Proceedings of International Joint Conference on Neural Networks, San Jose, California, USA, July 31 – August 5, 2011

Self-Organizing Neural Population Coding for Improving Robotic Visuomotor Coordination

Tao Zhou[1], Piotr Dudek[2], Bertram E. Shi[1]

Abstract—**We present an extension of Kohonen's Self Organizing Map (SOM) algorithm called the Self Organizing Neural Population Coding (SONPC) algorithm. The algorithm adapts online the neural population encoding of sensory and motor coordinates of a robot according to the underlying data distribution. By allocating more neurons towards area of sensory or motor space which are more frequently visited, this representation improves the accuracy of a robot system on a visually guided reaching task. We also suggest a Mean Reflection method to solve the notorious border effect problem encountered with SOMs for the special case where the latent space and the data space dimensions are the same.**

I. Introduction

Human infants are thought to develop accurate visuomotor coordination ability through a circular reaction process, where they move their limbs randomly and observe the movement with their eyes [1]. Through this process, they gradually learn the relationship between their motor output and their vision input. During the past twenty years, it has been demonstrated by many researchers that this circular reaction process can also enable a robot system to develop visuomotor coordination [1-4]. The hope is that by learning mappings by a circular reaction process, we can avoid the need for complex system calibration and explicit computation of the geometrical transformation required in classical robotics, as well as enable adaptation in real time visuomotor coordination.

In learning visuomotor coordination, one key problem we need to address is how to represent the information from both vision and motor domains. Georgopoulos and colleagues have shown that neurons in motor cortex are tuned to the directions of limb movements [5]. The final direction of the limb movement may be decoded by taking a vectorial sum of the activities across the neural population [6]. Inspired by this, we adopt a similar neural population coding method in our system.

In the literature modeling visuomotor coordination, researchers have often used a fixed neural population encoding where neurons have tuning curves with identical widths and respond maximally at a set of visual or motor coordinates that are uniformly distributed across the stimulus space [1, 4, 7, 8]. This presupposes that all regions of the stimulus space must be represented with equal resolution, which may not be true. In addition, this requires advance knowledge of the range of the sensory or motor space that must be represented.

To remove these assumptions, we propose an algorithm called the self organizing neural population coding (SONPC) algorithm, which adjusts online the selectivity and tuning widths of the neurons encoding visual and motor coordinates, according to the underlying distribution of the data it encounters. Because the adaptation is online, even if the underlying data distribution is not stationary, the SONPC algorithm can adapt the coding by the neural population to the data distribution as it changes. The algorithm is based on the framework of Kohonen's Self Organizing Map (SOM) [9]. The SOM is a data driven, biologically inspired learning algorithm. The key feature of SOM is that it enforces topology constraints on the neural population through the concept of a latent space. A similar topographic mapping is thought to exist in the mammalian cortex[10].

Being based upon the SOM, our algorithm also suffers from a border effect problem. The distribution of neural selectivities is biased away from the borders of the data space towards the center. This has the disadvantage that if we decode activity by taking a vectorial sum of the neural activity, where each neuron's activity is weighted by its optimal stimulus, the decoded output displays a similar bias towards the center of the data space. To deal with this, we propose a Mean Reflection method, which can be used to reduce this problem when the dimension of latent space lattice is the same as the data space.

In addition to an efficient coding of the visual and motor coordinates, the other essential part of visuomotor coordination development is a method that learns the relationship between these two coordinate systems. The learning algorithm we adopt here is similar to our previous work [8] where we use the supervised Hebbian learning algorithm to learn the cross correlation between vision and motor domain.

The paper is organized as follows. Section 2 gives a brief description of the robotic system we run our visuomotor experiments on. Section 3 introduces the SONPC algorithm in detail. Section 4 explains the supervised Hebbian learning algorithm, which learns the relationship between the neural population codes of visual and motor space that develop

[1] Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, HONG KONG, eebert@ee.ust.hk

[2] School of Electrical and Electronic Engineering, The University of Manchester, Manchester, UK.

according to the SONPC algorithm, so that the robotic system can learn to point to where it is looking. Section 5 describes how we solve the border effect problem using the Mean Reflection method. Section 6 presents our experimental results. Section 7 concludes the paper with a discussion of our results.

II. System Description

Our robotic system consists of a 7DoF robot arm and an active vision system. The vision system consists of a camera mounted on a pan-tilt unit, so that together can mimic the motion of a human head. We are currently controlling only two joints of the robot arm: the shoulder and elbow joint, so that the endpoint of the arm moves in a plane. These two joint angles are our motor coordinates.

During the training stage, the end effector of robot arm holds an LED, which is much more salient than the environment and easy for the camera to detect. The robot arm moves its two arm joints randomly inside a preset range. While the arm is moving in front of the camera, a computer uses the camera image to determine the position of the LED, and controls the pan-tilt unit so that the LED is always located at the center of the camera image. We use the positions of the pan and tilt joints of the pan-tilt head to represent the vision signal. As the arm and pan-tilt unit move, we use the SONPC algorithm described below to learn neural population codes for the vision and motor coordinates. We also use the supervised Hebbian learning algorithm described below to learn the relationship between the neural population codings of vision and motor coordinates.

