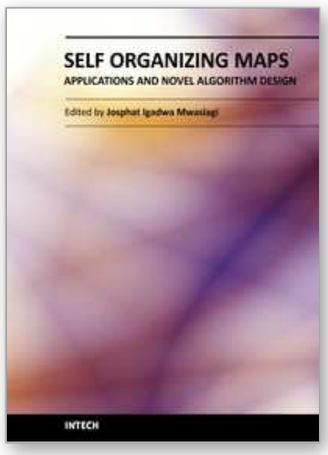
Martinetz, T.; Schulten, K. (1991). A "Neural-Gas" Network Learns Topologies. In: *Artificial Neural Networks*, T. Kohonen, K. Makisara, O. Simula and J. Kangas, Elselvier Science, pp. 397- 402, North-Holland.

Sloughter, D. (2001) *The Calculus of Functions of Several Variables*, Furman University, 260p.

Sutton, R. S. & Barto, A. G. (1998) *Reinforcement Learning: An Introduction*. Mit Press.

Vilmann, T. (1999) Topology preservation in self-organizing maps, In: *Kohonen maps*, Erkki Oja and Samuel Kaski, editors, Elselvier Science, pp. 279-291.

Zernike, F. (1934) *Physics*, pp. 1- 689.



Self Organizing Maps - Applications and Novel Algorithm Design

Edited by Dr Josphat Igadwa Mwasiagi

ISBN 978-953-307-546-4
Hard cover, 702 pages
Publisher InTech
Published online 21, January, 2011
Published in print edition January, 2011

Kohonen Self Organizing Maps (SOM) has found application in practical all fields, especially those which tend to handle high dimensional data. SOM can be used for the clustering of genes in the medical field, the study of multi-media and web based contents and in the transportation industry, just to name a few. Apart from the aforementioned areas this book also covers the study of complex data found in meteorological and remotely sensed images acquired using satellite sensing. Data management and envelopment analysis has also been covered. The application of SOM in mechanical and manufacturing engineering forms another important area of this book. The final section of this book, addresses the design and application of novel variants of SOM algorithms.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen