Fast Approximation of Distance Between Elastic Curves using Kernels

Hedi Tabia, David Picard, Hamid Laga, Philippe-Henri Gosselin

Fast Approximation of Distance Between Elastic Curves using Kernels

Hedi Tabia[1]
http://perso-etis.ensea.fr/tabia
David Picard[1]
http://perso-etis.ensea.fr/~picard
Hamid Laga[2]
http://people.unisa.edu.au/Hamid.Laga
Philippe-Henri Gosselin[1,3]
http://perso-etis.ensea.fr/~gosselin/

[1] ETIS/ENSEA, University of Cergy-Pontoise, CNRS, UMR 8051, France
[2] Phenomics and Bioinformatics Research Centre, University of South Australia, Australia
[3] INRIA Rennes Bretagne Atlantique, France

Abstract

Elastic shape analysis on non-linear Riemannian manifolds provides an efficient and elegant way for simultaneous comparison and registration of non-rigid shapes. In such formulation, shapes become points on some high dimensional shape space. A geodesic between two points corresponds to the optimal deformation needed to register one shape onto another. The length of the geodesic provides a proper metric for shape comparison. However, the computation of geodesics, and therefore the metric, is computationally very expensive as it involves a search over the space of all possible rotations and re-parameterization. This problem is even more important in shape retrieval scenarios where the query shape is compared to every element in the collection to search. In this paper, we propose a new procedure for metric approximation using the framework of kernel functions. We will demonstrate that this provides a fast approximation of the metric while preserving its invariance properties.

1 Introduction

Shape comparison and registration are fundamental problems and building blocks to many applications in pattern recognition, computer vision, and computer graphics. Examples include scan registration, deformable shape matching [5], animation reconstruction and shape classification and retrieval. Many problems that appear in these applications can be formulated as the problem of matching and comparing 2D or 3D elastic curves. This includes 3D shape retrieval [14], 3D face recognition [10], facial expression recognition [7] and gesture and action recognition [1]. They involve the representation of shapes with curves, the description of the curves with some local or global descriptors and then matching these descriptors using some similarity metric. One common way of describing the shape of curves involves defining and measuring specific geometric or topological characteristics such as shape area, contour length, orientation and curvature. This can be seen as a mapping of the shape onto some feature space. Shape comparison is then reduced to the computation of

distances in the feature space. Feature based analysis of shapes are very popular in retrieval and classification tasks since they provide an easy way to compute compact shape signatures. However, not every shape property can be represented with these features limiting the type of analysis that can be done in the feature space.

Elastic shape analysis on non-linear Riemannian manifolds provides an efficient and elegant way for simultaneous comparison and registration of non-rigid shapes. In such formulation, shapes become points on some high dimensional shape space. A geodesic between two points corresponds to the optimal deformation needed to register one shape onto another. The length of the geodesic provides a proper metric for shape comparison [3, 5, 8]. Joshi et al. [3] and Srivastava et al. [13] proposed the Square Root Velocity Function parametrization (SRVF) that allow to compute geodesic distances between closed curves $\mathbb{R}^n$. The distance is invariant to different shape preserving geometric transformation including translation, rotation and re-parametrization. The computation of geodesics, and therefore the metric, is computationally very expensive as it involves a search over the space of all possible rotations and re-parameterizations. This problem is even more important in shape retrieval scenarios where the query shape is compared to every element in the collection to search.

In this paper, we propose a fast approximation of this metric, using Kernel functions, while keeping its nice properties such as the invariance to geometric transformations. We search a subspace, equipped with the standard dot product, and in which the discriminative power of the original distance between two curves is retained. The key idea is to design a kernel function $k(\cdot,\cdot)$ associated with the elastic metric, and build a mapping function such that the dot product between mapped elements is as close as possible to the original kernel $k(x,y)$. This mapping is based on the Nyström method for kernel approximation [2, 16]. The advantage of this formulation is that the heavy-computational metric becomes now a dot product in the new subspace with very low dimensions. This reduces significantly the computation time needed to compare one shape to all the elements of the collection to search. We further show that the approximated distance preserves the invariance properties and achieves retrieval performance that is competitive with the original metric.

