Trajectory Analysis on Spherical Self-Organizing Maps with Application to Gesture Recognition

Artur Oliva Gonsales and Matthew Kyan

Ryerson University
350 Victoria Street, Toronto, Ontario, Canada, L5J 1C9
aolivago@ryerson.ca, mkyan@ee.ryerson.ca

Abstract. We propose a new approach to gesture recognition using the properties of Spherical Self-Organizing Map (SSOM). Unbounded mapping of data onto a SSOM creates not only a powerful tool for visualization but also for modeling spatiotemporal information of gesture data. Once mapped onto a SSOM the gesture data is treated as a series of postures. A set of postures describing a specific path on the SSOM for a gesture is used as a trajectory. Although some trajectories may share the same postures, the path consisting of posture transitions will always be unique. Different variations of posture transitions occurring within a gesture trajectory are used to classify new unknown gestures. Experimental results on datasets involving full body and hand gestures show the effectiveness of our proposed method.

Keywords: spherical SOM, gesture recognition, trajectories.

1 Introduction

Nowadays, many applications require the use of powerful visualization tools that can assist data analysts in evaluating their data. This allows deriving meaningful inferences, therefore gaining deeper understanding about the physical phenomenon characterizing their data. One example of such visualization tool is immersive virtual reality which was created due to recent advances in research and can provide a rich visualization and interactive modeling and analysis tool [1]. Such advanced, interactive, and task-driven display and analysis tools utilize a full range of human sensorimotor capabilities and provide an insight on large volumes of experimentally acquired data. The data modeling approach discussed in this paper based on trajectory analysis presents a different view into the use of Self-Organizing Maps, which gives a viable mechanism to generate a spatio-temporal representation of data from multi-dimensional data. Several gesture data sets are used to demonstrate the effectiveness of the proposed methodology in building spatio-temporal trajectories.

In this paper, we create gesture trajectories with the help of Self-Organizing Maps (SOM) [2]. In particular, we use the Spherical SOM structure (SSOM) [3], because of its ability to map multi-dimensional data without boundaries. SOM is an unsupervised clustering approach proposed by Kohonen [2] that clusters data from high-dimensional

P.A. Estévez et al. (Eds.): *Advances in Self-Organizing Maps*, AISC 198, pp. 125–134.
DOI: 10.1007/978-3-642-35230-0_13

space into low-dimensional space, while still preserving its topology. For sequences of input data whose features are expected to temporarily change in a smooth way, it is anticipated that a topology preserved mapping can allow for the formation of a smooth trajectory on the map. Regular SOMs map the data points onto a flat 2D lattice during training by updating the weights of the nodes in the lattice. However, this setup has a restricted boundary, normally pushing the data to be mapped along its boundaries. Also, it can be argued that the opposite sides of the boundary are not topologically close in the SOM space. A more optimal choice for SOM structure consists of sphere [3], which minimizes topological discontinuity. This SOM structure is created by sub-dividing an Icosahedron, providing the SOM structure with a symmetric node distribution depicted in Fig. 1. One of the advantages of the SSOM is that regions of density found in the feature space will map equally spaced and well separated locations on the sphere due to the wrap-around effect of the lattice. In this work, we leverage this property to build trajectory based features to distinguish between human full body motion gestures.

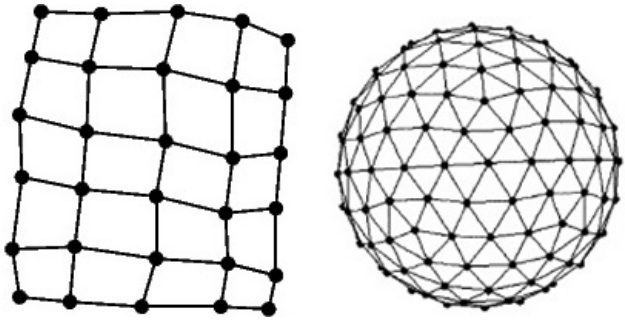

Fig. 1. Open vs. Closed structure of 2D- and Spherical SOM

The rest of the paper is organized as follows: Section 2 discusses the related works in gesture recognition using SOMs. Section 3 gives details about the experimental setup and the datasets. Section 4 provides experimental results. Finally, in Section 5 we conclude the paper with a conclusion.