During the testing period, the LED is removed from the end effector. When the camera tracks the location of the LED, we use the pan and tilt angles of the pan-tilt head to generate activity in the neuron population encoding visual space. This activity then generates activity in the neural population encoding the motor space via the weights learned by supervised Hebbian learning. Finally, this motor space neural activity is decoded by a vectorial sum to generate a position control signal that causes the robot arm to point to the target that the camera is looking at.

III. Self Organizing Neural Population Coding

In our visuomotor coordination development system, we represent the current value of the visual or motor coordinates as the distributed activity across a population of neurons. Each neuron in the population has a tuning curve describing how its response changes as the current coordinate changes. The tuning curve is largest if the current coordinates are equal to a particular location in visual or motor space, and decreases as the distance between the current coordinates and the associated location increases. In designing a population of neurons to encode the current coordinates, we must not only choose an associated locations of the neurons so that they cover the range of values encountered, but we must also determine the shape of the tuning curve. In this section, we start with a discussion of the classical SOM model, which addresses the first of these problems. We then describe the SONPC model, which extends the SOM to address both problems.

A. The SOM algorithm

The classical SOM algorithm assumes that neurons are arranged topologically in a latent space, which is typically one or two dimensional. If we index each neuron by an integer i, each neuron is associated with a location $\mathbf{r}_i$ in the latent space. Each neuron has an associated weight vector $\mathbf{m}_i$, which represents a point in the data space. The SOM algorithm allocates neurons so that the distribution of weight vectors covers the distribution of a set of data in a way that preserves the underlying topology. Neurons which are close in the data space represent points that are close in the data space. Note that the dimensionality of the data space and the latent space may differ. The SOM is commonly used for dimensionality reduction, where the dimension of the data space is larger than that of the latent space.

The SOM algorithm updates the weight vectors iteratively. At each iteration t, the SOM algorithm considers one point in the data set, $\mathbf{x}$. It finds the neuron whose weight vector is closest in Euclidean distance. If we denote the index of this "winning" neuron by w, then

$$w = \arg\min_i \{\|\mathbf{x} - \mathbf{m}_i(t)\|\} \tag{1}$$

The weight vectors are then updated so that the weight vectors of the winning neuron and the neurons closest to it in the latent space move closer to the input data point $\mathbf{x}$. For each neuron i,

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + \alpha(t) \cdot h_{wi}(t)[\mathbf{x}(t) - \mathbf{m}_i(t)] \tag{2}$$

where $\alpha(t)$ is a learning rate which decreases over time, and $h_{wi}(t)$ is a neighborhood function between neurons w and i. The neighborhood function decreases as the distance between neurons in the latent space increases. Here we use a Gaussian neighborhood function:

$$h_{wi}(t) = \exp\left(\frac{-\|\mathbf{r}_i - \mathbf{r}_w\|^2}{2\sigma(t)^2}\right) \tag{3}$$

The standard deviation of the Gaussian typically remains constant or decreases over time.

B. The SONPC algorithm

The SONPC algorithm is similar to the SOM algorithm in that assumes an underlying topology between neurons, and allocates neurons to different regions of the data space. It extends the SOM by also modeling and adjusting the tuning curves of the individual neurons. The SOM does not explicitly model neural tuning responses, but rather simply assigns data points to neuron based upon the smallest Euclidean distance. The same distance measure is applied to all neurons.

Similarly to the SOM, the SONPC algorithm indexes each neuron by an integer i, and associates each neuron with a location $\mathbf{r}_i$ in the latent space. Each neuron's response to an input $\mathbf{x}$ is given by a Gaussian function in the data space, normalized so that the total activity in the population sums to one. If we denote the activity of neuron i in response to stimulus $\mathbf{x}$ by $a(\mathbf{x},i)$, then

$$a(\mathbf{x},i)=\frac{\mathcal{N}(\mathbf{x}\,|\,\mathbf{m}_i,\Sigma_i)}{\sum_{k=1}^{N}\mathcal{N}(\mathbf{x}\,|\,\mathbf{m}_k,\Sigma_k)} \tag{4}$$

where N is the total number of neurons in the population and

$$\mathcal{N}(\mathbf{x}\,|\,\mathbf{m}_i,\Sigma_i)=\frac{1}{(2\pi)^{\frac{d}{2}}\left|\Sigma_i\right|^{\frac{1}{2}}}\exp(-\frac{(\mathbf{x}-\mathbf{m}_i)^T\Sigma_i^{-1}(\mathbf{x}-\mathbf{m}_i)}{2}) \tag{5}$$

and we assume the data to have dimension d. The mean $\mathbf{m}_i$ is analogous to the weight vector of the SOM, and determines the preferred stimulus of neuron i. The covariance matrix Σ_i controls the shape of the tuning curve.