The remainder of this paper is organized as follows: Section 2.1 reviews the Riemannian elastic metric for shape analysis. Section 2.2 details the distance approximation method using kernel. Experimental results validating the proposed method are presented in Section 3.

2 The method

We are given a set of curves $C = \{\beta_i, i = 1,\ldots,n\}$, each curve β_i is represented with its Squared Root Velocity Function (SRVF) q_i. The first step is to define a kernel k as a function of the geodesic distance (metric in the Riemannian shape space) between two curves (Section 2.1). Then, we learn in a supervised manner a subspace that preserves the nice properties of the Riemannian metric and such that the dot product in the subspace corresponds to the metric in the Riemannian shape space (Section 2.2). We propose an algorithm that learns this subspace only using few samples from the collection C (Section 2.3). Let m be the number of training samples such as $m \ll n$.

At runtime, a query is first projected on the subspace and then compared to the other elements in the database using the dot product in the subspace (Section 2.2). The projection procedure involves the computation of the geodesic distance between the query model and the training samples and the angular distance with the other samples in the collection. However, since $m \ll n$, it leads to a significant reduction in computational complexity as

demonstrated in our experimental results (Section 3).

2.1 Curve analysis background

Let us consider a closed curve $\beta : \mathbb{S}^1 \to \mathbb{R}^3$. To analyze the shape of β, we mathematically represent it using a *square-root velocity function* (SRVF) $q(t) \doteq \frac{\dot{\beta}(t)}{\sqrt{\|\dot{\beta}(t)\|}}$. $q(t)$ is a special function introduced by [3] that captures the shape of β and is particularly convenient for shape analysis. It has been demonstrated in [3] that the elastic metric used for comparing shapes of curves becomes an $\mathbb{L}^2$-metric under the SRVF representation. This point is very important as it simplifies the calculus of elastic metric to the well-known calculus of functional analysis under the $\mathbb{L}^2$-metric.

In order to restrict our shape analysis to closed curves, we define the set: $\mathcal{C} = \{q : \mathbb{S}^1 \to \mathbb{R}^3 | \int_{\mathbb{S}^1} q(t)\|q(t)\|dt = 0\} \subset \mathbb{L}^2(\mathbb{S}^1, \mathbb{R}^3)$. Here $\mathbb{L}^2(\mathbb{S}^1, \mathbb{R}^3)$ denotes the set of all functions from $\mathbb{S}^1$ to $\mathbb{R}^3$ that are square integrable. The quantity $\int_{\mathbb{S}^1} q(t)\|q(t)\|dt$ denotes the total displacement in $\mathbb{R}^3$ as one traverses along the curve from start to end. Setting it equal to zero is equivalent to having a closed curve. Note that the elements of $\mathcal{C}$ are allowed to have different lengths. Due to a nonlinear (closure) constraint on its elements, $\mathcal{C}$ is a nonlinear manifold. For any $q \in \mathcal{C}$, let us denote the tangent space: $T_q(\mathcal{C}) = \{v : \mathbb{S}^1 \to \mathbb{R}^3 | \langle v, w \rangle = 0,\ w \in N_q(\mathcal{C})\}$, where $N_q(\mathcal{C})$, the space of normals at q is given by: $N_q(\mathcal{C}) = \text{span}\left\{ \frac{q^1(t)}{\|q(t)\|}q(t) + \|q(t)\|\mathbf{e}^1,\ \frac{q^2(t)}{\|q(t)\|}q(t) + \|q(t)\|\mathbf{e}^2, \frac{q^3(t)}{\|q(t)\|}q(t) + \|q(t)\|\mathbf{e}^3 \right\}$ and where $\{\mathbf{e}^1, \mathbf{e}^2, \mathbf{e}^3\}$ form an orthonormal basis of $\mathbb{R}^3$. We can make $\mathcal{C}$ a Riemannian manifold by using the following metric: for any $u, v \in T_q(\mathcal{C})$,

$$\langle u, v \rangle = \int_{\mathbb{S}^1} \langle u(t), v(t) \rangle \, dt \ . \tag{1}$$

The metric on the left side is in $\mathcal{C}$ while the metric inside the integral is the standard Euclidean inner product in $\mathbb{R}^3$.

