Technical Report OUCS-2018-03

A neural network model for learning to represent 3D objects via tactile exploration: technical appendix

Authors:

Xiaogang Yan, Alistair Knott, Steven Mills
Department of Computer Science, University of Otago, New Zealand

A neural network model for learning to represent 3D objects via tactile exploration: technical appendix

Xiaogang Yan, Alistair Knott, Steven Mills
yanxg@cs.otago.ac.nz; alik@cs.otago.ac.nz; steven@cs.otago.ac.nz

Abstract

This technical report is to present a neural network model for learning to represent 3D objects via tactile exploration, which complements a paper named "a neural network model for learning to represent 3D objects via tactile exploration" and published in the conference CogSci 2018. This report includes the details of the proposed model.

1 Introduction

How brains represent 3D objects remains unsolved though loads of effort has been spent. Obviously, brains can learn to represent 3D objects via different kinds of sensory information, like vision, hearing and touching, etc. Most researchers tend to study this tricky problem from the perspective of vision, while we aim to investigate such a problem from touching sensory information for several reasons. Firstly, the tactile information is the primary and the first source obtained when an infant starts to learn the world. Secondly, the blind people also can distinguish the objects trough touching. Thirdly, the circuit of vision about representing objects in brains is far well-established compared with that of touching. Therefore, it might be possible to use this model to shed light on what the circuit of touching could be.

Specifically, the model is to learn the topographical feature of the 3D object which a navigating agent explores. This model is analogous to the navigation model of mammals, which can learn an *allocentric* representation of the exploring environment via the *egocentric* movements [Moser et al., 2008]. By assuming the environment as surfaces of 3D objects and the navigating agent's movements as hand's movements, representations of the environment can be construed as representations of the 3D object. Firstly, the navigating agent (supposing the hand in this model) takes random movements to explore the object. Then, the perceptual information during the exploration, such as the texture and associated temperature information, together with the conducted action movements is inputted to a neural network. The network is like the neurons in brains, which store the representations of such an object. Based on the acquired representations, the agent can estimate its location as well as its orientation with a distributed probability and also guess the next

```
Data: Input dataset, assuming input pattern $x(t) \in \mathcal{R}^m$
Result: A convergent MSOM
Initialization: Randomly initialize all regular weights at time instance 0
$w_i(0) \in (0,1)$ and set all context weights $c_i(0) = 0$, $i = 1, 2, \ldots n^2$;
while feature map is not convergent do
    1. Sampling: Draw sample input $x(t) \in \mathcal{R}^m$;
    2. Competition: Find best matching unit based on a distance discriminant
       function:
          $f(x) = \operatorname*{argmin}_i \ (1-\xi)\|x(t) - w_i\|_2^2 + \xi\|c(t) - c_i\|_2^2,$
       where $c(t) = (1-\kappa)w^*(t-1) + \kappa c^*(t-1)$, $\xi \in (0,1)$ $\kappa \in (0,1)$;
    3. Cooperation: Neurons selected by a decreasing neighbourhood function
       $\mathcal{H}(i, f(x))(t)$ excite;
    4. Adaptation: Update regular weights and context weights of all excited
       neurons:
          $w_i(t+1) = w_i(t) + \mathcal{L}(t)\mathcal{H}(i, f(x(t)))(t)(x(t) - w_i(t)),$
          $c_i(t+1) = c_i(t) + \mathcal{L}(t)\mathcal{H}(i, f(x))(t)(c(t) - c_i(t)),$
       where $\mathcal{L}(t)$ is a time-varying monotonically decreasing learning rate function.
end
```

Algorithm 1: Description of MSOM

possible action that can be performed. Based on the probability distribution of possible actions, the adjusted most probable action is chosen as the next action, which then update the input to the neural network as well as the representation.

2 Modified Self-Organizing Map (MSOM)

Inspired from the biological observation that the information is processed in brains with a topological fashion, the self-organising map (SOM) is developed in [Kohonen, 1982]. Specifically, a SOM is to learn input patterns like a normal neural network and meanwhile incorporate the topology feature involved. Apart from its biologically supporting evidence, the SOM has a lot of other advantages for tricky problems in loads of domains, such as dimension reduction, stock prediction, data mining and manipulator control [Yin, 2008b, Kohonen, 1998, 1990, Ritter et al., 1992, Kaski, 1997, Haykin, 1999, Yin, 2002, 2008a, Leoni et al., 1998], etc.

Though there are some advantages in SOM, it could not include the temporal information, while the time-series data are normally encountered in practice. Therefore, a modified self-organizing map (MSOM) is developed in [Strickert and Hammer, 2005]. Considering that the action sequences are constrained by the object's geometry feature and implicitly include the temporal information, the MSOM is employed to learn 3D object representations via the performed action sequences. The input of MSOM includes two parts: one is the current input and the other one is the previous state input.