2 Related Work

The use of SOFM in the area of gesture recognition has been relatively recent. Some methods which will be discussed shortly in this section have used SOFM in various ways to divide the sample data into clusters of phases and are further processed with the help of other tools and techniques. In [4], Oshit and Matsunaga use a SOM to first process the gesture data and then apply a Support Vector Machine (SVM) to partition the feature space into regions belonging to separate classes. Their approach is interesting because they divide each gesture into short phases and then apply a pattern recognition technique for multi-dimensional data to recognize each phase. By using Dynamic Programming (DP), authors match the trajectory from the input

signals and the sample trajectory from a gesture. By *trajectory* we refer to the temporal path that the data maps into on the SOM lattice based on a set of consecutive Best Matching Units (BMUs). The disadvantage of this approach is that the trajectories must be projected onto low-dimensional feature space. Furthermore, a valid threshold must be specified to measure the similarity of two trajectories. One of the limitations of standard SOM that is tried to be overcome in this work is to eliminate the restricted boundary of a 2D lattice. This is due to the boundaries being open and nodes on the boundaries not having the same number of neighbors as the inner nodes.

A. Shimada and R. Taniguchi in [5] use a Sparse Code of Hierarchical SOM (HSOM). A Hierarchical SOM [6] is a two layer SOM network, where the lower layer has a connection with an input layer. In this case, the second layer receives an input vector from the first layer directly. The method proposed by Shimada uses the property of HSOM to first learn postures (minimum unit of a gesture) in the first layer, and the learn short gestures consisting of some time-series postures in the second layer. Authors argue that the time length of a human gesture is not always the same even if same gestures are compared. They highlight that the key issue in their method is to absorb the time variant appropriately in order to make clusters which include the same gesture class.

The interesting part of the approach by Shimada and Taniguchi for gesture recognition is how they tackle the problem of time invariance or length invariance, speaking in terms of trajectories. The use of multi-layer SOM allows them to obtain a more general gesture path on the SOM lattice without worrying about its length.

Another method to gesture recognition is suggested in *Video-Based Gesture Recognition Using Self-Organizing Feature Maps* [7], [8]. This work introduces a probabilistic recognition scheme for hand gestures, where SOFMs are used to model spatiotemporal information extracted from images. It uses a combination of SOFM and Markov models for gesture classification. The classification scheme consists of tracking the transformation of gesture representations from a series of coordinate movements.

3 Experimental Setup and Datasets

All of the methods discussed previously and most seen in literature involve the use of two-dimensional SOFMs. The attempt in this paper is to show how the properties of a SOFM can be used for smoothly varying data such as body or hand gestures to create a good recognition system by implementing a 3D version of the SOFM. Following this section is an overview of the experimental set up and the datasets used.

3.1 Dataset

A dataset involving full body gestures was used in this work. This dataset was collected using sensor equipment in the Microsoft Kinect camera. A virtual version of the game Charades was used to collect full body gesture data. Nineteen gestures were selected randomly out of a classic commercial version of Charades. Figure 2

alphabetically lists the 19 different gestures that were used in the database. It is easy to see how these gestures are very open to interpretation. Of the 19 gestures (classes), 50 full samples of each gesture were sampled. The Kinect primarily samples user 'gesture' information from the IR depth camera. The data coming from the camera is oriented relative to its distance from the Kinect. This becomes problematic when searching for the solution to universal truths in gestures. Normalization was used to that convert all depth and position data into vectors relative to a single joint presumed most neutral. In this case the torso was considered as the neutral position of the body. Figure 2 shows the skeleton model with the points (body parts) used in the dataset. The result includes positive and negative x, y, and z-axis values. The feature vector consists of 60 features (three displacement vectors -x,y,z multiplied by 20 body points). The average temporal length of each gesture in the database is 200-300 frames.