In order to make the metric invariant to rotation and re-parameterization, Srivastava et al. [13] defines orbits of the rotation group $SO(3)$ and the re-parameterization group Γ as equivalence classes in $\mathcal{C}$. Here, Γ is the set of all orientation-preserving diffeomorphisms of $\mathbb{S}^1$ (to itself) and the elements of Γ are viewed as re-parameterization functions.

For example, for a curve $\beta : \mathbb{S}^1 \to \mathbb{R}^3$ and a function $\gamma : \mathbb{S}^1 \to \mathbb{S}^1$, $\gamma \in \Gamma$, the curve $\beta(\gamma)$ is a re-parameterization of β. The corresponding SRVF changes according to $q(t) \mapsto \sqrt{\dot{\gamma}(t)}q(\gamma(t))$. We set the elements of the set

$$[q] = \{\sqrt{\dot{\gamma}(t)}Oq(\gamma(t)) | O \in SO(3),\ \ \gamma \in \Gamma\}$$

to be equivalent from the perspective of shape analysis. The set of such equivalence classes, denoted by $\mathcal{S} \doteq \mathcal{C}/(SO(3) \times \Gamma)$ is called the *shape space* of closed curves in $\mathbb{R}^3$. $\mathcal{S}$ inherits a Riemannian metric from the larger space $\mathcal{C}$ and is thus a Riemannian manifold itself.

The main ingredient in comparing and analyzing shapes of curves is the construction of a geodesic between any two elements of $\mathcal{S}$, under the Riemannian metric of Eq. 1. Given two curves β_1 and β_2, represented by their SVRFs q_1 and q_2, we want to compute a geodesic path between the orbits $[q_1]$ and $[q_2]$ in the shape space $\mathcal{S}$. This task is accomplished using a *path straightening approach* which was introduced in [9].

The basic idea here is to connect the two points $[q_1]$ and $[q_2]$ by an arbitrary initial path α and to iteratively update this path using the negative gradient of an energy function

$E[\alpha] = \frac{1}{2}\int_s \langle \dot{\alpha}(s), \dot{\alpha}(s) \rangle ds$. The interesting part is that the gradient of E has been derived analytically and can be used directly for updating α. As shown in [4], the critical points of E are actually geodesic paths in $\mathcal{S}$. Thus, this gradient-based update leads to a feature point of E which, in turn, is a geodesic path between the given points. We will use the notation $d(\beta_1, \beta_2)$ to denote the geodesic distance, or the length of the geodesic in $\mathcal{S}$, between elements of $\mathcal{S}$ which represent two curves β_1 and β_2.

Although the elastic metric becomes the $\mathbb{L}^2$-metric under the SRVF representation, the removal of the rotations and re-parameterizations is computationally very expensive. We propose in the next section a fast approximation of this metric while preserving its nice invariance properties.

2.2 Distance approximation using kernels

Given the elastic distance between two curves $d(\beta_1, \beta_2)$, we consider a kernel function $k(\beta_1, \beta_2)$ associated with it. The approximation we propose finds a mapping function P to some specific space such that the dot product between mapped elements $P(\beta_1)^\top P(\beta_2)$ is as close as possible to the original kernel $k(\beta_1, \beta_2)$. We first consider the expression of the triangular kernel [11], $k(\beta_1, \beta_2)$ associated to $d(\beta_1, \beta_2)$ as follows:

$$k(\beta_1, \beta_2) = 1 - \frac{d(\beta_1, \beta_2)^2}{2} \tag{2}$$

With this expression of k, we derive the projection into a lower dimensional space that preserves most of the metric proprieties. Let us consider a training set $\mathcal{A} = \{\beta_i\}$ used for the training of the projection $P(\cdot)$. We want P such that $\forall \beta_i, \beta_j \in \mathcal{A}$, $P(\beta_i)^\top P(\beta_j) = k(\beta_i, \beta_j)$. The Gram matrix of the kernel k on $\mathcal{A}$ is given by $K = [k(\beta_i, \beta_j)]_{\beta_i, \beta_j \in \mathcal{A}}$. We propose to approximate the matrix K by a low rank version and to compute the corresponding projection. This is known as the Nyström approximation for kernels, and has been used to speed up large scale kernel based classifiers in machine learning [2, 16]. Since k is a Mercer kernel and thus positive semi-definite, we can compute its eigen-decomposition: $K = V \Lambda V^\top$. The non-linear projection $P(\cdot)$ is then given by:

$$P(\beta) = \Lambda^{-\frac{1}{2}} V^\top [k(\beta_i, \beta)]_{\beta_i \in \mathcal{A}} \tag{3}$$

Where $[k(\beta_i, \beta)]_{\beta_i \in \mathcal{A}}$ is the vector of entry-wise computation of the kernel between β and the elements of the training set. Let us consider the matrix Y of the projected elements of $\mathcal{A}$: $Y = [P(\beta)]_{\beta \in \mathcal{A}}$. The Gram matrix in the projection space using the linear kernel is thus:

$$Y^\top Y = (\Lambda^{-\frac{1}{2}} V^\top K)^\top \Lambda^{-\frac{1}{2}} V^\top K = K V \Lambda^{-1} V^\top K = K K^{-1} K = K \tag{4}$$

Furthermore, the Euclidean distance between mapped elements perfectly fits to the original distance on the training set, $\forall \beta_i, \beta_j \in \mathcal{A}$:

$$\begin{aligned} \|P(\beta_i) - P(\beta_j)\|^2 &= P(\beta_i)^\top P(\beta_i) + P(\beta_i)^\top P(\beta_i) - 2P(\beta_i)^\top P(\beta_j) \\ &= k(\beta_i, \beta_i) + k(\beta_j, \beta_j) - 2k(\beta_i, \beta_j) \\ &= 1 - \frac{d(\beta_i, \beta_i)^2}{2} + 1 - \frac{d(\beta_j, \beta_j)^2}{2} - 2 + 2\frac{d(\beta_i, \beta_j)^2}{2} \\ &= d(\beta_i, \beta_j)^2. \end{aligned}$$

The more $\mathcal{A}$ is a good sampling of the original space, the closer the linearization P along with the dot product are to the original kernel k and thus the better the distance $d(\cdot,\cdot)$ is approximated. Some of the eigenvalues of the Kernel matrix are often very small compared to the other ones and can be safely discarded to further gain in computational efficiency. In this case, the reconstruction of the Gram matrix (respectively the distance matrix) on the training set is not perfect. However, the associated projectors often encode noise in the data, and discarding them can act as a de-noising process, as a PCA would do with linear projections. Since we are using a non-linear mapping, this process is analog to Kernel PCA [12].

2.3 Training set selection using active learning

As the choice of the training set $\mathcal{A}$ is essential to ensure the generalization capability of the approximation process, we extend the common uniform sampling used in Nyström approximation as follows. We propose an iterative process where examples are added to the training set so as to minimize the error in the mapped space at each iteration. The key idea is analogous to that of *active learning* used in machine learning algorithms [15]. In classification, active learning aims at adding to the training set the samples that will enhance the most the classifier. In our case, we add elements to the training set so as to maximize the gain in error reduction of the new projection compared to the old one. However, such criterion is difficult to compute. Instead, we propose to add to the training set the elements that are subject to the highest approximation error using the current projection. This is analog to adding the most uncertain samples (most misclassified) in the classification context of [15]. Algorithm 1 details this procedure. In order to build a training set of size M, we repeat the active selection until M samples have been added to the set. At each iteration, a sub-sample of T curves of the collection is examined and the N with the highest approximation error are added to $\mathcal{A}$.