Similarly with the SOM, the SONPC uses competitive learning rule with neighborhood constraints to update the parameters $\mathbf{m}_i$ and Σ_i iteratively. At each iteration, a winning neuron is chosen by the neuron whose activity is the largest

$$w=\arg\max_i(a(\mathbf{x},i)) \tag{6}$$

The mean vectors $\mathbf{m}_i$ are updated according to equation (2). The covariance matrices are updated in an analogous way

$$\begin{aligned}\Sigma_i(t+1)=\Sigma_i(t)+\alpha(t)\cdot h_{wi}(t)\cdot\\ \{[\mathbf{x}(t)-\mathbf{m}_i(t)]\cdot[\mathbf{x}(t)-\mathbf{m}_i(t)]^T-\Sigma_i(t)\}\end{aligned} \tag{7}$$

In our implementation, we schedule the learning rate and the standard deviation of the neighborhood function so that they decrease during training. We use similar sigmoidal curves for both. For example, the learning rate decreases according to

$$\alpha(t)=\alpha_i-\frac{\alpha_i}{(1+\exp(-0.0002\cdot(t-L/320)))}+\alpha_f \tag{8}$$

where L denotes the pre-assigned total training length. The final value of the standard deviation of the latent space neighborhood function has a significant influence on the final coding scheme. If final sigma is large, the receptive fields of the neurons will have large overlap and will be less spread out. If sigma is small, neurons will have little or no overlap and will be more spread out. In our implementation, the final value of sigma is set to 0.5 which provides us the balance between continuity and coverage of the neural population coding scheme.

C. *Relationship with previous work*

Previous work in learning visuomotor coordination that has used population codes to represent visual and motor coordinates used neurons with Gaussian tuning curves that are homogenous (circularly symmetric) and homoscedastic (identical variance across all neurons) [4, 7]. The Gaussian functions in the SONPC algorithm are more general, being both heterogeneous and heteroscedastic. As demonstrated by our experimental results, this allows more flexibility, which is especially useful when coding non-uniformly distributed data, as we find in our visuomotor coordination task. Because the coordinate transformation is nonlinear, even if the data is uniformly distributed in one space (e.g. the motor space), it will be non-uniformly distributed in the other space (e.g. the visual space). There is also an underlying assumption of homogeneity and homoscedasticity in the SOM algorithm, since it uses the same Euclidean distance measure when comparing the distances between the input data and the weight vectors of each neuron.

Ritter et al. have proposed to apply the SOM for learning visuo-motor coordination [2]. However, while Ritter et al. model visual and motor space simultaneously with a single SOM, we use two neural populations to represent the two visual and motor spaces separately.

The SONPC algorithm is similar to an online version of EM algorithm for updating Gaussian mixture models [11-13]. Consider a Gaussian mixture model for the data given by

$$p(\mathbf{x})=\frac{1}{N}\sum_{k=1}^{N}\mathcal{N}(\mathbf{x}\,|\,\mathbf{m}_k,\Sigma_k) \tag{9}$$

where each mixture has equal prior probability. The activity $a(\mathbf{x},i)$ is equal to the responsibility or the conditional probability that mixture i generated the observation $\mathbf{x}$. However, the EM algorithm assumes no topological relationship between the mixtures. In the EM algorithm, the influence of each data point on the next set of parameter estimates for mixture i depends upon the responsibility. In the SONPC algorithm, the influence of each data point on the next set of parameter estimates for neuron i depends upon the distance between i and the winning neuron in the latent space. The EM algorithm also updates the prior probabilities of the mixtures. In SONPC, we constrain the prior probabilities to be equal.

Past work has also incorporated topology into the Gaussian mixture model by combining it with the SOM algorithm. Yin and Allison also use a similar competitive rule for choosing a winning neuron and update only the parameters of neurons close to the winning neuron [14]. However, they weight the updates by the responsibility, rather than the neighborhood function, which we expect will reduce the topological constraints enforced by the latent space. Van Hulle uses a more complex formulation based upon a bias-variance decomposition of a weighted error function [15], but considers only the case of homogenous Gaussians in the mixture.

IV. Development of Visuomotor Map

For the visuomotor map learning, we adopt the a method similar to that reported in [8], except that we use supervised Hebbian Learning rule to learn the cross correlation between the vision and motor spaces.

Given a set of training data, which consists of pairs of visual and motor coordinates, we first update a neural population coding for each space separately using the SONPC algorithm described above.

For each data pair, we then generate neural population activity corresponding to the vision and motor coordinates using equation (4) with the current estimated parameters. Similar to work by Salinas and Abbot [4], we learn the relationship between the neural population activity encoding the vision coordinates and the neural population activity encoding the arm coordinates using a supervised Hebbian rule. We assume that neuron i in the arm population is connected to neuron j in the vision population by a weight w_{ij} , and that the weights are updated according to the correlation between their activity

$$\Delta w_{ij} = \eta \cdot a_m(\mathbf{x}_m, i) \cdot a_v(\mathbf{x}_v, j) \tag{10}$$

where the subscripts m and v indicate the motor and vision spaces respectively.

