A Fast Algorithm for Elastic Shape Distances between Closed Planar Curves

Günay Doğan[1,2], Javier Bernal[2], Charles R. Hagwood[2]
[1]Theiss Research. [2]National Institute of Standards and Technology, USA.

A major challenge of modern data sets is that their elements usually represent complex geometric structures and objects, e.g., proteins, cells, mechanical parts, facial surfaces, and other morphologies. Typically, one wants to cluster, classify or compare elements of such data sets, but performing such analyses requires defining a proper topological space where these data entities reside. In the particular case where the elements of the data set represent shapes of objects, shape spaces offer the appropriate framework. Accordingly, one then needs the notion of shape distance to quantify dissimilarity of such entities. In this paper, we focus on the elastic shape distance of Srivastava et al. [2] for closed planar curves. This provides a flexible and intuitive geodesic distance measure between curve shapes, invariant to translation, scaling, rotation and reparametrization. Computing this distance, however, is computationally expensive. The original algorithm proposed in [2] using dynamic programming (*DP*) runs in $O(N^3)$ time, N the number of nodes per curve. In this paper, we propose a new fast iterative algorithm to compute the elastic geodesic distance between shapes of closed planar curves. The asymptotic time complexity of our algorithm is roughly $O(N^2)$. However, in our experiments, we have observed a linear trend with running times depending on the type of curve data.

Mathematically, the shape distance computation is formulated as a global minimization over triplets that consist of possible starting points (on the curve), rotations and reparametrizations. To be specific, given planar closed curves β_1 and β_2 of unit length in class C^2, and their shape functions $q_i(t) = \dot{\beta}_i(t)/\sqrt{\|\dot{\beta}_i(t)\|}$, $i = 1, 2$, respectively, the shape distance between them corresponds to the minimum over triplets (t_0, θ, γ) of the following energy

$$E(t_0,\theta,\gamma) \equiv \int_0^1 \|q_1(t) - \sqrt{\dot{\gamma}(t)}R(\theta)q_2(t_0+\gamma(t))\|^2 dt, \qquad (1)$$

where t_0 is a starting point, θ, $R(\theta)$ are a rotation angle and matrix, and γ is a diffeomorphism of $[0,1]$ into $[0,1]$ (with $\gamma(0) = 0$, $\gamma(1) = 1$, $\dot{\gamma}(0) = \dot{\gamma}(1)$).

We propose to optimize (1) using an alternating approach: We fix γ and optimize (1) with respect to t_0, θ (computing $\theta = \theta(t_0, \gamma)$ with the Kabsch algorithm [1] where required). With optimal t_0, θ fixed in (1), we then optimize E with respect to γ. We alternate between these optimizations until convergence. This is summarized in Algorithm 1. The optimization with respect to t_0 is done in $O(N^2)$ time by looping through $t_i = (i-1)/(N-1)$, $i = 1,\ldots,N$, and evaluating (1) for each t_0. The optimization with respect to γ for fixed t_0, θ is an $O(kN)$ iterative nonlinear constrained optimization problem initialized in $O(\varepsilon N^2)$ time with a fast *DP* algorithm, where k is the number of iterations and ε is a small constant. We elaborate on details of γ optimization below. Thus the overall computational cost of our algorithm is $O(\varepsilon N^2 + K(N^2 + kN))$, K the number of iterations of alternating approach.

Algorithm 1 The main optimization algorithm

Initialize $\gamma(t) = t$, $t_i = (i-1)/(N-1)$, $i = 1,\ldots,N$.
repeat
  Fix current γ in (1) and loop over all t_i to find optimal t_0 and $\theta = \theta(t_0, \gamma)$ with Kabsch algorithm.
  Fix current t_0, θ in (1).
  If 1^{st} iteration, then compute new γ with fast *DP*.
  Optimize E in (1) w.r.t. γ with iterative nonlinear constrained optimiz. algo. initialized with current γ.
until Energy change $< tol = 10^{-6}$ or iteration # > 50.