Algorithm 1 : Training set selection using active learning

1: Select N_0 samples randomly to populate the initial training set $\mathcal{A}_0$, and compute the resulting projectors P_0. Set $t \leftarrow 0$.
2: Select randomly a set $\mathcal{B}_t = \{\beta\}$ of T samples.
3: Compute their projection $P_t(\beta)$
4: Compute for each sample in $\mathcal{B}_t$ the mean quadratic error between the real distance and the approximation $err(\beta) = \sum_{\beta_i \in \mathcal{A}_t} (d(\beta,\beta_i) - \|P_t(\beta) - P_t(\beta_i)\|)^2$
5: Add the N samples with maximal $err(\beta)$ to $\mathcal{A}_t$ to form the set $\mathcal{A}_{t+1}$
6: Train the projector P_{t+1} on $\mathcal{A}_{t+1}$
7: If the size of $\mathcal{A}_{t+1} < M$, set $t \leftarrow t+1$ and go to 2

In the best case scenario, T should be equal to the size of the collection (*i.e.* all samples are considered at each active learning round), and N should be equal to 1 (*i.e.* only one sample is added at a time). To reduce the overall complexity of the algorithm, we propose to examine only a sub-sample of T elements (sampled uniformly). Thus we avoid the computation of the costly distance d for all samples in the collection. We also propose to add $N > 1$ training samples to the current training set at each iteration to reduce the number of active learning rounds needed to reach a training set size of M.

3 Experiments and results

We illustrate the effectiveness of the proposed method using three different experiments: shape retrieval from two databases and 3D face recognition application.

The first one, a 2D curve dataset provided by Srivastava et al. [13], contains 65 shape categories with 1300 shapes in total. The second one is the Multiview Curve Database (MCD). It has been proposed by Zuliani et al. [17] to test the performance of shape descriptors in the presence of perspective distortions. 40 shapes (taken from the MPEG-7 shape database) have been imaged under seven different points of view. Then, an arbitrary rotation has been applied to the contour of each image. Finally, seven new contours have been generated by mirroring the original ones. The dataset contains thus 14 contours per object (40 objects in total). For the 3D face recognition application, we use a subset of 100 faces selected from the FRGC database [9].

To objectively evaluate the performance of the proposed distance, we compute its Mean Square Error (MSE) to the original distance. We also evaluated its classification performance using five quality metrics, namely: the Nearest Neighbor (NN), First-tier (1-Tier), Second-tier (2-Tier), E-Measures and Discounted Cumulative Gain (DCG).

3.1 Shape retrieval from 2D curve dataset

We first randomly select 20 curves to form the initial training set and iteratively add 5 curves with maximal error according to Algorithm 1 until we form the final set of 100 curves, which is shown in Figure 1.

Figure 3 plots the similarity matrices. Each element (i, j) of these matrices represents the similarity between two curves (i, j). Red color represents better matches (high similarity score), while blue elements indicate bad matches (low similarity scores). The first five matrices in the figure represent the similarities between 100 curves selected from the whole dataset, and shown in Figure 2, using the approximated distance. These curves have not been used in the training stage. Figures 3-(a), (b), (c), (d) and (e) show the similarities obtained when using respectively $60, 50, 40, 20, 15$ dimensions for the signature. We show also in Figure 3-(f) the pairwise similarity matrix between the same curves using the original distance [13]. We see that using only a signature of size 20, our approach behaves in a similar way to the original distance. A dimension lower than 20 (e.g., 15) reduces the performance.

Finally, we quantitatively evaluate the proposed metric on the whole dataset using classification performance measures and compare it to the original one. Table 1 shows the results. As we can see, using a signature of 50 and 20 dimensions, the method performs similarly to the original distance. When using 50 dimensions, we reach the lowest error value that measures the deviation from the original distance. This is confirmed by the performance, of the 50-dimension signatures, which is very close to the performance of the original distance measure. Using 20 dimensions the MSE is higher than using a signature of 50 dimensions. However the performance of the method is still close to the original one.