Since Hebbian learning without competition leads to unbounded weight evolution, after each update we normalize the weight so that the weights connecting to one particular motor neuron sum to one.

$$w_{ij} = \frac{w_{ij}}{\sum_{k=1}^{N} w_{ik}} \tag{11}$$

During the testing stage, we remove the LED from the end effector of the arm and allow the camera to track the LED. Given the resulting visual coordinates, we generate visual neuron population activity using equation (4) and generate motor neuron population activity using the weights learned by supervised Hebbian learning.

$$a_m(i) = \sum_{j=1}^{N} w_{ij} a_v(\mathbf{x}_v, j) \tag{12}$$

We translate the motor population activity into a commanded arm position $\mathbf{y}_m$ sent to the robot arm by taking a vectorial sum where each neurons activity is weighted by its preferred set of arm coordinates

$$\mathbf{y}_m = \frac{1}{S} \sum_{i=1}^{N} \mathbf{m}_{m,i} a_m(i) \tag{13}$$

where

$$S = \sum_{i=1}^{N} a_m(i) \tag{14}$$

and $\mathbf{m}_{m,i}$ are the set of Gaussian means learned for the motor population encoding by SONPC. We evaluate performance by determining how close the commanded arm position is to the actual set of arm positions required for the end effector to reach the LED.

V. Reducing the Border Effect

The border effect is a well-known problem in applications using the SOM. The positions of neurons in the data space do not reach the border of the data range, but rather are shifted towards the center of the data range. Due to its similarity to the SOM, the SONPC algorithm suffers from the same border effect problem. This increases the errors between the commanded and desired arm positions when they are at the edges of the motor space.

In the literature, there are several methods which try to solve the border effect of SOM including spherical SOM [16] and the heuristic weighting strategy proposed by Kohonen [17]. However, the method proposed in [16] cannot solve the border effect problem when the data has abrupt borders. When using the method proposed in [17], it is hard to determine whether a data point is inside the neuron range or outside the neuron range.

Solving the border effect problem in general is difficult, since it is difficult to determine which data points are near the border. However, if the dimension of the latent space is the same as the dimension of data space, it seems reasonable to assume that the borders of the neural array in the latent space (which are known), will coincide with the borders of the data space (which are a priori unknown). In this case, we propose to reduce the border effect by creating "virtual data points" outside the border for each input data point. The "virtual data points" are obtained by reflecting the mean of the winning neuron around the data space positions of the border neurons in the latent space.

This "Mean Reflection" method is based upon two assumptions. First, the data is distributed uniformly within the winning range of the border neuron up to the border. This is reasonable if the data distribution changes continuously, as it does in our application. Second, the border neuron should be is located right at the border, which is abrupt.

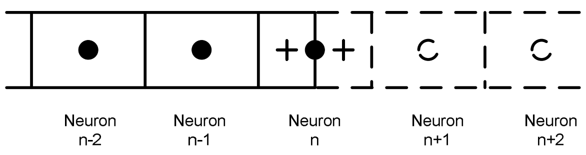

Fig. 1 One-dimensional mean reflection method

Fig. 1 illustrates the Mean Reflection method for the one dimensional case. The solid dots indicate the mean locations of the neurons in the data space. The vertical solid lines indicate the range of inputs for which each neuron wins. The figure assumes that the mean location of border neuron n falls on the border of the data. Assume that for an input data point $\mathbf{x}$, a non-border neuron indexed by w wins, we create two "virtual data points" by reflecting the mean of the winning neuron with respect to the mean location the border neurons 1 and n. These are shown as dashed circles. We also create two "virtual winning neurons" outside the array in the latent space claim these data points. For example, if neuron w wins, we create virtual neurons indexed 2-w and 2n-w to win the data points reflected outside the left and right sides of the array. Combining the parameter update equations for these three data points (one real and two virtual), we obtain the following update equation for the mean of each neuron i

$$\begin{aligned}\Delta \mathbf{m}_i &= \alpha \cdot h_{w,i} \cdot (\mathbf{x} - \mathbf{m}_i) + \alpha_f \cdot h_{2n-w,i} \cdot (\mathbf{m}_{rn} - \mathbf{m}_i) \\ &+ \alpha_f \cdot h_{2-w,i} \cdot (\mathbf{m}_{r1} - \mathbf{m}_i)\end{aligned} \tag{15}$$

where $\mathbf{m}_{rn}$ is the reflection of the mean of winning neuron around border neuron n and $\mathbf{m}_{r1}$ is the reflection of the mean around border neuron 1. If a border neuron wins, then instead of reflecting its mean, we reflect a point halfway between the mean and the border of the data range it covers assuming that it lies on the border, as illustrated in Fig. 1 by the solid cross. The reflected mean with respect to border neuron b is given by

$$\mathbf{m}_{rb}(w) = 2 \cdot \mathbf{m}_b - \mathbf{m}(w) \qquad (16)$$

where

$$\mathbf{m}(w) = \begin{cases} 0.75\mathbf{m}_1 + 0.25\mathbf{m}_2 & \text{if } w = 1 \\ \mathbf{m}_w & \text{if } 1 < w < n \\ 0.25\mathbf{m}_{n-1} + 0.75\mathbf{m}_n & \text{if } w = n \end{cases} \qquad (17)$$

The Mean Reflection method can be easily generalized higher dimensions. Fig. 2 illustrates the Mean Reflection rule for 2D. The key difference is that we now create four virtual data points and neurons, rather than two, since we must reflect around the top and bottom borders, as well as the left and right borders.

