Fast Dynamic Programming for Elastic Registration of Curves

Javier Bernal\textsuperscript{1} Günay Doğan\textsuperscript{1,2} Charles R. Hagwood\textsuperscript{1}  
\textsuperscript{1} National Institute of Standards and Technology, \textsuperscript{2} Theiss Research  
\{javier.bernal,gunay.dogan,charles.hagwood\}@nist.gov

Abstract

Curve registration problems in data analysis and computer vision can often be reduced to the problem of matching two functions defined on an interval. Dynamic Programming (DP) is an effective approach to solve this problem. In this paper, we propose a DP algorithm that runs in $O(N)$ time to compute optimal diffeomorphisms for elastic registration of curves with $N$ nodes. This algorithm contrasts favorably with other DP algorithms used for this problem: the commonly used algorithm of quadratic time complexity, and the algorithm that guarantees a globally optimal solution with $O(N^4)$ time complexity. Key to our computational efficiency is the savings achieved by reducing our search space, focusing on thin strips around graphs of estimates of optimal diffeomorphism. Estimates and strips are obtained with a multigrid approach: an optimal diffeomorphism obtained from a lower resolution grid using DP is progressively projected to ones of higher resolution until full resolution is attained. Additionally, our DP algorithm is designed so that it can handle nonuniformly discretized curves. This enables us to realize further savings in computations, since in the case of complicated curves requiring large numbers of nodes for a high-fidelity representation, we can distribute curve nodes adaptively, focusing nodes in parts of high variation. We demonstrate effectiveness of our DP algorithm on several registration problems in elastic shape analysis, and functional data analysis.

1. Introduction

Curve registration problems in data analysis and computer vision, e.g., horizontal alignment of chromatograms by domain warping, computation of elastic shape distances, can usually be reduced to the problem of matching two functions defined on an interval $I$ in the real line. The problem of matching in turn usually involves computing an orientation-preserving diffeomorphism on the interval $I$ to match each point in the range of one function with a point in the range of the other function, and vice versa. This is done by optimizing with respect to diffeomorphism, a data mismatch energy defined by data associated with the two functions. Dynamic Programming (DP) is widely recognized as an effective approach to solve such problems. However, although it computes globally optimal solutions, it is computationally expensive.

In the context of shape analysis, Srivastava et al. \cite{1, 6, 9} proposed an algorithm for computing elastic shape distance between two closed curves in the plane using an $O(N^2)$ DP component for elastic registration of the curves, $N$ the number of nodes per curve. It is computationally expensive as its total complexity is $O(N^3)$. A faster DP algorithm was proposed in \cite{3} that works in a reduced search space (still $O(N^2)$ with a small constant), but computes very good diffeomorphisms. Note that a DP algorithm that would actually guarantee a globally optimal diffeomorphism by conducting a complete search would run in $O(N^4)$ time.

In what follows, we build on the works \cite{5, 8}, and describe computation in linear time of approximately optimal diffeomorphisms for elastic registration of curves. The diffeomorphisms are not guaranteed to be globally optimal, but we observed very convincing results in our experiments. Our DP approach uses concepts in \cite{8}, and similarly restricts its search to thin strips around graphs of estimates of optimal solution. It essentially uses a multigrid approach that projects, using DP, a diffeomorphism at a low resolution grid to one of higher resolution with this process continued recursively until a diffeomorphism of full resolution is obtained. This results in a fast $O(N)$ DP algorithm. We note, furthermore, that we implemented our DP to allow for curves of possibly unequal and nonuniform discretized domains of definition. In particular, our algorithm can be used for computing more efficiently elastic shape distances between closed curves in the plane with algorithms in \cite{1, 9, 2, 3} by replacing their DP components with ours. We present numerical results showing that our algorithm is much faster than the aforementioned DP components, and that with it, in particular, shape distance computation in \cite{9} is indeed much faster while still producing distances as good as before. Finally, we present numerical results from using our algorithm for alignment of chromatograms in context of elastic functional data analysis.
2. Elastic Registration Formulation

For \( F : [0, 1] \times [0, 1] \times \mathbb{R} \rightarrow \mathbb{R} \), we minimize energies of the following general form with respect to \( \gamma \), \( \beta \) a diffeomorphism of \([0, 1]\) onto itself with \( \gamma(0) = 0, \gamma(1) = 1, \dot{\gamma} > 0 \):

\[
E(\gamma) = \int_0^1 F(t, \gamma(t), \dot{\gamma}(t))dt. \tag{1}
\]

Many problems we find in applications of computer vision and scientific data analysis fall in the category of nonlinear data fitting, in which a target data function \( y : \mathbb{R} \rightarrow \mathbb{R}^d \) is given, and the problem is then that of fitting or registering a model \( f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^d \) to this data. This problem is often solved by optimizing (1) above with respect to \( \gamma \) for \( F(t, \gamma(t), \dot{\gamma}(t)) = \| y(t) - f(t, \gamma(t), \dot{\gamma}(t)) \|_p, p \geq 1 \), a nonlinear regression problem. The elastic curve registration problems that we address in this paper fall in this category.