Regarding an input $x(t) \in \mathcal{R}^m$ at time instance t, the activity of unit i of a $n \times n$ MSOM at that time instance is defined as

$$a_i(t) = \exp(-\eta d_i(t)), \tag{1}$$

where $i \in 1, 2, \cdots, n^2$, $\eta > 0$ is a design parameter, and $d_i(t)$ is a distance function formulated as

$$d_i(t) = (1 - \xi)\|x(t) - w_i(t)\|_2^2 + \xi\|c(t) - c_i(t)\|_2^2, \tag{2}$$

in which $\xi \in (0, 1)$ is a weight factor, $\|\cdot\|_2$ denotes the 2-norm of a matrix or vector, $w_i(t)$ is the regular weight and $c(t)$ is the context weight. The context weight $c(t)$ in (2) is

$$c(t) = (1 - \kappa)w^*(t - 1) + \kappa c^*(t - 1), \kappa \in (0, 1), \tag{3}$$

where $w^*(t - 1)$ and $c^*(t - 1)$ denote the regular weight and context weight of the unit in MSOM with the maximal activity $a_i(t)$ at previous time instance $t - 1$, respectively. In addition, by norming the activities of all MSOM units shown in (1),

$$p_i(t) = \frac{a_i(t)}{\sum_{j=1}^{n^2} a_j(t)}, \tag{4}$$

which denotes the activity probability of unit i for the current input at time instance t. During training, the regular weight $w_i(t)$ is updated as

$$w_i(t + 1) = w_i(t) + \mathcal{L}(t)\mathcal{H}(i, f(x(t)))(t)(x(t) - w_i(t)), \tag{5}$$

and the context weight $c_i(t)$ is changed as

$$c_i(t + 1) = c_i(t) + \mathcal{L}(t)\mathcal{H}(i, f(x))(t)(c(t) - c_i(t)), \tag{6}$$

where $\mathcal{L}(t)$ is a time-varying monotonically decreasing learning rate function, $\mathcal{H}(i, f(x(t)))(t)$ is a time-varying monotonically decreasing neighbourhood function with $f(x)$ denoting the index of the unit in MSOM with the maximal activity for the current input $x(t)$. At the beginning of training, the regular weight $w_i(0)$ is randomly selected between $(0, 1)$ and the context weight $c_i(0) = 0$. For illustration, the process of MSOM is shown in Algorithm 1. Note that MSOM is a convergent algorithm and therefore after the training, the map becomes to be stable, which makes it usable in implementations.

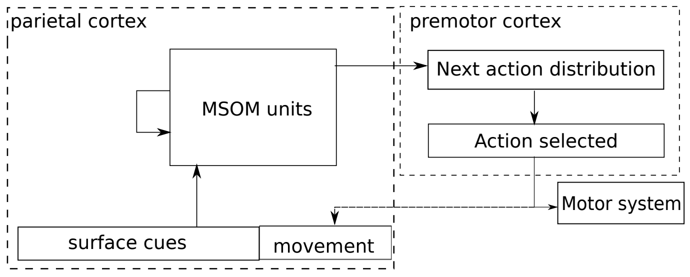


Figure 1: The architecture of the model for representing *3D* objects

```
Data: Constrained action sequences of the object to be explored and represented
Result: A representation of the object explored
Initialization: Randomly initialize the exploration starting position and
orientation of the agent;
while training steps are not finished do
    1. Input: Input sensorimotor information of the navigation agent perceived at
       current time instance, including surface cues and constrained actions;
    2. MSOM units: Activate MSOM units to be responsive to the current input;
    3. Next action distribution: Predict next possible actions allowed by the
       object with a probability distribution;
    4. Action selected: Select the most possible action allowed by the object
       based on the obtained probability distribution and action selection policy and
       then copy to the motor system to execute for updating the agent's position and
       orientation to further explore the object.
end
```

Algorithm 2: Description of the proposed model

3 Model architecture

In the last section, we have presented MSOM and in this section, we configure the MSOM to learn 3D object representations based on haptic information. The architecture of the model is illustrated in Fig. 1, which gets input from perceptual information, such as the texture and temperature information, and the performed action movements. As illustrated in Fig. 1, the model consists of several components and the detailed description of the model is presented in Algorithm 2 and as follows.

3.1 Input

As we can see from Fig. 1, the model gets input from constrained action sequences as well as some supplementary surface cues. Note that the action sequences include translative movements (i.e., move directly forward, move directly left, move directly right and move

Table 1: Agent state, when exploring a $2 \times 2 \times 2$ cube, denoted as [surface number, orientation, x, y] at next time instance $t+1$ after being executed translate-directly movements when its current state at time instance t is (x, y), with N, S, E and W respectively denoting North, South, East and West