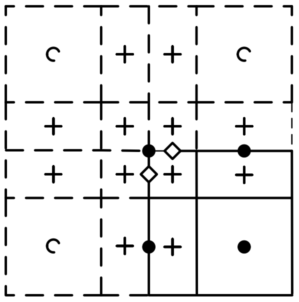
Fig. 2 Two-dimensional mean reflection method

VI. Experimental Results and Comparison

A. Experimental Results

In order to verify that the Self Organizing Neural Population Coding can result in a better representation, which improves performance on the visuomotor mapping task, we implemented the algorithm on the robot arm system described previously. We use 10x10 arrays of neurons to represent the visual and motor domains. The supervised Hebbian learning algorithm for learning visuomotor mapping algorithm remains the same. Only the neural population coding scheme changes. We perform 40,000 training iterations in each experiment. Each training iteration includes parameter updates for the visual and motor neural population codes using SONPC and a weight update using the supervised Hebbian learning algorithm. To test the results, we choose 1600 arm joint angles pairs according to a uniform distribution and measure the mean square error (MSE) between the mapping result calculated from the corresponding pan-tilt position and the ground truth.

We first compare the neural population codes learned using SONPC with and without the Mean Reflection algorithm. Figs. 3 and 4 show the results for the motor space. Figs. 5 and 6 show the results for the visual space. In these figures, the blue dots indicate the training data points, which were obtained by uniformly sampling the arm joint space and measuring the resulting pan-tilt angles in the vision space. The solid cross indicates the mean location of each neuron and the ellipses indicate the covariance matrices of each neuron. Different colors are used to enable the parameters for different neurons to be distinguished. Comparing the figures, we see that that the Mean Reflection algorithm is effective in pushing the neuron mean locations out to, but not beyond, the border of the data range. In addition, we see that the covariance ellipses for the motor data, which is uniformly distributed are all approximately the same. The covariance ellipses show more variation in size for the vision data. They are smaller in the lower right hand corner where the data is more densely concentrated, but larger in the upper left hand corner, where the data is sparser.

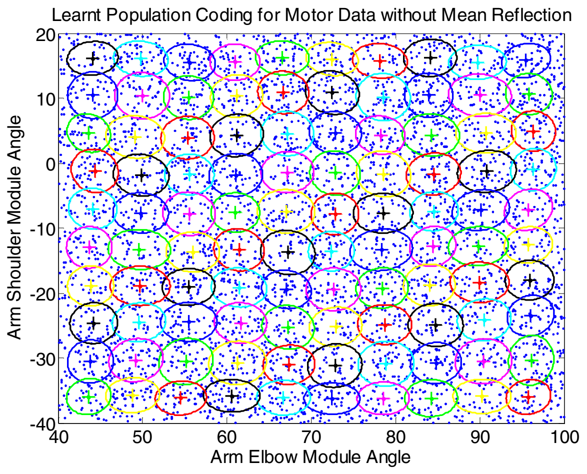

Fig. 3 Population coding for motor data without Mean Reflection.

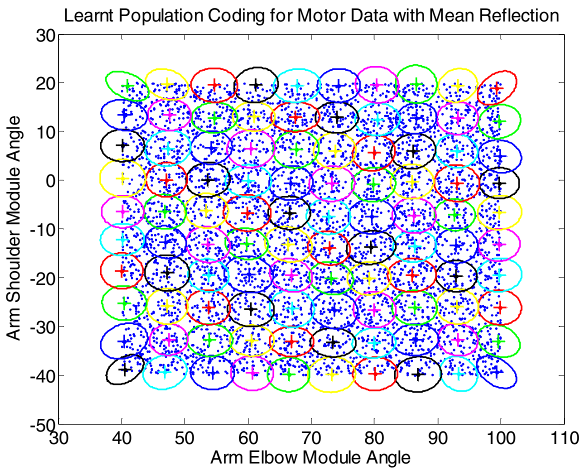

Fig. 4 Population coding for motor data with Mean Reflection.

Figs. 7 and 8 show the visuomotor mapping result for population codes learned by SONPC with and without Mean Reflection. The blue dots are the ground truth arm joint angles. The red dots are commanded arm joint angles calculated from the vision data. With Mean Reflection, the performance near the border is greatly improved.