Air Guitar	Clapping	Laughing
Archery	Crying	Monkey
Baseball	Driving	Skip Rope
Boxing	Elephant	Sleeping
Celebration	Football	Swimming
Chicken	Heart Attack	Titanic
		Zombie

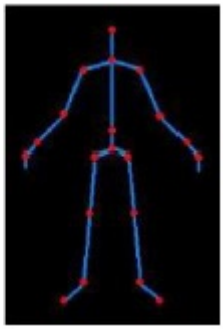

Fig. 2. Microsoft Kinect Full body gesture data and skeleton showing the gestures (left), body parts (right) being tracked

3.2 Experimental Setup

In all the experiments performed in this work the setup was identical. A Spherical SOFM was used with specific settings and size which will be discussed shortly. All the experiments were performed on standalone PC with Windows 7, 4GB of RAM and Intel Core i7 CPU (2.67GHz). MATLAB R2011b environment was used for all the experiments and the visualization part. On average it took several minutes to train the network with one gesture depending on the feature vector size of a specific gesture.

3.3 SSOM Training

The training phase of the SSOM is identical to the conventional 2D SOM [2]. Let, the input space of N nodes be represented by $\chi = \{x_i\}_{i=1}^{N}$. Let the SSOM be represented by M nodes ($M << N$). Each node in the SSOM lattice has a corresponding weight vector w. All these weight vectors together represent the SSOM space $\Psi = \{\psi_i\}_{i=1}^{M}$. Each node also has a *neighborhood* associated with it. A neighborhood is a set of

nodes consisted of the node itself and its neighbors. Let, the neighbourhood set for node *i* be represented by Θ_i^r. Here, *r* represents the neighborhood spread, *r = 1,…,R. R* is the maximum neighborhood radius, which is set to a value such that it covers half of the spherical space [3].

The input nodes are randomly introduced to the SSOM during training. For each voxel, a *Best Matching Unit (BMU)* among all the nodes is selected. BMU is the node which is closest to the input voxel according to some similarity measure. Euclidean distance is usually used as distance measure. The update step then takes place, where the weight vector of the BMU and its neighboring nodes (Θ_{BMU}^r) are updated in a way so that they are pulled closer to the weight of the input voxel. After training, the SSOM weight vectors are arranged in such a way that represents the underlying distribution of the input data (the node features in this case). The training algorithm is described below:

- **Initialization:** The weight vectors of the SSOM nodes are initialized first. Random values can be used for initialization, but as pointed out by Kohonen et al. in [2], random initialization will take more time to converge. A count vector $0 = \{o_i\}_{i=1}^{M}$ [3] is used to keep track of the hits to each node. This vector is initialized to zero. This is used in the BMU selection step (explained below) to prevent cluster underutilization.

  Training: For each input *x* (a feature vector containing the coordinates of the 20 body points), do the following:

— **BMU Selection:** Calculate the Euclidean distance of the node feature vector x with all the nodes as follows:

$$e_i = (o_i + 1)\left||x - w_i|\right|, i = 1, \dots M \tag{1}$$

BMU is the node for which this distance is the smallest.

— **Weight Update:** Update the weights for the BMU and its neighboring nodes (defined by Θ_{BMU}^r) as follows:

$$w = w + b(t) * h(s,r) * \left||x - w|\right|, \tag{2}$$

$$c_w = c_w + h(s,r), \tag{3}$$

Where $w \in \Theta_{BMU}^r$, $b(t) = \alpha e^{-\frac{t}{T}}$ and $h(s,r) = e^{-\frac{r^2}{s*R}}$.

The functions *b(t)* and *h(s,r)* control the rate of learning and neighborhood effect, respectively. *b(t)* decreases in value as the epoch number *t= 1,2,…T* increases. It also depend on the learning rate α. *h(s,r)* depends on the neighborhood size parameter *s,* which is user controlled. *h(s,r)* is a Gaussian function. The further a neighboring node is from a BMU, the less its weight will be affected.

- Repeat the training steps for a pre-defined number of epochs (20)

The main control parameters in SSOM training are the learning rate α, the number of epochs T and the neighborhood size parameter *s*. In the following experiments, the setting used were T=20, and *s*=4.

4 Experimental Results

Figure 3 shows some sample trajectories that were obtained during the mapping process. These gesture trajectories are a representation of the BMUs hit sequence that gesture mapped onto the SSOM. The data that is being used in the trajectory mapping comes from the training portion of the datasets. All the BMUs are in 3D space although they appear as 2D images. The lattice of the Spherical SOFM was removed on purpose so that the trajectories could be seen more clearly.