Surface number	Orientation	Translate-directly movement			
		Forward	Left	Right	Backward
# 1	N	[1, N, x, $y-1$]	[1, N, $x-1$, y]	[1, N, $x+1$, y]	[1, N, x, $y+1$]
	S	[1, S, x, $y+1$]	[1, S, $x+1$, y]	[1, S, $x-1$, y]	[1, S, x, $y-1$]
	E	[1, E, $x+1$, y]	[1, E, x, $y-1$]	[1, E, x, $y+1$]	[1, E, $x-1$, y]
	W	[1, W, $x-1$, y]	[1, W, x, $y+1$]	[1, W, x, $y-1$]	[1, W, $x+1$, y]
# 2	N	[2, N, x, $y-1$]	[2, N, $x-1$, y]	[2, N, $x+1$, y]	[2, N, x, $y+1$]
	S	[2, S, x, $y+1$]	[2, S, $x+1$, y]	[2, S, $x-1$, y]	[2, S, x, $y-1$]
	E	[2, E, $x+1$, y]	[2, E, x, $y-1$]	[2, E, x, $y+1$]	[2, E, $x-1$, y]
	W	[2, W, $x-1$, y]	[2, W, x, $y+1$]	[2, W, x, $y-1$]	[2, W, $x+1$, y]
# 3	N	[3, N, x, $y-1$]	[3, N, $x-1$, y]	[3, N, $x+1$, y]	[3, N, x, $y+1$]
	S	[3, S, x, $y+1$]	[3, S, $x+1$, y]	[3, S, $x-1$, y]	[3, S, x, $y-1$]
	E	[3, E, $x+1$, y]	[3, E, x, $y-1$]	[3, E, x, $y+1$]	[3, E, $x-1$, y]
	W	[3, W, $x-1$, y]	[3, W, x, $y+1$]	[3, W, x, $y-1$]	[3, W, $x+1$, y]
# 4	N	[4, N, x, $y-1$]	[4, N, $x-1$, y]	[4, N, $x+1$, y]	[4, N, x, $y+1$]
	S	[4, S, x, $y+1$]	[4, S, $x+1$, y]	[4, S, $x-1$, y]	[4, S, x, $y-1$]
	E	[4, E, $x+1$, y]	[4, E, x, $y-1$]	[4, E, x, $y+1$]	[4, E, $x-1$, y]
	W	[4, W, $x-1$, y]	[4, W, x, $y+1$]	[4, W, x, $y-1$]	[4, W, $x+1$, y]
# 5	N	[5, N, x, $y-1$]	[5, N, $x-1$, y]	[5, N, $x+1$, y]	[5, N, x, $y+1$]
	S	[5, S, x, $y+1$]	[5, S, $x+1$, y]	[5, S, $x-1$, y]	[5, S, x, $y-1$]
	E	[5, E, $x+1$, y]	[5, E, x, $y-1$]	[5, E, x, $y+1$]	[5, E, $x-1$, y]
	W	[5, W, $x-1$, y]	[5, W, x, $y+1$]	[5, W, x, $y-1$]	[5, W, $x+1$, y]
# 6	N	[6, N, x, $y-1$]	[6, N, $x-1$, y]	[6, N, $x+1$, y]	[6, N, x, $y+1$]
	S	[6, S, x, $y+1$]	[6, S, $x+1$, y]	[6, S, $x-1$, y]	[6, S, x, $y-1$]
	E	[6, E, $x+1$, y]	[6, E, x, $y-1$]	[6, E, x, $y+1$]	[6, E, $x-1$, y]
	W	[6, W, $x-1$, y]	[6, W, x, $y+1$]	[6, W, x, $y-1$]	[6, W, $x+1$, y]

directly back; and move forward over the edge, move left over the edge, move right over the edge and move back over the edge) and rotational movements (i.e., rotate left and rotate right). After performing a certain movement, the state of the agent (or say, the navigating agent) is changed. For example, when exploring a $2 \times 2 \times 2$ cube, the agent's state is changed differently with different movements, which is illustrated in Table 1 and Table 2. Therefore, starting from a random exploration location and orientation, different constrained action sequences lead the agent to have different states, which implicitly include the topography information of the object. More sensorimotor information, such as cues in the vision and smell, of the object contributes to represent it more accurately, while in this model, we only consider the tactile information of the object. This part is assumed to simulate the process executed in the somatosensory cortex for representing a 3D object.

Table 2: Agent state, when exploring a $2 \times 2 \times 2$ cube, denoted as [surface number, orientation, x, y] at next time instance $t+1$ after being executed translate-over-the-edge movements when its current state at time instance t is $(x,\ y)$, with N, S, E and W respectively denoting North, South, East and West