Interestingly, when using a signature of 20 dimensions only, the approximated distance outperforms the original one in terms of 2-Tier and E-Measure. This can be due to the de-noising effect produced when taking into account only relevant dimensions.

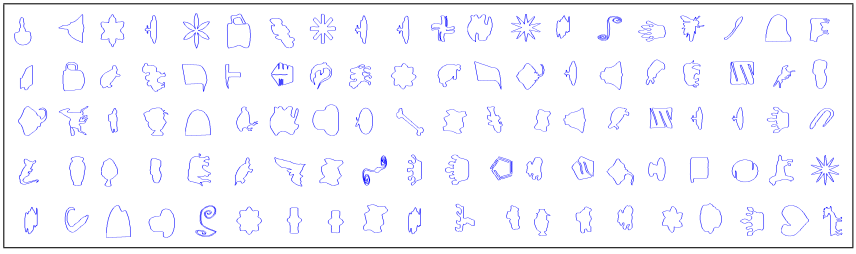


Figure 1: 100 curves representing the training dataset. These curves are selected using Algorithm 1 from the dataset.

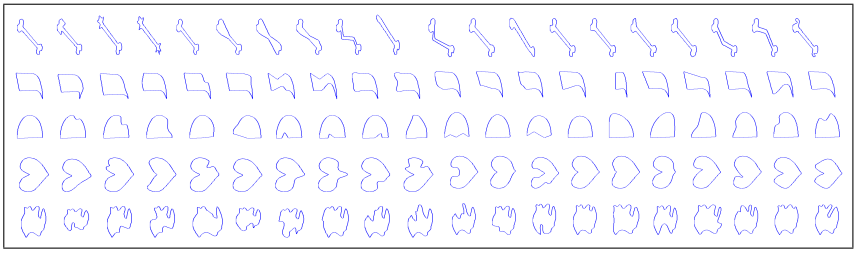


Figure 2: 100 curves selected from the test dataset

3.2 Shape retrieval from the MCD dataset

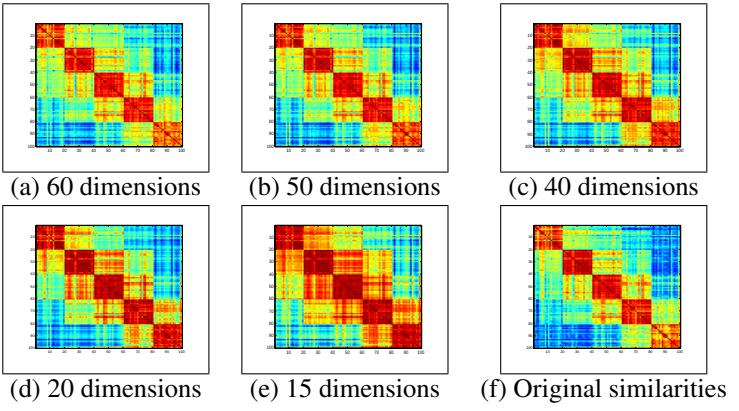
We also evaluate our approach on the MCD dataset using the same evaluation procedure described in Section 3.1. Table 2 shows the results. From this table, one can notice that the approximated distance performs similarly to the original one using a signature of 50 and 20 dimensions. Here also when using 50 dimensions, we find the lowest error value that measures the deviation from the original distance and the signatures have more discriminative power than using only 20 dimensions.

3.3 Application to 3D face recognition

In this experiment, we take 100 faces from the FRGC dataset [9]. In total we consider 50 subject each one represented with two faces. We represent each 3D face with a set of closed curves. To build these curves, we first detect the nose tip, compute at each vertex the geodesic distance from that vertex to the nose tip and then extract the level sets of the resulting function. Figure 4-(b) presents a set of 3D curves extracted from the face surface

Methods	MSE	NN	1-Tier	2-Tier	e-Measure	DCG
Original distance	-	**0.9731**	**0.7019**	0.7755	0.5652	**0.8933**
Approximated distance 50	0.0035	0.9685	0.7016	0.7713	0.5633	0.8897
Approximated distance 20	0.0042	0.9685	0.7013	**0.7781**	**0.5662**	0.8922

Table 1: Comparison of the original and approximated distance-measure performances on the dataset[13]. The first row shows the performance of the original distance, the second row presents the performance of the approximated distance when keeping 50 dimensions, and the third row presents the performance of the approximated distance using only 20 dimensions.