Each gesture class was displayed at a different angle from the rest in order to show the data path more clearly. From Figure 3 it is evident that the trajectories for each gesture class trace a similar if not identical path on the spherical lattice of the SOFM. It is also clear that each gesture leaves a path which is unique if comparing to other gestures. It is important to note that many gestures may share common BMUs since they may contain similar postures that trace a specific c gesture. We count every BMU hit of a gesture as a posture belonging to a given gesture class. A collection of these postures form a gesture. In the next sections the methods for gesture recognition will be described.

4.1 Gesture Recognition: Using All Postures (Method 1)

The gesture recognition and classification initially starts with a simple approach. As mentioned earlier, all the BMUs that trace a trajectory for a specific gesture are considered as postures. All the BMUs from each gesture class are used as a collection of points or postures for the purpose of classification of new unknown data. This is done in the following manner:

1. All the BMUs falling into trajectories belonging to a specific gesture are recorded into a set G_i as follow:

$$G_i = \{Tr_{(i,1)}, \dots, Tr_{(i,m-1)}, Tr_{(i,m)}\}, \tag{4}$$

Where $Tr_{(i,j)}\ for\ j = 1,2, \dots m$ is a trajectory forming a gesture G_i and i is the gesture index which represents a gesture class, also

$$Tr_{(i,j)} = \{P_k, \dots, P_{n-1}, P_n\}, \tag{5}$$

Where P_k is the k^{th}node in the SSOM lattice (i.e. posture) and n is the number of nodes or postures in the trajectory $Tr_{(i,j)}$.

2. Feature vectors (consisting of the coordinates of the body parts) of an unknown gesture coming from the testing portion of the dataset are then compared against the weights of the SOFM and the BMUs from the new trajectory of an unknown gesture are collected into a new set *Tp*.
3. A frequency posture counter K_i assists in determining the class of the unknown gesture, where i represents the index of a known gesture. The counter K_i for a gesture i is incremented if a posture from an unknown gesture belongs to a gesture

being compared against. This way, Tp is compared against all the G_i in the database. So, if $K_i \geq K_1, K_2, \ldots, K_n$, where K_n is a counter belonging to a gesture with index *n,* then K_i is chosen as the winning counter the unknown gesture is classified as gesture G_i.

4.2 Gesture Recognition: Weighted Aggregation of All Postures (Method 2)

The approach taken for gesture recognition in the last section only takes into consideration a set of postures as a primary data to classify an unknown gesture. This set does not take into account the frequency with which a specific posture is encountered in a gesture's trajectories. For this reason a frequency factor is introduced in this approach. All postures are aggregated into a set, but at the same time each posture is associated with a weight. The more a posture appears in a gesture path while training the network, the more weight it has towards that gesture. The reason behind the weight factor is that the path that a trajectory representing a specific gesture maps on the lattice of the SOFM tends to activate the same neurons (postures), giving it a higher probability to appear again if the same gesture is traced.

4.3 Gesture Recognition: Using Posture Transitions (Method 3)

Previous methods tested, treated each BMU as a posture and no dynamic information was used. Dynamic information in this case refers to the posture transitions that occur during the tracing of a gesture onto the SOFM. The main argument here is that trajectories belonging to the same gesture should follow not only the same path on the Spherical lattice but also have similar transitions in terms of postures. For example, when a person performs a "Driving" gesture, he or she will follow the same posture transition as he or she moves the hands in a 3D space. A *similar* posture transition is defined as having two identical consecutive BMU hits from node *A* to node *B,* in two different trajectories. For this specific reason the classification of an unknown gesture is evaluated based on similar posture transitions during the formation of its trajectory.

4.4 Gesture Recognition: Weighted Aggregation of All Posture Transitions (Method 4)

The last approach in this paper uses a weighted aggregation of all posture transitions. Similarly as in the method with weighted aggregation of all postures a weight is introduced. This approach not only takes in consideration the posture transitions happening in a gesture trajectory but also the frequency with which these transitions occur.