Surface number	Orientation	Translate-over-the-edge movement			
		Forward	Left	Right	Backward
# 1	N	[4, E, 1, x]	[6, S, 1, y]	[2, N, 1, y]	[5, W, 1, x]
	S	[5, E, 1, x]	[2, S, 1, y]	[6, N, 1, y]	[4, W, 1, x]
	E	[2, E, 1, y]	[4, S, 1, x]	[5, N, 1, x]	[6, W, 1, y]
	W	[6, E, 1, y]	[5, S, 1, x]	[4, N, 1, x]	[2, W, 1, y]
# 2	N	[4, N, x, 2]	[1, N, 2, y]	[3, N, 1, y]	[5, N, x, 1]
	S	[5, S, x, 1]	[3, S, 1, y]	[1, S, 2, y]	[4, S, x, 2]
	E	[3, E, 1, y]	[4, E, x, 2]	[5, E, x, 1]	[1, E, 2, y]
	W	[1, W, 2, y]	[5, W, x, 1]	[4, W, x, 2]	[3, W, 1, y]
# 3	N	[4, W, 2, x]	[2, N, 2, y]	[6, S, 2, y]	[5, E, 2, x]
	S	[5, W, 2, x]	[6, N, 2, y]	[2, S, 2, y]	[4, E, 2, x]
	E	[6, W, 2, y]	[4, N, 2, x]	[5, S, 2, x]	[2, E, 2, y]
	W	[2, W, 2, y]	[5, N, 2, x]	[4, S, 2, x]	[6, E, 2, y]
# 4	N	[6, N, x, 2]	[1, E, y, 1]	[3, E, y, 1]	[2, N, x, 1]
	S	[2, S, x, 1]	[3, E, y, 1]	[1, E, y, 1]	[6, S, x, 2]
	E	[3, S, y, 1]	[6, E, x, 2]	[2, E, x, 1]	[1, N, y, 1]
	W	[1, S, y, 1]	[2, W, x, 1]	[6, W, x, 2]	[3, N, y, 1]
# 5	N	[2, N, x, 2]	[1, E, y, 2]	[3, W, y, 2]	[6, N, x, 1]
	S	[6, S, x, 1]	[3, E, y, 2]	[1, W, y, 2]	[2, S, x, 2]
	E	[3, N, y, 2]	[2, E, x, 2]	[6, E, x, 1]	[1, S, y, 2]
	W	[1, N, y, 2]	[6, W, x, 1]	[2, W, x, 2]	[3, S, y, 2]
# 6	N	[5, N, x, 2]	[1, S, 1, y]	[3, S, 2, y]	[4, N, x, 1]
	S	[4, S, x, 1]	[3, N, 2, y]	[1, N, 1, y]	[5, S, x, 2]
	E	[3, W, 2, y]	[5, E, x, 2]	[4, E, x, 1]	[1, W, 1, y]
	W	[1, E, 1, y]	[4, W, x, 1]	[5, W, x, 2]	[3, E, 2, y]

3.2 MSOM units

Then, the units in MSOM are trained to learn input patterns, which are constrained by the explored object. Regarding an initial exploration position and orientation, the action sequence corresponds to one particular location on the object. Therefore, each unit in the MSOM is driven to be responsive to one particular location in the object, which indicates the geometry property of the object. After training, the learning model is expected to reconstruct or say predict the agent's position as accurately as possible due to the obtained representations of objects. This part is attempting to implement the function fulfilled in the parietal cortex (as well as premotor cortex).

3.3 Next action distribution

For each input pattern, there is an activity in the MSOM. Then, based on the resultant activity, the model attempts to predict the next action possibly allowed by the object based on its learned representation of such an object. Specifically, the MSOM activity is inputted to a network, which is implemented by the multiple layer perception (MLP), and the output of MLP is the probability distribution of all possible actions predicted to be allowed to perform for a certain agent's location and orientation with regard to the special 3D object. At the start of exploration, since there is no information about the object, all actions are chosen with the same probability, which means the next actions are with a uniform distribution. After performing movements, the model comes to obtain the geography information about the object, which predicts the next actions in a probability distribution (normally a non-uniform distribution). The MLP is trained by the back-propagation algorithm.

3.4 Action selected

Based on the obtained actions' probability distribution, the next action to be performed is selected, which is implemented by the Boltzmann selection. Note that the selection can be adjusted by changing the selection decision involved in the Boltzmann selection. For example, if the agent failed one action for the previous attempt, then the probability of this action can be changed to be zero, which means that it cannot be selected for the next action. Plus, to speed up the learning process, the agent is expected to find the boundary of the object as fast as it can and therefore, the probability of the moving forward action is added by a bias number. Furthermore, to avoid the agent exploring in a loop, a 'boredom' routine is introduced, which increases a defined boredom parameter when the agent travels the same location again during a time interval. If the boredom parameter is greater than the previously assigned threshold, the corrected next action's probability distribution is set to distribute in a uniform distribution. The next action selection policy is a decision problem, which needs to balance the accuracy (exploring the same location of the object leads to high accuracy but lower completeness) and completeness (exploring an unknown location gets the lower accuracy but higher completeness) for representing an object. Different action selection policy makes the efficiency of exploration different and therefore a more efficient exploration algorithm is one of the future work. Note that the part about action selection is analogous to the function performed in the premotor cortex. The signal of the selected action is transferred to the motor system to command exploration muscle to execute the corresponding action, which then results in further exploration of the object.

4 Simulation results

In this section, to test the effectiveness for representing 3D objects, the proposed model is tested to learn representations of two typical 3D objects. Meanwhile, to evaluate the performance of the proposed model, three indicators are designed.

4.1 Position reconstruction

Firstly, we investigate the underlying theory that guarantees the accuracy of proposed MSOM for representing a 3D object. To lay a basis for studying the position reconstruction in a 3D object, the reconstruction in a 2D object is presented.

4.1.1 Reconstruction in a 2D object

With regard to a 2D object, we assume it can be regarded as a surface. The surface, then, is partitioned as discrete grid locations. Except for unsuccessful movements (e.g., attempting to cross a blocking wall), every translative movement except for movements of translating over the edge leads to the change of agent's locations in the 2D object. We assume there are $m > 0$ discrete locations in the 2D object and location i is denoted as l_i with $i = 1, \cdots, m$.

