A Riemannian Self-Organizing Map

Dongjun Yu[1,2], Edwin R. Hancock[2], and William A.P. Smith[2]

[1] School of Computer Science, Nanjing University of Science and Technology, Nanjing 210094, China
njyudj@mail.njust.edu.cn
[2] Department of Computer Science, The University of York, York YO10 5DD, UK
{erh,wsmith}@cs.york.ac.uk

Abstract. We generalize the classic self-organizing map (SOM) in flat Euclidean space (linear manifold) onto a Riemannian manifold. Both sequential and batch learning algorithms for the generalized SOM are presented. Compared with the classical SOM, the most novel feature of the generalized SOM is that it can learn the intrinsic topological neighborhood structure of the underlying Riemannian manifold that fits to the input data. We here compared the performance of the generalized SOM and the classical SOM using real 3-Dimensional facial surface normals data. Experimental results show that the generalized SOM outperforms the classical SOM when the data lie on a curved Riemannian manifold.

1 Introduction

The self-organizing map (SOM) [1], also known as the topology-preserving map, can be used to learn the underlying topological neighborhood structure of the input space by applying a very simple adaptation rule. Because of its simplicity and effectiveness, the SOM has been widely used. However, the classical SOM can only reveal flat Euclidean topological neighborhood structure in the input space, and will fail to discover a non-Euclidean topological neighborhood structure of the input space [3].

To solve this problem, many researchers [3,4] have investigated how to combine the classical SOM with kernel methods and this has led to the Kernel-SOM. Different non-Euclidean topological neighborhood structures can be learned by applying different kernel functions, i.e. the resulting non-Euclidean topological neighborhood structure is kernel function dependent. Thus, when the data lie on a specific Riemannian manifold, the non-Euclidean topological neighborhood structure learned by Kernel-SOM is **not** the *intrinsic* topological neighborhood structure of the underlying Riemannian manifold on which the data reside.

In fact, in many real applications, the data may reside on some specific Riemannian manifold. For example, an unit surface normal of an object in $\mathbb{R}^3$ is a data point lying on unit 2-sphere manifold S^2. A set of unit surface normals of an object in $\mathbb{R}^3$ is a data point lying on Riemannian manifold $S^2(n)$, where n is the cardinality of the set. To learn the *intrinsic* topological neighborhood structure

P. Foggia, C. Sansone, and M. Vento (Eds.): ICIAP 2009, LNCS 5716, pp. 229–238, 2009.

of the underlying Riemannian manifold from such data, both the classic SOM and the Kernel-SOM do not apply. This paper aims to overcome this problem by generalizing the classical SOM onto a Riemannian manifold.

To the best of our knowledge, studies on this topic are far less extensive. Ritter [5] proposed method for learning a SOM in non-Euclidean space. However, it can only arrange the output neurons in a spherical or hyperbolic lattice topology, and still uses the Euclidean distance metric for locating the BMU. Shi [6] proposed a geodesic distance based SOM, referred to as GDBSOM. The GDBSOM uses a geodesic distance metric instead of a Euclidean distance metric to locate the BMU. However, when updating the codebook vectors, it does **not** consider how to guarantee that the updated codebook vectors still reside on the underlying manifold. Simila [7] incorporated manifold learning technique (such as LLE) into the classical SOM and proposed the M-SOM. In M-SOM, manifold learning is first applied to learn the internal coordinates on the underlying manifold, next the codebook vectors of the SOM are adapted in both the internal and observation coordinates, with similarities defined on the internal coordinates.

In this paper, we approach with this problem in a quite different style by directly generalizing the classical SOM onto a Riemannian manifold. More specifically, to locate the BMU, the Riemannian geodesic distance metric is applied. In the adaptation step, codebook vectors are adapted along the intrinsic geodesic curve with the aid of the Riemannian *Exponential* and *Log* maps.