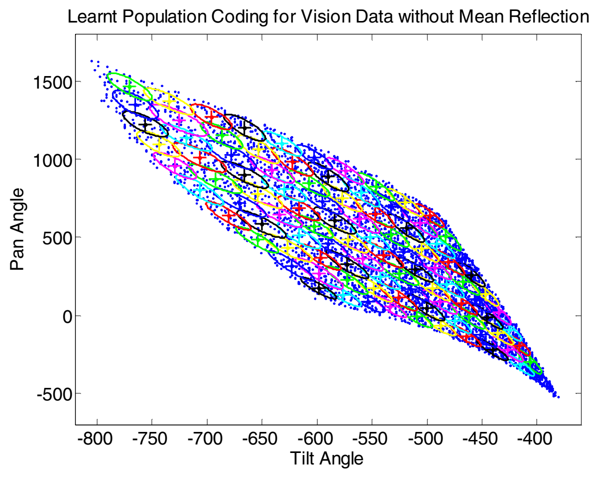

Fig. 5 Population coding for vision data without Mean Reflection

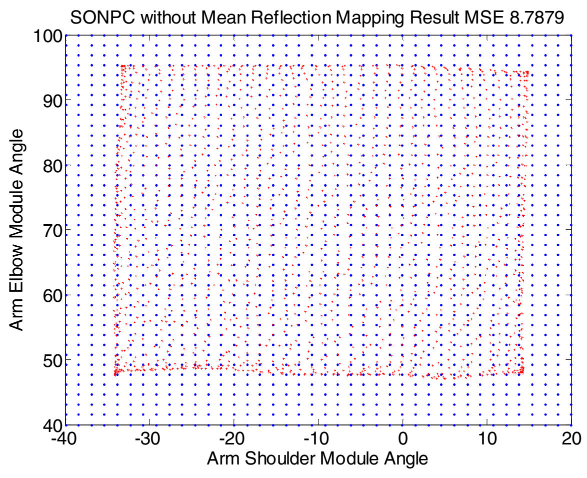

Fig. 7 Visuomotor mapping for SONPC without Mean Reflection

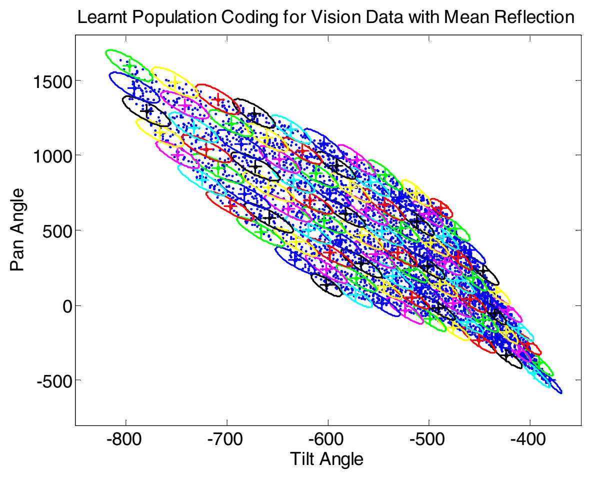

Fig. 6 Population coding for vision data with Mean Reflection

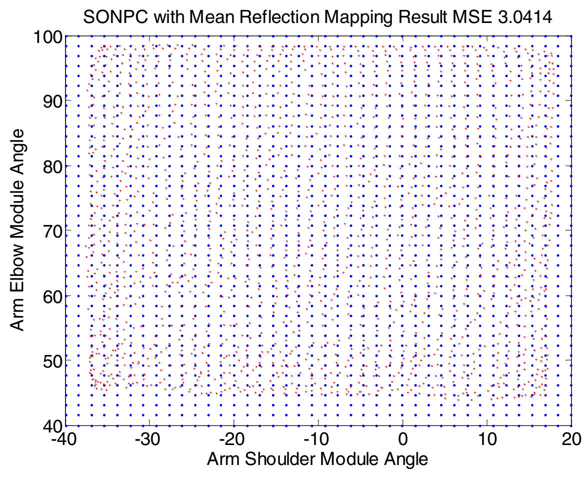

Fig. 8 Visuomotor mapping for SONPC with Mean Reflection

B. *Comparison with Alternative Approaches*

We first compare the performance of the neural population codes learned by the SONPC algorithm with the fixed neural population coding used in our previous work [8]. In that work, the neural population activities were calculated using equation (4), where the mean locations were fixed to equally sample the rectangular range containing the data. Data was rescaled so that the edges of this rectangular range correspond to values of 0 and 1. While this is probably effective for representing the motor space, where the data points are distributed evenly over the entire range, it is not effective for the vision data, as it allocates neurons to locations in visual space that are never used. The covariances were equal to the identity matrix scaled by the variance σ^2.

For the fixed neural population coding algorithm, the performance of the mapping depends upon the value of σ. Fig. 9 shows the MSE between the commanded and ground truth arm joint angles as a function of the distance from the center of the array.