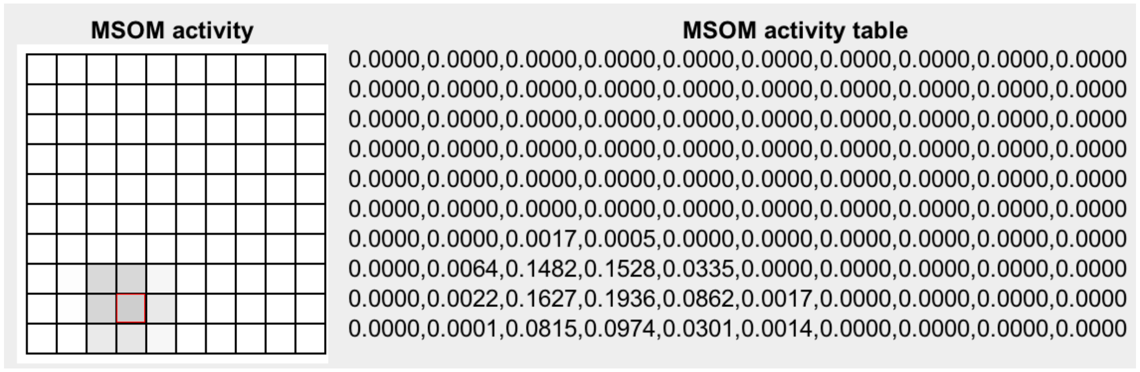
In terms of the applied learning neural network, the so-called MSOM, we assume there are $n > 0$ learning units and learning unit j is denoted as n_j with $j = 1, \cdots, n$. When the agent explores a 2D object, the executed movement combined with encoded surface information at time instance $t - 1$ leads to different activities of learning units in the MSOM. Taking MSOM unit j for example, the activity $a_j(t)$ of neuron's at time instance t is defined as

$$a_j(t) = \exp(-\gamma d_j(t)),$$

where $\gamma > 0$ is a constant and function $d(t)$ stands for the previously defined discriminant function, which is utilized to choose a best matching unit or called winner for a specific input pattern. By normalizing activities of MSOM units to sum to one, the activity of whole MSOM map can be interpreted as a probability distribution. Regarding the unit j, its activity probability $p_j(t)$ for an input pattern at time instance t is formulated as

$$p_j(t) = \frac{a_j(t)}{\sum_{c=1}^{n} a_c(t)}, c = 1, 2, \cdots, n.$$

A MSOM activity distribution at one time instance is shown in Fig. 2. In Fig. 2, the winner is represented with a red frame and corresponding activities of the active MSOM units are illustrated in the MSOM activity table. Note that at the start of training the MSOM, the network might not represent the input pattern accurately and some MSOM units, such as those shown in Fig. 2, may have a similar activity probability. Thus, the whole MSOM activity generally follows a distribution with a high deviation $\delta_1 > 0$. Due to the self-organising process, after a period of time for training, the MSOM network is driven to represent input patterns more accurately and the resultant MSOM activity tends to follow a distribution with a lower deviation $0 \leq \delta_2 < \delta_1$. Theoretically speaking, as time $t \to +\infty$, for an input pattern, there is a winner k in MSOM to make $d_c(t) \to 0$, thereby leading to $a_c(t) \to 1$. Taking into account that other MSOM units are not active, we can have $p_c(t) \to 1$.



MSOM activity table

```
0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000
0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000
0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000
0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000
0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000
0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000
0.0000,0.0000,0.0017,0.0005,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000
0.0000,0.0064,0.1482,0.1528,0.0335,0.0000,0.0000,0.0000,0.0000,0.0000
0.0000,0.0022,0.1627,0.1936,0.0862,0.0017,0.0000,0.0000,0.0000,0.0000
0.0000,0.0001,0.0815,0.0974,0.0301,0.0014,0.0000,0.0000,0.0000,0.0000
```

Figure 2: When exploring a 2D object, a MSOM activity distribution at one time instance.

To learn about how much the object's information represented in MSOM activities, we use the current MSOM activity to reconstruct the agent's location as well as orientation. Apart from the MSOM activity distribution needed for estimating possible agent position in the 2D object, information about the previously visited location combined with the corresponding MSOM winner is also required. Therefore, a named hit map is created, which records traveled position and corresponding MSOM winner. A MSOM hit map at one time instance is illustrated in Fig. 3. For each MSOM neuron j with $j = 1, \cdots, n$, the hit map records the agent actual position as well as orientation in a 2D object being explored when one MSOM neuron acts as the winner at that moment. Note that although for an input pattern, several MSOM neurons might be active, the hit map only takes care of the winner corresponding to current agent position. For each MSOM neuron, there is a counter designed for each position to records how many times the agent visit such a position. In other words, when a neuron becomes a winner, the counter of such a neuron for the actual agent position is forced to increase. Thus, it is easy to obtain the conditional probability $p(\mathrm{l}_i|\mathrm{n}_j)$, which is described as

$$p(\mathrm{l}_i|\mathrm{n}_j) = \frac{\mathrm{T}(i,j)}{\sum_{u=1}^{m} \mathrm{T}(u,j)}, \tag{7}$$

where $\mathrm{T}(i,j)$ denotes the happened times of the agent actual position being i and meanwhile MSOM winner being neuron j; and $\mathrm{T}(u,j)$ is similarly defined.