2 Self-Organizing Map

The classical self-organizing map (SOM) in a flat Euclidean space (linear manifold) consists of a regular output grid (usually 1- or 2-dimensional), onto which the distribution of input vectors is projected nonlinearly [1]. The mapping tends to preserve the topological-metric relations between input vectors, i.e., similar inputs fed into the SOM will give similar outputs.

A SOM can be represented as a set of output neurons, denoted by $\{i\}_{i=1}^{K}$, on the output layer, where K is the number of neurons on the output layer. Each output neuron i is fully connected to all the input neurons and contains a d-dimensional *codebook* vector $\mathbf{w}_i \in \mathbb{R}^d$ (also referred to as a *prototype* vector). A SOM can be trained by iterative sequential or batch learning algorithms.

Sequential Learning Algorithm: Let $\{\mathbf{x}_i | \mathbf{x}_i \in \mathbb{R}^d\}_{i=1}^{N}$ be the learning dataset. One round of the sequential learning algorithm for SOM is briefly described as follow.

1) Choose the current learning sample: Randomly select a sample from the dataset $\{\mathbf{x}_i\}_{i=1}^{N}$ as the *current* learning sample $\mathbf{x}(t)$.

2) Locate the best-matching unit (BMU): The output neuron whose codebook vector is closest to the current learning sample $\mathbf{x}(t)$ is referred as the BMU, denoted by $c(\mathbf{x}(t))$ and satisfying $c(\mathbf{x}(t)) = \arg\min_{1 \le i \le K} \parallel \mathbf{x}(t) - \mathbf{w}_i(t)$.

3) Update the codebook vectors: The codebook vectors of the BMU and its topological neighborhood output nodes are updated using the following simple rule

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \underbrace{\alpha(t) \cdot h_{c(\mathbf{x}(t)),i}(t) \cdot (\mathbf{x}(t) - \mathbf{w}_i(t))}_{\Delta}. \quad (1)$$

where t denotes learning step, which decreases monotonically with the learning steps and $\alpha(t)$ is the learning rate. The Gaussian neighborhood function defined as

$$h_{c(\mathbf{x}(t)),i}(t) = exp\left(-\frac{\|\mathbf{r}_i - \mathbf{r}_{c(\mathbf{x}(t))}\|^2}{2\sigma^2(t)}\right) \quad (2)$$

where $\mathbf{r}_i$ and $\mathbf{r}_{c(\mathbf{x}(t))}$ are the vector locations of neuron i and neuron $c(\mathbf{x}(t))$ on the output grid, respectively and $\sigma(t)$ is the width of the neighborhood function, which is also decreasing monotonically with the learning steps. Notice that the neighborhood function is not limited to be a Gaussian, and alternative neighborhood functions can also apply [1].

Batch Learning Algorithm: In the sequential learning algorithm, only a single learning sample is fed to the SOM at each learning step, and the codebook vectors of the BMU together with its topological neighborhood nodes are updated. However, in a batch learning algorithm, the entire learning dataset is fed to the SOM before any adaptations are made. The batch learning algorithm can effectively accelerate the leaning process and lessen the impact of the order of the sample data fed to the SOM. When the size of the learning dataset is large, batch learning is to be preferred.

3 Riemannian Manifolds

Let M be a Riemannian manifold. A Riemannian metric on M is a smoothly varying inner product $< \cdot, \cdot >$ on the tangent plane T_wM at each point $w \in M$. The norm of a vector $v \in T_wM$ is given by $\|v\| = < \cdot, \cdot >^{\frac{1}{2}}$. Given a smooth curve segment on M, its length is computed by integrating the norm of the unit tangent vectors along the curve. Let w and x be two points lying on M, the Riemannian distance between them, denoted by $d(w, x)$, is defined as the minimum of lengths over all possible smooth curves between w and x. The *geodesic* is a smooth curve that locally minimizes the length between two points on M.