(a) 60 dimensions (b) 50 dimensions (c) 40 dimensions

(d) 20 dimensions (e) 15 dimensions (f) Original similarities

Figure 3: Similarities matrix using approximated distances of 100 curves selected from the testdata (see Figure 2). Curves are represented by different dimensions vector.

Methods	MSE	NN	1-Tier	2-Tier	e-Measure	DCG
Original distance	-	0.9571	0.3739	0.4096	0.2431	0.7115
Approximated distance 50	0.0046	0.9498	0.3452	0.3976	0.2433	0.7028
Approximated distance 20	0.0089	0.8756	0.2985	0.3245	0.2097	0.6121

Table 2: Comparison of the original and approximated distance-measure performances on the dataset [17]. The first row shows the performance of the original distance, the second row presents the performance of the approximated distance when keeping 50 dimensions, and the third row presents the performance of the approximated distance using only 20 dimensions.

presented in Figure 4-(a). In order to test the performance of the approximated distance, we compare it with the original metric in an identification scenario.

The comparison between two faces using the original metric is performed based on Samir et al. method [10] which consists to compare facial curves of same level together and accumulate their distances over all levels. Using the approximated distance, we just concatenate signature of the curves starting from the closest to the noise tip, and used a dot product to compute a full similarity between faces. The number of level curves taken in both experiments is fixed to ten.

Figure 4(c) presents the performance of our method with respect the size of the curve signature used. Knowing that the original elastic metric gives a recognition rate of 92% on the dataset described above, the approximated distances proposed in this paper achieves a very similar performance: using a signature of 40 dimensions, we report 91.85% recognition rate.

3.4 Timing

The experiments presented in this paper were carried on a 2.60 GHz Intel(R) Core(TM) $i5-3320M$ CPU with 5.88GB memory. Computing the original geodesic distance between two curves takes in average 0.997 seconds. For instance, in the first experiment, the computation of the similarity matrix of 1300 shapes takes approximately 234 hours (since the matrix is symmetric). Using the proposed approximation, the computation time is drastically reduced to approximately 36 hours. The time taken to compute the Gram matrix (respectively

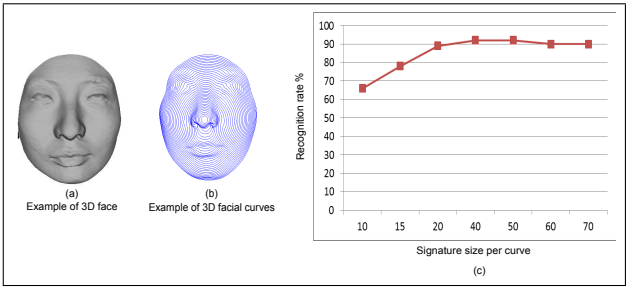


Figure 4: Rank one recognition rates versus size of the curve signature (the number of considered eigenvalues). The signature of the face is the concatenation of its curve signatures starting from the noise tip. In this experiment, we take only the ten first curves.

the distance matrix) with the projected curves is negligible since it involves a simple linear algebra computation with a low dimensional space.

More importantly, the computation time becomes linear with respect to the size of the collection since the number of curves used to compute the projections is fixed. Hence, doubling the size of the collection will double the time needed to compute the projections. The original distance however will take four times (the Gram matrix computation would remain almost instantaneous). Note that although we retain fewer projectors than the size of the training set (e.g., 20 or 50 eigenvectors for a training set of 100 shapes), the distance to all elements of the training set has to be computed, since the projectors are non-sparse linear combinations of all the elements of $\mathcal{A}$.