Considering that the whole MSOM follows a probability distribution, which is shown in (7), we can reconstruct the probable agent position l_i based on total probability theorem, which is described as

$$p(B) = p(B|A_1)p(A_1) + p(B|A_2)p(A_2) + \cdots + p(B|A_k)p(A_k), \tag{8}$$

where B is an arbitrary event; A_1, A_2, $\cdots$, A_k are mutually exclusive events whose probabilities sum to one; and $p(B|A_k)$ is the conditional probability of B assuming A_k happens.

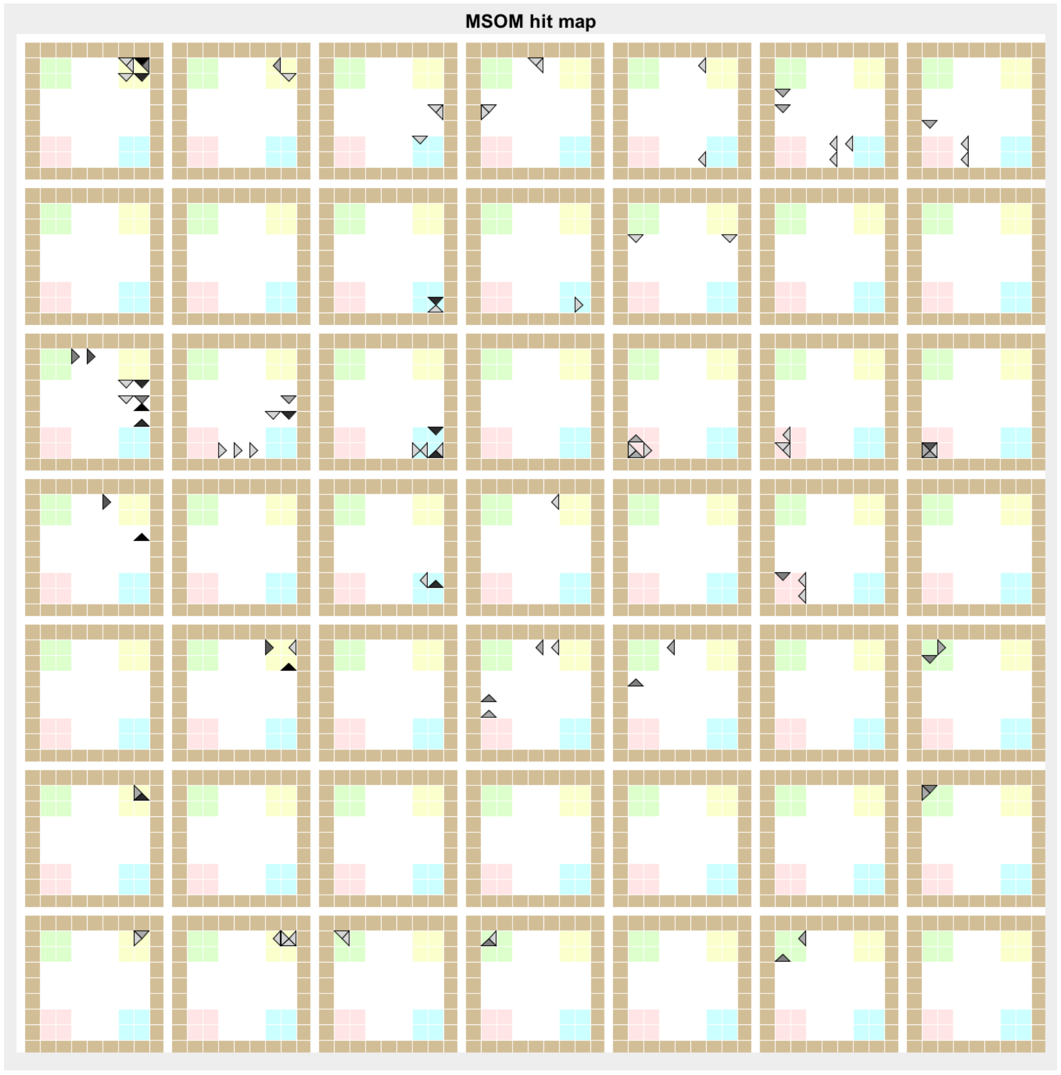


Figure 3: When exploring a 2D object, a MSOM hit map at one time instance.