3.1 Riemannian *Exponential* Map and Riemannian *Log* Map

Let $v \in T_wM$ be a vector on the tangent plane to M at $w \in M$ and $v \neq 0$. γ_w^v be the geodesic that pass through point w (a.k.a. the base point) in the direction of v . The Riemannian *Exponential* map of v at base point w, denoted by $\mathrm{Exp}_w(v)$, maps v to the point, say x, on M along the *geodesic* at distance $\|v\|$ from w, i.e. $x = \mathrm{Exp}_w(v)$.

Note that the *Exponential* map preserves the geodesic distance from the base point to the mapped point, i.e. $d(w,x) = d(w, \mathrm{Exp}_w(v)) = \|v\|$.

The Riemannian *Log* map is the inverse of Riemannian *Exponential* map, i.e. $v = \mathrm{Log}_w(x)$.

3.2 Intrinsic Average and Weighted Intrinsic Average

The intrinsic average of the N points $\{x_1, x_2, \cdots, x_N\}$ lying on a Riemannian manifold M is defined as

$$\overline{x} = \mathbf{IntrinsicAvg}(x_1; x_2; \cdots; x_N) = \arg\min_{x \in M} \sum\nolimits_{i=1}^{N} (d(x, x_i))^2. \tag{3}$$

Pennec [8] first proposed an iterative gradient-descent method to solve the aforementioned minimization problem

$$\overline{x}_{j+1} = \mathrm{Exp}_{\overline{x}_j}\left(\frac{\tau}{N} \sum\nolimits_{i=1}^{N} \mathrm{Log}_{\overline{x}_j}(x_i)\right). \tag{4}$$

where τ is the step size. The uniqueness of the solution can be guaranteed when the data are well localized.

Let w_i be the weight value of x_i, $w_i \geq 0$, $1 \leq i \leq N$. Likewise, the weighted intrinsic average $\overline{x} = \mathbf{WIntrinsicAvg}(w_1, x_1; w_2, x_2; \cdots; w_N, x_N)$, can be computed using the following iteration equation:

$$\overline{x}_{j+1} = \mathrm{Exp}_{\overline{x}_j}\left(\frac{\tau}{\sum_{i=1}^{N} w_i} \sum\nolimits_{i=1}^{N} w_i \cdot \mathrm{Log}_{\overline{x}_j}(x_i)\right). \tag{5}$$

Embedding: Every Riemannian manifold M can be isometrically embedded into some Euclidean space $\mathbb{R}^d$, i.e., there exists embedding mapping $\Phi : M \to \mathbb{R}^d$, which embed M into its ambient Euclidean space $\mathbb{R}^d$. Under the embedding Φ, points on M can be depicted by corresponding vectors in the Euclidean space $\mathbb{R}^d$. For example, a unit 2-sphere manifold S^2 can be embedded into a 3-dimensional Euclidean space by defining embedding mapping $\Phi : M \to \mathbb{R}^3$. Under this embedding, data point w lying on manifold S^2 can be depicted by a corresponding vector $\mathbf{w}$ in 3-dimensional Euclidean space. From now on, data point on M, unless otherwise stated, is depicted by the corresponding vector in the ambient Euclidean space $\mathbb{R}^d$.

Exponential and Log maps on $S^2(n)$: We commence by considering how to implement Riemannian *Exponential* map and Riemannian *Log* map on a unit 2-sphere manifold S^2, which has been embedded into the Euclidean space $\mathbb{R}^3$.

Let point $\mathbf{p}_0 = (0, 0, 1)^T \in S^2$ be the base point, then a vector on the tangent plane $T_{\mathbf{p}_0}S^2$ can be written in the form of $\mathbf{v} = (v_x, v_y)^T$. Notice that here we choose the x and y axes of the ambient Euclidean space $\mathbb{R}^3$ as the coordinate system of $T_{\mathbf{p}_0}S^2$. Thus the Riemannian *Exponential* map on S^2 with base point $\mathbf{p}_0$ is given as [9]

$$\mathrm{Exp}_{\mathbf{p}_0}(\mathbf{v}) = \left(v_x \cdot \frac{\sin\|\mathbf{v}\|}{\|\mathbf{v}\|}, v_y \cdot \frac{\sin\|\mathbf{v}\|}{\|\mathbf{v}\|}, \cos\|\mathbf{v}\|\right)^T. \tag{6}$$