4.5 Discussion

Tables 2-5 show the gesture recognition rate for all the approaches implemented in this work. From Fig. 3 it is evident that a good data separation is obtained, which can

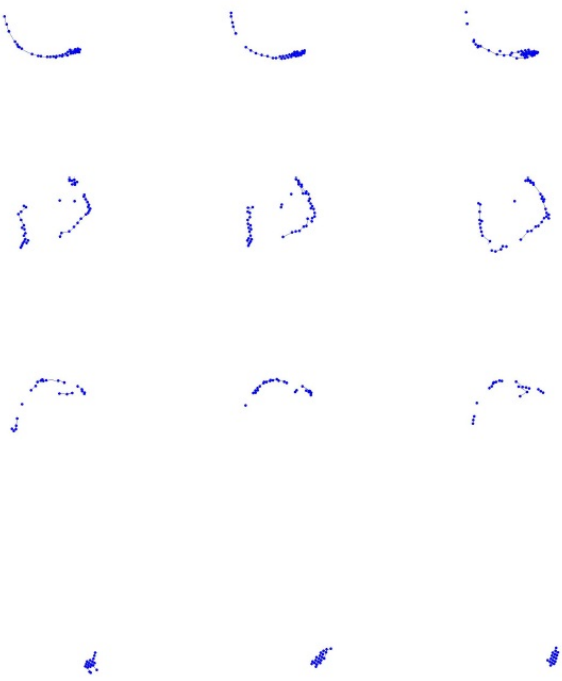

Fig. 3. Sample Microsoft Kinect Gesture trajectories. (Three samples for four gestures). From first to last row: Air Guitar, Archery, Baseball, Boxing.

Table 1. Recognition rate: using all postures (Method 1).

Gesture	Rate %	Gesture	Rate %	Gesture	Rate %
Air Guitar	80(20)	Clapping	52(13)	Laughing	80(20)
Archery	72(18)	Crying	56(14)	Monkey	84(21)
Baseball	76(19)	Driving	68(17)	Skip Rope	64(16)
Boxing	88(22)	Elephant	64(16)	Sleeping	72(18)
Celebration	84(21)	Football	68(17)	Swimming	96(24)
Chicken	48(12)	Heart Attack	80(20)	Titanic	52(13)
Zombie	60(15)				

Table 2. Recognition rate. Weighted aggregation of all postures (Method 2).

Gesture	Rate %	Gesture	Rate %	Gesture	Rate %
Air Guitar	100(25)	Clapping	92(23)	Laughing	100(25)
Archery	100(25)	Crying	88(22)	Monkey	92(23)
Baseball	96(24)	Driving	100(25)	Skip Rope	80(20)
Boxing	88(22)	Elephant	96(24)	Sleeping	28(7)
Celebration	100(25)	Football	100(25)	Swimming	100(25)
Chicken	48(12)	Heart Attack	84(21)	Titanic	96(24)
Zombie	100(25)				

Table 3. Recognition rate. Using posture transitions (Method 3).

Gesture	Rate %	Gesture	Rate %	Gesture	Rate %
Air Guitar	100(25)	Clapping	100(25)	Laughing	100(25)
Archery	100(25)	Crying	80(20)	Monkey	100(25)
Baseball	88(22)	Driving	100(25)	Skip Rope	80(20)
Boxing	60(15)	Elephant	44(11)	Sleeping	100(25)
Celebration	96(24)	Football	100(25)	Swimming	100(25)
Chicken	76(19)	Heart Attack	100(25)	Titanic	100(25)
Zombie	100(25)				

Table 4. Recognition rate. Weighted aggregation of all posture transitions (Method 4).

Gesture	Rate %	Gesture	Rate %	Gesture	Rate %
Air Guitar	100(25)	Clapping	100(25)	Laughing	100(25)
Archery	92(23)	Crying	100(25)	Monkey	100(25)
Baseball	100(25)	Driving	100(25)	Skip Rope	92(23)
Boxing	100(25)	Elephant	100(25)	Sleeping	100(25)
Celebration	100(25)	Football	100(25)	Swimming	96(24)
Chicken	100(25)	Heart Attack	100(25)	Titanic	88(22)
Zombie	92(23)				