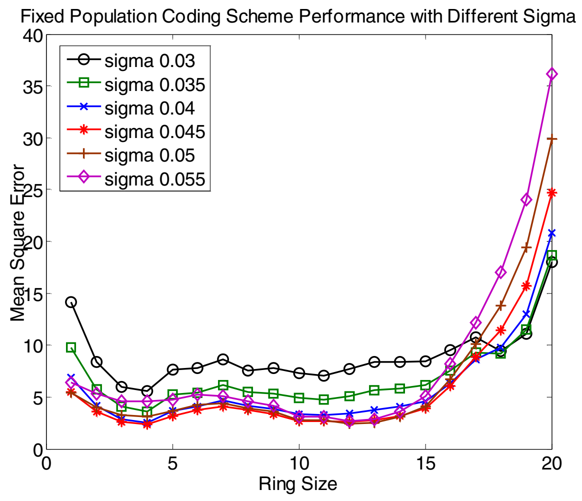

Fig. 9 MSE of the fixed neural population coding

As expected, the MSE near the border is much larger than the MSE in the center region due to the border effect. Fig. 9 shows that as the standard deviation σ increases from 0.03 to 0.055, the error near the border increases while the error in the center area decreases. For our later comparisons, we chose σ equal to 0.045, as a tradeoff. Further increases in σ do not significantly reduce the MSE in the center of the arm joint range, but lead to degraded performance at the boundaries.

Fig. 10 compares the visuomotor coordination performance between the fixed neural population coding and SONPC with and without Mean Reflection. In the central region, the performance of SONPC with and without Mean Reflection is comparable. Both show better performance than the fixed neural population coding scheme. This is not surprising, since the fixed neuron population wastes neurons by allocating them to regions of visual space which are not used. Without Mean Reflection, the performance near the boundaries degrades to the point that the MSE exceeds that of the fixed coding, where we manually located border neurons at the data border. The MSE of the neural coding learned by SONPC with Mean Reflection is smaller than that of fixed coding in all regions.

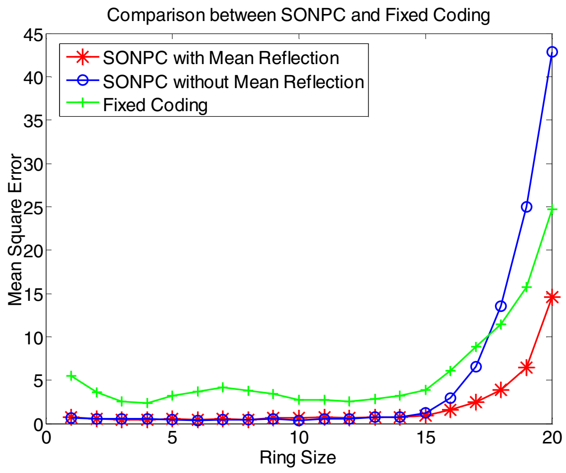

Fig. 10 MSE of the fixed neural population coding and the neural population coding learned by SONPC with and without Mean Reflection.

As described above, the classical SOM algorithm does not model neural responses, only the distribution of preferred stimuli in a population. Thus, it is impossible to directly compare performance of the SOM and the SONPC on this task. However, we noted above that the SONPC relaxes the homogeneity and the homoscedasticity assumptions implicit in the SOM algorithm. We examined here the importance of relaxing these constraints by constraining the SONPC algorithm so covariance matrix of all neurons is equal to the identity matrix scaled by a global variance σ^2, which is updated according to the equation

$$\sigma^2(t+1) = \sigma^2(t) + \alpha(t) \cdot \{\tfrac{1}{2}[\mathbf{x}(t) - \mathbf{m}_w(t)]^T \cdot [\mathbf{x}(t) - \mathbf{m}_w(t)] - \sigma^2(t)\} \quad (18)$$

Fig. 11 shows the MSE of the SONPC algorithm with and without homogeneity and homoscedasticity constraints. In neither case was Mean Reflection used. We see that the MSE of the SONPC without these constraints is much smaller than with the constraints.

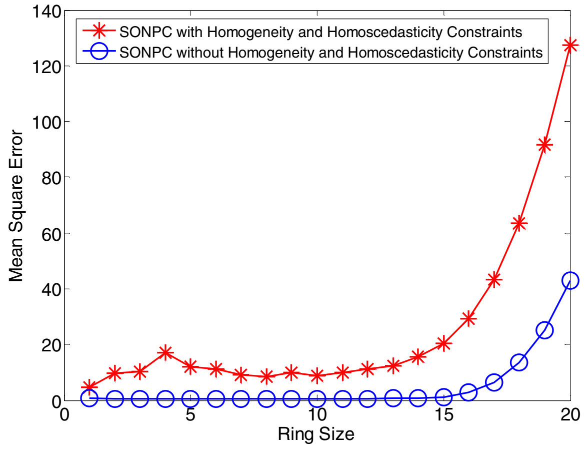

Fig. 11 MSE of the SONPC algorithm with and without homogeity and homoscedasticity constraints.