The corresponding Riemannian *Log* map for a point $\mathbf{q} = (q_x, q_y, q_z)^T$ on S^2 is given by

$$\mathrm{Log}_{\mathbf{p}_0}(\mathbf{q}) = \left(q_x \cdot \frac{\beta}{\sin\beta}, q_y \cdot \frac{\beta}{\sin\beta} \right)^T. \tag{7}$$

where $\beta = \arccos(q_z)$. Notice that the antipodal point $-\mathbf{p}_0$ is not in the domain of the Log map.

As stated above, a unit vector $\mathbf{p} \in \mathbb{R}^3$ can be considered as a point on the manifold S^2. Thus, a matrix $\mathbf{U} \in \mathbb{R}^{n\times 3}$, in which each row is a unit vector, can be considered as a point on manifold $S^2(n) = \prod_{i=1}^{n} S^2$. The *Exponential* and *Log* maps for $S^2(n)$ are just the direct products of n copies of the corresponding maps for S^2.

4 Learning SOM on Riemannian Manifold

In this section, we consider how to learn a SOM from data lying on a Riemannian manifold M. Note that the Riemannian manifold M has been embedded into an appropriate Euclidean space R^d by the embedding mapping $\Phi : M \to R^d$.

Sequential Learning Algorithm: First, we illustrate why the classic SOM can not *accurately* learn the *intrinsic* topological neighborhood structure of the underlying Riemannian manifold on where the data reside. Let us commence by starting from a simple case where the data reside on the Riemannian manifold S^2. Note that S^2 has been embedded into the Euclidean space $\mathbb{R}^3$ as shown in Fig.1.

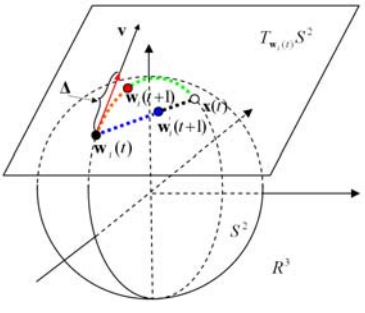

Fig. 1. Adaptation of codebook vector on S^2

To learn a SOM on S^2, if we use the classical updating equation (1), then the resulting effect is to move the *black* point corresponding to vector $\mathbf{w}_i(t)$ to the *blue* point corresponding to vector $\mathbf{w}_i'(t+1)$ by a distance of $\|\triangle\|$ along the direction of $\mathbf{x}(t) - \mathbf{w}_i(t)$, as shown in Fig.1. Note that the resulting data point corresponding to the updated codebook vector $\mathbf{w}_i'(t+1)$ is **not** guaranteed to

still reside on the manifold S^2. Thus the classical SOM fails to accurately learn the *intrinsic* geometric and topological properties of data lying on S^2.

The correct procedure of for moving $\mathbf{w}_i(t)$ closer to the current learning sample $\mathbf{x}(t)$ is to move the *black* point corresponding to vector $\mathbf{w}_i(t)$ to the *red* point by a distance of $\|\triangle\|$ along the *geodesic* between $\mathbf{w}_i(t)$ and $\mathbf{x}(t)$. This can be implemented by three successive steps as follows

1) Map point $\mathbf{x}(t)$ to the vector $\mathbf{v}$ on the tangent plane using Riemannian *Log* map.

2) Calculate $\triangle = \alpha(t) \cdot h_{c(\mathbf{x}(t)),i}(t) \cdot \mathrm{Log}_{\mathbf{w}_i(t)}(\mathbf{x}(t))$, the magnitude and the direction of adjustment, with respect to $\alpha(t)$ and $h_{c(\mathbf{x}(t)),i}(t)$.

3) Map the $\triangle$ back to the point on manifold S^2 using the Riemannian *Exponential* map.