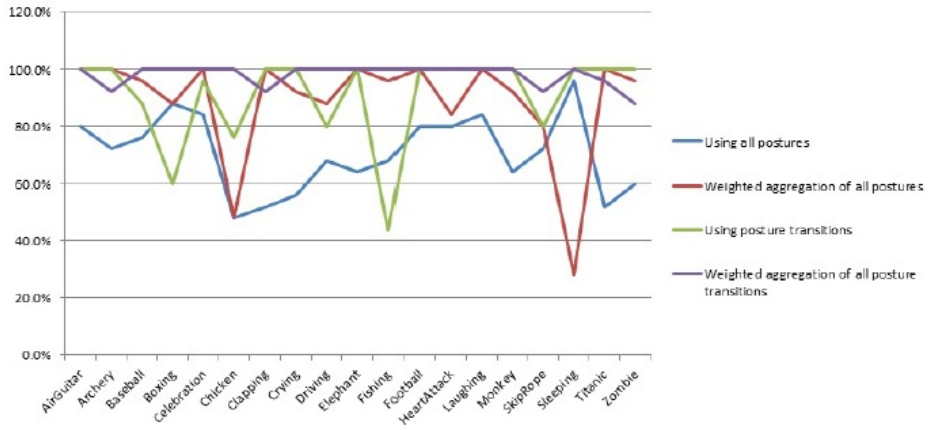

Fig. 4. Gesture recognition comparison chart

be seen from the trajectories: different gestures have different trajectories. As discussed earlier, such separation is reached because of the wrap-around effect of the SSOM lattice. It is also clear that a higher classification rate is obtained when using the dynamic structure of the trajectories such posture transitions. The reason why, for instance, Method 4 works better than others is because it uses dynamic information (posture transitions) that other methods do not. By introducing a weight factor, we increase the chances of classifying unknown gesture correctly, because gestures trajectories tend to have the same transitions from one posture to the next. Fig. 4 depicts

all the results in a chart, clearly showing the advantages of using posture transitions with implementation of frequency weights.

5 Conclusions

In this paper, we have proposed the use of SSOM for trajectory analysis with application to gesture recognition. We implemented four different approaches to classify new gesture data, clearly showing the advantages of using the dynamic structure of the gesture data. An overall result of 97.9% of correct classification was obtained by using the *weighted aggregation of all posture transition* method. As a future work, we would like to seek methods for describing full trajectories with a descriptor. The challenge that lies in creating such a descriptor is the length of the sample, which is always variable. Creating trajectories from samples of variable length (i.e. gesture data) also makes the trajectories to be different in its lengths. A low frequency descriptor such as Fourier Descriptor may be used to describe a trajectory, but first the length factor has to be addressed. The advantage of using a spherical SOM is that it offers a constrained spherical coordinate system on which such a descriptor can be based.

References

1. Furht, B. (ed.): Immersive virtual reality. Encyclopedia of Multimedia. Springer (2006)
2. Pratt, W.K. (ed.): Digital Image Processing. John Wiley and Sons, New York (2007)
3. Sangole, A., Leontitsis, A.: Spherical self-organizing feature map: An introductory review. International Journal of Bifurcation and Chaos 16, 3195–3206 (2006)
4. Oshita, M., Matsunaga, T.: Automatic Learning of Gesture Recognition Model Using SOM and SVM. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Chung, R., Hammoud, R., Hussain, M., Kar-Han, T., Crawfis, R., Thalmann, D., Kao, D., Avila, L. (eds.) ISVC 2010, Part I. LNCS, vol. 6453, pp. 751–759. Springer, Heidelberg (2010)
5. Shimada, A., Taniguchi, R.I.: Gesture recognition using sparse code of hierarchical SOM. In: International Conference of Pattern Recognition (ICPR), pp. 1–4 (2008)
6. Lampinen, J., Oja, E.: Clustering properties of hierarchical self-organizing maps. J. Mathematical Imaging and Vision 2(2-3), 261–272 (1992)
7. Caridakis, G., Pateritsas, C., Drosopoulos, A., Stafylopatis, A., Kollias, S.D.: Probabilistic Video-Based Gesture Recognition Using Self-organizing Feature Maps. In: de Sá, J.M., Alexandre, L.A., Duch, W., Mandic, D.P. (eds.) ICANN 2007. LNCS, vol. 4669, pp. 261–270. Springer, Heidelberg (2007)
8. Caridakis, G., Karpouzis, K., Pateritsas, C., Drosopoulos, I., Stafylopatis, A., Kollias, D.: Hand trajectory based gesture recognition using self-organizing feature maps and markov models. In: Internation Conference on Multimedia and Expo (ICME), pp. 1105–1108 (2008)