4 Conclusion

We proposed a fast approximation technique for curve distances computing. Using a kernel framework, we derived a kernel function from the elastic geodesic distance between curves. We proposed to learn a mapping to a low dimensional sub-space where the dot product is a good approximation of the derived kernel. We also proposed an iterative process to build the training set so as to enhance the approximation. The most important advantage is that the distance computing is reduced to a simple dot product on some projection space. This drastically reduces the running time of the computing process, since the non-linear computations have a complexity increasing linearly with the size of the collection (instead of quadratic with the original distance). Several applications can benefit from this work. This includes 3D face recognition, 3D biometrics and 3D shape retrieval and matching.

References

[1] Mohamed F. Abdelkader, Wael Abd-Almageed, Anuj Srivastava, and Rama Chellappa. Silhouette-based gesture and action recognition via modeling trajectories on riemannian shape manifolds. *Comput. Vis. Image Underst.*, 115(3), March 2011. ISSN 1077-3142.

[2] Petros Drineas and Michael W Mahoney. On the nyström method for approximating

a gram matrix for improved kernel-based learning. *The Journal of Machine Learning Research*, 6:2153–2175, 2005.

[3] S.H. Joshi, E. Klassen, A. Srivastava, and I. Jermyn. Removing shape-preserving transformations in square-root elastic (sre) framework for shape analysis of curves. In *EMMCVPR*, pages 387–398, 2007.

[4] E. Klassen and A. Srivastava. Geodesics between 3d closed curves using path-straightening. In *European Conference on Computer Vision (ECCV)*, pages 95–106, 2006.

[5] Sebastian Kurtek, Anuj Srivastava, Eric Klassen, and Hamid Laga. Landmark-guided elastic shape analysis of spherically-parameterized surfaces). *Computer Graphics Forum (Proceedings of Eurographics 2013)*, 32, 2013.

[6] Y. Laurent. Computable elastic distance between shapes. *SIAM Journal of Applied Mathematics*, 58(2):565–586, 1998.

[7] Ahmed Maalej, Boulbaba Ben Amor, Mohamed Daoudi, Anuj Srivastava, and Stefano Berretti. Shape analysis of local facial patches for 3d facial expression recognition. *Pattern Recogn.*, 44(8), 2011. ISSN 0031-3203.

[8] P. W. Michor and D. Mumford. Riemannian geometries on spaces of plane curves. *J. Eur. Math. Soc.*, 8:1–48, 2006.

[9] P. Jonathon Phillips, Patrick J. Flynn, Todd Scruggs, Kevin W. Bowyer, Jin Chang, Kevin Hoffman, Joe Marques, Jaesik Min, and William Worek. Overview of the face recognition grand challenge. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, Washington, DC, USA, 2005. IEEE Computer Society.

[10] Chafik Samir, Anuj Srivastava, Mohamed Daoudi, and Eric Klassen. An intrinsic framework for analysis of facial surfaces. *Int. J. Comput. Vision*, 82(1), 2009. ISSN 0920-5691.

[11] B Schölkopf. The kernel trick for distances. TR MSR 2000-51, Microsoft Research, Redmond, WA, 2000. Advances in Neural Information Processing Systems, 2001.

[12] Bernhard Scholkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *Advances in kernek methods - Support vector learning*, pages 327–352. MIT Press, 1999.

[13] Anuj Srivastava, Eric Klassen, Shantanu H. Joshi, and Ian H. Jermyn. Shape analysis of elastic curves in euclidean spaces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(7), 2011. ISSN 0162-8828.

[14] Hedi Tabia, Mohamed Daoudi, Jean-Philippe Vandeborre, and Olivier Colot. A new 3d-matching method of nonrigid and partially similar models using curve analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(4), 2011. ISSN 0162-8828.

[15] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.

[16] Chris Williams and Matthias Seeger. Using the nystroem method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press, 2001.

[17] M. Zuliani, S. Bhagavathy, B. S. Manjunath, and C. S. Kenney. Affine-invariant curve matching. In *IEEE International Conference on Image Processing*, Oct 2004.