The aforementioned 3 steps can be formulated as the following update equation

$$\mathbf{w}_i(t+1) = \overbrace{\mathrm{Exp}_{\mathbf{w}_i(t)}\left(\underbrace{\alpha(t) \cdot h_{c(\mathbf{x}(t)),i}(t) \cdot \overbrace{\mathrm{Log}_{\mathbf{w}_i(t)}(\mathbf{x}(t))}^{\text{(step 1)}}}_{\text{(step 2)},\triangle}\right)}^{\text{(step 3)}}. \tag{8}$$

Batch Learning Algorithm: Likewise, for data that reside on a Riemannian manifold M, points corresponding to the updated codebook vectors are not guaranteed to still reside on M. In fact, batch learning algorithm for a SOM on a Riemannian manifold should first compute the *intrinsic* average $\overline{\mathbf{x}}_j$ of samples in each Voronoi region V_j, and then update the codebook vector to a *weighted intrinsic average* of $\overline{\mathbf{x}}_j, 1 \leq j \leq K$. More specifically, the update steps should be

$$\overline{\mathbf{x}}_j = \mathbf{IntrinsicAvg}(\mathbf{x}_1; \mathbf{x}_2; \cdots; \mathbf{x}_{n_j}). \tag{9}$$

$$\mathbf{w}_i(t+1) = \mathbf{WIntrinsicAvg}(w_{1i}, \overline{\mathbf{x}}_1; \cdots; w_{Ki}, \overline{\mathbf{x}}_K). \tag{10}$$

where w_{ji} is the weight value of $\overline{\mathbf{x}}_j$, defined as

$$w_{ji} = \frac{h_{ji}(t) \cdot n_j}{\sum_{j=1}^{K} h_{ji}(t) \cdot n_j}. \tag{11}$$

The right-sides of (9) and (10) can be solved by applying (4) and (5), respectively.

Initialization: Another key issue in our proposed learning algorithms is the initialization of the SOM. An appropriate initialization scheme must guarantee that the points corresponding to the initialized codebook vectors lie on the Riemannian manifold M, from where data are sampled.

We here present a simple randomized initialization scheme. When initializing the codebook vector $\mathbf{w}_i$, we first randomly select two samples, denoted by $\mathbf{x}$ and $\mathbf{y}$, from dataset $\{\mathbf{x}_i\}_{i=1}^{N}$. Then, $\mathbf{w}_i$ is initialized by taking the mid point between

$\mathbf{x}$ and $\mathbf{y}$ using $\mathbf{w}_i = \mathrm{Exp}_{\mathbf{x}}(\frac{1}{2} \cdot \mathrm{Log}_{\mathbf{x}}(\mathbf{y}))$. We repeat the above procedure K times until all the codebook vectors of the SOM have been initialized.

5 Experimental Results

In this section, we compare the performance between the classic SOM and the generalized SOM by performing experiments on both synthetic data and real word 3D facial shape data.

Synthetic Data: We use standard Swiss-roll datasets in this section. We randomly select 2000 points in Swiss-roll dataset. These points are used as training samples. After training, the distribution of the codebooks of the SOM are plotted.

Results are shown in Fig.2. The left plot in Fig.2 is the distribution of 2000 sample points, the center and right plots are the learning results of the classical SOM and the generalized SOM, respectively.

The center plot in Fig.2 clearly demonstrates that the updated codebook vectors in the classical SOM can not be guaranteed to still reside on the underlying manifold, as claimed in previous section. Thus, the resulting codebook vectors can not accurately reflect the structure of the underlying manifold. However, since the generalized SOM moves the codebook vectors along geodesics, they can be guaranteed to still reside on the underlying manifold, as depicted in the right plot in Fig.2.

From Fig.2, we can easily conclude that compared with the classical SOM, the generalized SOM can more accurately learn the topological neighborhood structure of the underlying manifold.