A crucial step in Algorithm 1 for the optimization of the energy (1) is the computation of the optimal diffeomorphism γ when we fix t_0, θ in (1) as part of our alternating optimization approach. At that point, we are dealing with an energy depending on a single function variable. When discretized, the global minimum of this single variable (γ) energy can be obtained using a *DP* algorithm [2]. The drawback of the original *DP* algorithm (*original-DP*) used for this problem is that it has $O(N^2)$ time complexity [2]. This is expensive for curves with many nodes, especially if we need to repeat this computation for many t_0 candidates. An efficient alternative is to use an iterative algorithm, which can have $O(N)$ cost per iteration, but usually converges to a local minimum of the energy. We combine the strengths of the two approaches: we first use a fast approximate *DP* algorithm (*fast-DP*). Our *fast-DP* algorithm works on a reduced search space and produces a rough approximate global minimum very fast. It still has quadratic time complexity, albeit with a very small constant. Then we use this approximation as the initial iterate for an efficient iterative nonlinear constrained optimization algorithm (*iter*), which takes it to the precise global minimum very fast in a small number of iterations, each of which has $O(N)$ cost.

	N=64	*128*	*256*	*512*	*1024*	*2048*	*4096*
original-DP	.02 sec	.086	.37	1.4	5.8	23	94
fast-DP+iter	.11	.16	.22	.45	.90	2.3	7.4

Table 1: **Timings of diffemorphism computations for increasing N.** Reparametrized $\beta_2(t)$ (cell boundary) with test γ, used shape functions $q_2(t)$ of original curve $\beta_2(t)$, $q_1(t)$ of reparametrized curve $\beta_1(t)$ to recover γ.

	$N = 64$	128	256	512	1024
Original Algorithm	1.4 sec	11	92	735	5917
Algorithm 1	4	11	19	50	89

Table 2: **Timings of distance computations for increasing N.** Computed theoretically zero distance between shapes of two versions of limaçon.

Finally, we present some of our experimental results that demonstrate the efficiency gains of our new algorithm compared to the original shape distance algorithm in [2] (see Tables 1, 2, 3, and the full paper for all results). We test our algorithm on synthetic curves, cell boundary curves, and subsets of Leaf and MPEG7 shape data sets. Our first test is to identify the correct diffeomorphism γ. Our *fast-DP+iter* algorithm accomplishes this an order of magnitude faster than *original-DP* for large curves (see Table 1). The next test is about actual shape distance computations. Table 2 shows the timings for a limaçon curve, sampled at an increasing number of nodes for the purpose of investigating scalability. We change the starting point of the curve, rotate the curve, and apply a synthetic reparameterization to it to obtain a second version of it, thus of the same shape and zero distance from it. The computational cost of the original algorithm becomes very expensive as N increases beyond 256, whereas Algorithm 1 yields reasonable computation times for all N. Finally, we compute the pairwise distance matrix of all the shape data sets (with $N = 256$). Our algorithm is an order of magnitude faster (see Table 3). It is even faster for finer samplings, e.g.,$N = 512, 1024$.

[1] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics*, 32(5):922–923, 1976.

[2] A. Srivastava, E. Klassen, S.H. Joshi, and I.H. Jermyn. Shape analysis of elastic curves in Euclidean spaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(7):1415–1428, 2011.

	Synthetic	Cells	Leaves	MPEG7
Matrix Size	$6^2 = 36$	$10^2 = 100$	$75^2 = 5625$	$100^2 = 10000$
Original Algo.	1 hr	2.5 hrs	129 hrs	240 hrs
Algorithm 1	12 min	4.5 min	12.5 hrs	38.5 hrs

Table 3: **Timings for distance matrices.** Distance matrices obtained by computing shape distances for all curve pairs (at $N = 256$) in each data set.

This is an extended abstract. The full paper is available at the Computer Vision Foundation webpage.