Therefore, given a probability distribution of MSOM neurons, the probability distribution of reconstructed agent positions is derived as

$$p(\mathrm{l}_i) = \sum_{j=1}^{n} p(\mathrm{l}_i|\mathrm{n}_j)p(\mathrm{n}_j), i = 1, 2, \cdots, m. \quad (9)$$

Apart from such a reconstruction method shown in (9), there are some other approaches to predict possible agent positions. A typical one is to ignore distribution of MSOM neurons

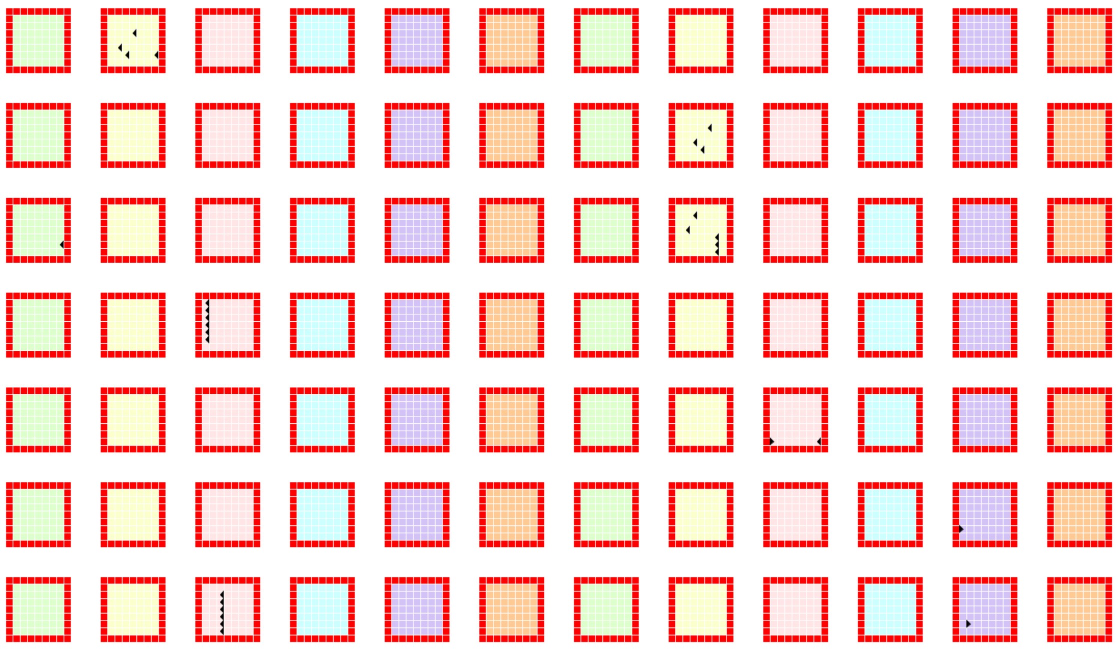


Figure 4: When exploring a 3D object, a part of a MSOM hit map at one time instance.

and just focus on the winner. Based on the winner as well as the obtained conditional probability shown in (7), the distribution of probable agent positions can be calculated. Compared with the first approach, this approach is easier and simpler to implement. Another approach to reconstruct is only paying attention to the MSOM winner and meanwhile only focusing on the most probable agent position based on previously gleaned conditional probability of agent positions. That is to say, it assumes the most probable agent position to be the reconstructed agent position. Note that the in spite of the fact that the last approach is simplest, it might reconstruct with an appreciable error, especially when the training is not enough.

4.1.2 Reconstruction in a 3D object

Regarding a 3D object, different grids in surfaces of the object means distinct positions. Therefore, reconstruction not only considers different positions in one surface, but also a position in different surfaces. Differing from the hit map for a 2D object, a hit map for exploring a 3D object is then modified to incorporate surface information. A part of a hit map when the agent explores a 3D object is shown in Fig. 4 and different colors stand for different surfaces. Therefore, for each MSOM neuron, a counter is utilized to record hit times for each position *in each surface*. For an input pattern, the MSOM similarly follows a probability distribution. For simplicity, we assume the number of surfaces in a 3D object is $q \geq 2$ and every surface s_v with $v = 1, 2, \cdots, q$ includes m positions. Therefore, in total, there are mq positions in the 3D object. Via keeping record of each position in each surface, the conditional probability $p(\mathrm{s}_v, \mathrm{l}_i|\mathrm{n}_j)$ is formulated as

$$p(\mathrm{s}_v, \mathrm{l}_i|\mathrm{n}_j) = \frac{\mathrm{T}(v, i, j)}{\sum_{v=1}^{q} \sum_{i=1}^{m} \mathrm{T}(v, i, j)}, \tag{10}$$

where $p(\mathrm{s}_v, \mathrm{l}_i|\mathrm{n}_j)$ denotes the probability of the agent in position l_i of the surface s_v, given the winner being n_j. In addition, $T(\cdot, \cdot, \cdot)$ is defined as before.

Based on the activity distribution of MSOM units and the recorded conditional probability $p(\mathrm{s}_v, \mathrm{l}_i|\mathrm{n}_j)$, we can evidently obtain the probability distribution of reconstructed positions in a 3D object. The probability distribution for reconstructed positions is expressed as

$$p(\mathrm{s}_v, \mathrm{l}_i) = \sum_{j=1}^{n} p(\mathrm{s}_v, \mathrm{l}_i|\mathrm{n}_j)p(\mathrm{n}_j), i = 1, 2, \cdots, m. \tag{11}$$

Similar with estimating probable positions in a 2D object, we can generalize the other two reconstruction approaches to obtain probability distribution of possible positions in surfaces in the explored 3D object. If the surface and location of the maximal value in the reconstructed probability obtained in (11) are the same with the actual agent's surface and location, the parameter indicating the times of this event's occurrence is forced to increase, which suggests the reconstruction accuracy and the representation ability of the model.