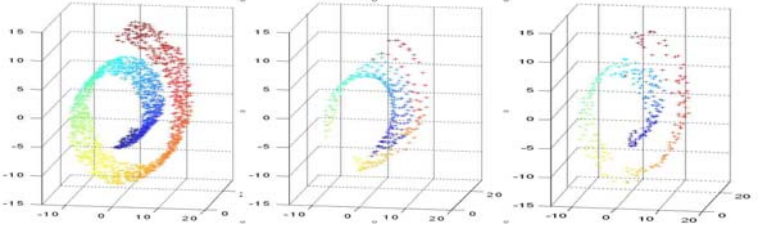

Fig. 2. Experimental results on Swiss-roll data. Left: samples of Swiss-roll data. Center: learning result of the classical SOM. Right: learning result of the generalized SOM.

Real Data: Our second application involves facial needle-map (a set of facial surface normals) as the representation of a 3D facial surface shape.

To explore the effectiveness of the generalized SOM, both facial needle-maps extracted from range images, referred as Dataset-I, and the facial needle-maps recovered from 2D brightness images using shape-from-shading, referred as

Dataset-II, are used in our experiments. Dataset-I contains range images obtained from UND database [10], and Dataset-II is a set of facial needle-maps recovered from FERET database by applying non-Lambertian shape-from-shading method, which was proposed by Smith&Hancock [11].

Dataset-I: Facial Needle-maps from UND: Dataset-I is obtained from the biometrics database from University of Notre Dame [10]. UND biometrics database provides both 3D range images and 2D face images for each subject. We select 200 range images for 200 subjects(100 females and 100 males, each subject has only one range image). Each selected 3D range image is geometrically aligned and get a corresponding facial surface height matrix $\mathbf{H}_{114\times100}$. Entry value $\mathbf{H}(i,j)$ is the facial surface height value corresponding to facial image location (i,j). The needle-map $\mathbf{N}_{114\times100\times3}$ then can be directly computed from the facial surface height matrix $\mathbf{H}_{114\times100}$ of the aligned 3D range image. More details about constructing Dataset-I can be found in Wu&Hancock [12].

Dataset-II: Recovered Facial Needle-maps from FERET: FERET [13] - the outcome of FERET program sponsored by DARPA - has become a standard face image database in the face recognition literatures. Each subject has different facial images with a variety of pose, angle of view, illumination, expression and age. Different gender and ethnicity categories are also covered by the database. In our experiments, 200 frontal facial images for 200 subjects are selected from FERET (103 females and 97 males, each subject has only one face image). Each selected image is pre-processed to a resolution of 142×124 by cropping, rotating, scaling and aligning. Face images are aligned with each other based on the central points of the left and right eyes.

After pre-processing, we use non-Lambertian shape-from-shading to recover the needle-map $\mathbf{N}_{142\times124\times3}$ for each facical image [11].

Results and Analysis: In the following experiments, the size of SOM is set to be 30×16 and the output neurons of the SOM are arranged in 2D hexagonal lattice structure. Batch learning algorithms for both the classical SOM and the generalized SOM are implemented. Note that 10-fold cross validation is applied.

Columns 2 and 3 of Table 1 show the experimental results for both the classical SOM and the generalized SOM on Dataset-I, while columns 4 and 5. Table present the experimental results on Dataset-II. From the table, it is clear that

Table 1. Number of erroneously classified needle-maps and corresponding accuracy rate

sub-block size	Method			
	classical SOM (I)	general SOM (I)	classical SOM (II)	general SOM (II)
$4\times4\times3$	31 (84.5%)	25 (87.5%)	34 (83.0%)	21 (89.5%)
$8\times8\times3$	36 (82.0%)	27 (86.5%)	37 (81.5%)	26 (87.0%)
$16\times16\times3$	50 (75.0%)	39 (80.5%)	37 (81.5%)	26 (87.0%)
$32\times32\times3$	53 (73.5%)	48 (76.0%)	55 (72.5%)	36 (82.0%)

both classical SOM and the generalized SOM achieve the best accuracy rates when the size of sub-block of is $4\times4\times3$. The best accuracy rate for the generalized SOM on Dataset-I is 87.5%, which is 3% higher than that (84.5%) obtained with the classical SOM. The best accuracy rate for the generalized SOM on Dataset-II is 89.5%, which is 6.5% higher than that (83.0%) obtained using the classical SOM.