Finally, we note that increasing the number of neurons improves the accuracy of the visuomotor mapping, at the expense of additional computation time. When we increase the neural populations representing the visual and motor spaces from 10x10 to 20x20, the total MSE over the entire range drops from 3.04 to1.34. However, the training time when running in MATLAB on a PC increases from 25 minutes to 100 minutes.

VII. Conclusion

The aim of this work was to develop an algorithm for learning a neural population code that can effectively represent an data set, and to demonstrate that this algorithm improves performance on a visuomotor coordination behavior development task. The Self Organizing Neural Population Coding algorithm we describe is an extension of the SOM algorithm, and is related to the Gaussian Mixture Model. Given a data set, it can learn online a topography preserving neural population coding scheme to represent that data set. Our experimental results demonstrate that this adaptive algorithm finds a better population coding of visual and motor data, resulting in improved performance on a visually guided pointing task.

The complexity of SONPC algorithm is larger than the classical SOM algorithm, since the SONPC needs to estimate the covariance parameters in addition to the mean parameters estimated by the SOM. Given *N* neurons and a *D* dimensional data space, the number of parameters that must be estimated by the SOM is *ND*, versus *N*(*D*+*D*(*D*+1)/2). However, our experimental results indicate that this additional flexibility results in large performance gains. Importantly, the complexity is still linear in the number of neurons *N*. An additional advantage of the SONPC algorithm is that it also models population responses, rather than simply the distribution of preferred stimuli in the neural population

We also proposed a Mean Reflection algorithm, which results in a better representation of the data near the borders of the data set. Our experimental results show that incorporating this Mean Reflection algorithm improves task-related performance near the borders.

REFERENCES

[1] M. Kuperstien, "Neural Model of Adaptive Hand-Eye Coordination for Single Postures," *Science,* vol. 239, pp. 1308-1311, Mar 11 1988.

[2] H. J. Ritter*, et al.*, "Topology-Conserving Maps for Learning Visuo-Motor-Coordination," *Neural Networks,* vol. 2, pp. 159-168, 1989.

[3] D. Bullock*, et al.*, "A Self-Organizing Neural Model of Motor Equivalent Reaching and Tool Use by a Multijoint Arm," *Journal of Cognitive Neuroscience,* vol. 5, pp. 408-435, Fal 1993.

[4] D. Caligiore*, et al.*, "Toward an integrated biomimetic model of reaching," in *Development and Learning, 2007. ICDL 2007. IEEE 6th International Conference on*, 2007, pp. 241-246.

[5] A. Georgopoulos*, et al.*, "On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex," *J. Neurosci.,* vol. 2, pp. 1527-1537, November 1, 1982 1982.

[6] A. Georgopoulos*, et al.*, "Primate motor cortex and free arm movements to visual targets in three- dimensional space. II. Coding of the direction of movement by a neuronal population," *J. Neurosci.,* vol. 8, pp. 2928-2937, August 1, 1988 1988.

[7] E. Salinas and L. F. Abbott, "Transfer of Coded Information from Sensory to Motor Networks," *Journal of Neuroscience,* vol. 15, pp. 6461-6474, Oct 1995.

[8] W. Yiwen*, et al.*, "Hebbian learning of visually directed reaching by a robot arm," in *Biomedical Circuits and Systems Conference, 2009. BioCAS 2009. IEEE*, 2009, pp. 205-208.

[9] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE,* vol. 78, pp. 1464-1480, 1990.

[10] G. D. SCHOTT, "Penfield's homunculus: a note on cerebral cartography," *Joumal of Neurology, Neurosurgery, and Psychiatry,* vol. 56, pp. 329-333, 1993.

[11] A. P. Dempster*, et al.*, "Maximum Likelihood from Incomplete Data Via Em Algorithm," *Journal of the Royal Statistical Society Series B-Methodological,* vol. 39, pp. 1-38, 1977.

[12] M.-a. Sato and S. Ishii, "On-line EM Algorithm for the Normalized Gaussian Network," *Neural Computation,* vol. 12, pp. 407-432, 2000.

[13] B. Awwad Shiekh Hasan and J. Gan, "Sequential EM for Unsupervised Adaptive Gaussian Mixture Model Based Classifier," in *Machine Learning and Data Mining in Pattern Recognition*. vol. 5632, P. Perner, Ed., ed: Springer Berlin / Heidelberg, 2009, pp. 96-106.

[14] H. Yin and N. M. Allinson, "Bayesian learning for self-organising maps," *Electronics Letters,* vol. 33, pp. 304-305, Feb 13 1997.

[15] M. M. Van Hulle, "Maximum likelihood topographic map formation," *Neural Computation,* vol. 17, pp. 503-513, Mar 2005.

[16] Ritter, "Self-Organizing Maps on non-euclidean Spaces," ed: Elsevier, 1999.

[17] T. Kohonen, *Self-Organizing Maps*, 3rd ed. Verlag, Berlin, Heidelberg, New York: Springer, 2001.