4.2 Distance

Secondly, we introduce the distance between reconstructed positions and the actual agent's position to indicate the model's effectiveness. We first investigate the model exploring a 2D object and then study the situation when the model tries to learn 3D objects.

4.2.1 Manhattan distance in a 2D object

When the agent moves in four orthogonal directions (i.e., North, South, East and West), we adopt the Manhattan distance $M(\cdot)$ to evaluate the accuracy of the proposed model, which is defined as

$$M(x_1, y_1, x_2, y_2) = |x1 - x2| + |y1 - y2|, \tag{12}$$

where (x_1, y_1) and (x_2, y_2) are two points based on a Cartesian coordinate system in a 2D space and $|\cdot|$ denotes its absolute value. The reconstructed error $e \geq 0$ is calculated as

$$e = \sum_{i=1}^{m} M(\mathrm{p}_{\mathrm{act}}, \mathrm{l}_i)p(\mathrm{l}_i), \tag{13}$$

where $\mathrm{p}_{\mathrm{act}}$ and l_i denotes agent actual position and estimate position, respectively.

4.2.2 Geodesic distance in a 3D object

Considering that all movements of the agent when exploring a 3D object are along the surface instead of passing through the object, Geodesic distance has an advantage over Manhattan distance for this case. To find the Geodesic path between two points, which may exist in different surfaces, a well-studied graph-based search algorithm Dijkstra algorithm is implemented. Therefore, the model's reconstruction error e can be estimated based on the Geodesic distances $G(\cdot,\cdot,\cdot,\cdot)$ between the actual agent position and the reconstructed agent position, which is calculated as

$$e = \sum_{v=1}^{q} \sum_{i=1}^{m} G(\mathrm{s}_{\mathrm{act}}, \mathrm{p}_{\mathrm{act}}, \mathrm{s}_v, \mathrm{l}_i) p(\mathrm{s}_v, \mathrm{l}_i), \tag{14}$$

where $\mathrm{s}_{\mathrm{act}}$ and $\mathrm{p}_{\mathrm{act}}$ denotes agent actual surface and position, respectively; and s_v and l_i denotes agent estimate surface and position, respectively. Therefore, this indicator defined in (14) with a smaller value means the model with a better representation ability.

4.3 Uniqueness

To avoid the model exploring a 3D object in a loop, a called uniqueness is defined as $U = \tau/\epsilon$, where U is the uniqueness rate, τ is the number of unique positions and ϵ is the number of exploration steps within a sliding window.

4.4 Results

The simulation results indicate that the model is effective on representing 3D objects and it is more efficient and accurate for representing an asymmetrical 3D object than a symmetrical object, which can refer to the CogSci 2018 paper.

5 Conclusion

Benefiting from that the allowed action sequences are constrained by the geometry of a 3D object, this paper investigates a neural network model for representing 3D objects through tactile exploration. The simulation results demonstrate the model's effectiveness for learning 3D objects' representations. Future work might investigate the multi-dimensional MSOM for learning representations of more complex 3D objects, such as a cube as well as a cube in a desk.

6 Acknowledgement

The authors would like to thank Martin Takac for his work of SOM-based navigation model with Alistair Knott and this help given to this paper.

References

Simon Haykin. *Neural Networks: A Comprehensive Foundation, 2nd ed.* Englewood Cliffs, NJ: Prentice-Hall, 1999.

Samuel Kaski. Data exploration using self-organizing maps. In *Acta polytechnica scandinavica: Mathematics, computing and management in engineering series no. 82.* Citeseer, 1997.

Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982.

Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

Teuvo Kohonen. The self-organizing map. *Neurocomputing*, 21(1):1–6, 1998.

Fabio Leoni, Massimo Guerrini, Cecilia Laschi, Davide Taddeucci, Paolo Dario, and Antonina Starita. Implementing robotic grasping tasks using a biological approach. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 3, pages 2274–2280. IEEE, 1998.

Edvard I Moser, Emilio Kropff, and May-Britt Moser. Place cells, grid cells, and the brain's spatial representation system. *Annual review of neuroscience*, 31:69–89, 2008.

Helge Ritter, Thomas Martinetz, Klaus Schulten, Daniel Barsky, Marcus Tesch, and Ronald Kates. *Neural computation and self-organizing maps: an introduction.* Addison-Wesley Reading, MA, 1992.

Marc Strickert and Barbara Hammer. Merge som for temporal data. *Neurocomputing*, 64: 39–71, 2005.

Hujun Yin. Visom-a novel method for multivariate data projection and structure visualization. *IEEE Transactions on Neural Networks*, 13(1):237–243, 2002.

Hujun Yin. On multidimensional scaling and the embedding of self-organising maps. *Neural Networks*, 21(2):160–169, 2008a.

Hujun Yin. The self-organizing maps: background, theories, extensions and applications. In *Computational intelligence: A compendium*, pages 715–762. Springer, 2008b.