We also carried out experiments by gradually changing the size of sub-block from $4 \times 4 \times 3$ to $32 \times 32 \times 3$. From the table the accuracy rates for both the classical SOM and the generalized SOM decrease when the size of sub-block becomes large. However, the generalized SOM consistently outperforms the classical SOM under different sub-block sizes on both datasets.

6 Conclusion

In this paper, we generalize the classical SOM from a flat Euclidean space to a curved Riemannian manifold. Both sequential and batch learning algorithms for learning a SOM on a Riemannian manifold are presented. We prove that the classic SOM learning algorithms are a special cases of the generalized learning algorithms of the generalized SOM. The generalized SOM can learn the *intrinsic* topological neighborhood structure of the underlying Riemannian manifold on where the data reside. We compare the generalized SOM with the classical SOM on both synthetic data and real world facial surface normal data. Experimental results show that the generalized SOM outperform the classical SOM when the data lie on curved manifold.

References

1. Kohonen, T.: Self-Organization and Associative Memory, 2nd edn. Springer, Berlin (1988)
2. Kohonen, T., Kaski, S., Lagus, K., Salojarvi, J., Honkela, J., Paatero, V., Saarela, A.: Self Organization of a Massive Document Collection. IEEE Transactions on Neural Networks 11(3), 574–585 (2000)
3. Andras, P.: Kernel-Kohonen Networks. Int. J. Neural. Syst. 12(2), 117–135 (2002)
4. Manuel, A.M., Alberto, M.: Extending the SOM algorithm to Non-Euclidean Distances via the Kernel Trick. In: 11th International Conference on Neural-Information-Processing, pp. 150–157. Springer, Calcutta (2004)
5. Ritter, H.: Self-organiring Maps on non-Euclidean Spaces. In: WSOM 99, pp. 1321–1344. IEEE Press, Espoo (1999)
6. Shi, C.Q., Zhang, S.L., Shi, Z.Z.: Geodesic Distance based SOM for Image Clustering. In: International Conference on Sensing, Computing and Automation, pp. 2483–2488. Watam Press, Chongqing (2006)
7. Simila, T.: Self-organizing Map Learning Nonlinearly Embedded Manifolds. J. Information Visualization 4, 22–31 (2005)
8. Pennec, X.: Probabilities and Statistics on Riemannian Manifolds: Basic Tools for Geometric Measurements. In: IEEE Workshop on Nonlinear Signal and Image Processing, pp. 194–198. IEEE Press, Antalya (1999)

9. Fletcher, P.T., Conglin, L., Pizer, S.M., Joshi, S.: Principal Geodesic Analysis for The Study of Nonlinear Statistics of Shape. IEEE Transactions on Medical Imaging 23(8), 995–1005 (2004)
10. Chang, K., Bowyer, K.W., Flynn, P.J.: Face Recognition using 2D and 3D Facial Data. In: ACM Workshop on Multimodal User Authentication, pp. 25–32. ACM Press, Santa Barbara (2003)
11. Smith, W.A.P., Hancock, E.R.: New Framework for Grayscale and Colour Non-Lambertian Shape-from-shading. In: ACCV, pp. 869–880. Springer, Heidelberg (2007)
12. Wu, J., Smith, W.A.P., Hancock, E.R.: Gender Classification using Shape from Shading. In: BMVC, UK, pp. 499–508 (2007)
13. Phillips, P.J., Moon, H., Rauss, P.J., Rizvi, S.: The FERET Evaluation Methodology for Face Recognition Algorithms. Transactions on Pattern Analysis and Machine Intelligence 22(10), 1090–1104 (2000)