In practice, we need to solve a discretized version of the problem, either because the data itself is discrete, or due to the need to approximate the functions numerically. Thus, given a positive integer \( N \), we choose a partition \( \{ t_i \}_{i=1}^N \) of \([0, 1]\), \( t_0 = 0 < t_1 < \ldots < t_N = 1 \), and discretize (1) with the trapezoidal rule:

\[
E(\gamma) = \frac{1}{2} \sum_{i=1}^{N-1} h_i(F(t_{i+1}, \gamma_{i+1}, \dot{\gamma}_{i+1}) + F(t_i, \gamma_i, \dot{\gamma}_i)), \tag{2}
\]

where \( \gamma_1 = 0, \gamma_N = 1, h_i = t_{i+1} - t_i, \gamma_l = \gamma(t_i), \dot{\gamma}_l = (\gamma_{i+1} - \gamma_i)/h_i \), for \( l = 1, \ldots, N - 1 \). We also add \( \gamma_N = \gamma_1 \) as a boundary condition for the derivative.

An important question then is the choice of discretization points \( \{ t_i \}_{i=1}^N \). This impacts both the efficiency and the accuracy of the solution. We propose an adaptive nonuniform discretization scheme that follows the complexity of the input data. In the case of a curve \( \beta(t) : [0, 1] \rightarrow \mathbb{R} \) of curvature \( \kappa(t) \), we can sample the curve to obtain nodes \( \{ \beta_i \} \), compute its discretized curvature \( \{ \kappa_i \} \) to measure geometric complexity, and choose discretization points accordingly. This motivates a two-step procedure:

1. **Distribute sample nodes \( \{ \beta_i \} \) such that pointwise geometric discretization error is below an acceptable tolerance: \( \| \beta_i - \beta_{i-1} \|_2 \leq \max(\kappa_i, \kappa_{i-1}) < 0.002 \).**

2. **Compute lengths between consecutive nodes \( \{ \beta_i \} \), define arclength parameterization summing up lengths, and make discretization points associated with parameterization the choice of \( \{ t_i \} \).**

Results of this procedure are illustrated in Figure 1.

3. Dynamic Programming (DP)

For positive integers \( N, M \), not necessarily equal, and possibly nonuniform partitions of \([0, 1]\), \( \{ t_i \}_{i=1}^N, t_1 = 0 < t_2 < \ldots < t_N = 1, \{ z_j \}_{j=1}^M, z_1 = 0 < z_2 < \ldots < z_M = 1 \), we consider the \( N \times M \) grid on the unit square with grid points labeled \( (i, j) \), \( i, j \) integers, \( 1 \leq i \leq N, 1 \leq j \leq M \), each grid point \((i, j)\) coinciding with planar point \((t_i, z_j)\).

If the mesh of each partition, i.e., \( \max(t_{m+1} - t_m), 1 \leq m \leq N - 1 \), and \( \max(z_{m+1} - z_m), 1 \leq m \leq M - 1 \), is sufficiently small, then the set of diffeomorphisms \( \gamma \) of \([0, 1]\) onto itself with \( \gamma(0) = 0, \gamma(1) = 1, \dot{\gamma} > 0 \), can be approximated by the set of homeomorphisms of \([0, 1]\) onto itself whose graphs are piecewise linear paths from grid point \((1, 1)\) to grid point \((N, M)\) with grid points as vertices. We refer to the latter set as \( \Gamma \). Then \( \gamma \) in \( \Gamma \) is an approximate diffeomorphism of \([0, 1]\) onto itself and as such an energy conceptually faithful to (2) can be defined and computed for it. This is done one linear component of the graph of \( \gamma \) at a time.

Accordingly, given grid points \((k, l), (i, j), k < i, l < j\), that are endpoints of a linear component of the graph of \( \gamma \), an energy of a trapezoidal nature over the line segment joining \((k, l)\) and \((i, j)\) is defined as follows:

\[
E_{(k, j)}^{(i, j)} = \frac{1}{2} \sum_{m=k}^{i-1} (t_{m+1} - t_m)(F_{m+1} + F_m), \tag{3}
\]

\[
F_m = F(t_m, \alpha(t_m), L), \quad m = k, \ldots, i.
\]

Here \( \alpha \) is the linear function from \([t_k, t_i]\) onto \([z_k, z_j]\) whose graph is the line segment, \( \alpha(t_k) = z_k, \alpha(t_i) = z_j \), and \( L \) is the slope of the line segment. Note \( L = \frac{z_j - z_k}{t_i - t_k} > 0 \) as \( z_j > z_k, t_i > t_k \). The energy for \( \gamma \) is then defined as the sum of the energies over the linear components of the graph of \( \gamma \) with \( \alpha \) in (3) coinciding with \( \gamma \) on each component.
For the purpose of efficiently computing $\gamma^*$ in $\Gamma$ of minimum energy, we present algorithm in next section that uses DP on grid points in strips around graphs of estimates of $\gamma^*$, one strip at a time. In this algorithm, a general DP procedure, Procedure DP, whose outline follows, is executed, for each strip, on set $R$ of grid points inside strip. For such sets computational cost is low (search space is relatively small), and their selection is such that it is highly likely final DP solution is $\gamma^*$ itself or at least close to it. Since the collection of such strips has the appearance of one single strip whose shape evolves as it mimics the shapes of graphs of estimates of $\gamma^*$, we think of the collection as indeed being one single strip, a dynamic strip that we call adapting strip accordingly. In [2], Dogan, Bernal and Hagwood proposed using a strip $R$ of linear ($O(N)$) width around the diagonal of $[0,1]^2$ connecting planar points $(0,0)$ and $(1,1)$, for a fast DP algorithm. In this work, rather than rigidly fixing $R$, we propose using an adapting strip as described above with a width that is constant ($O(1)$) as it evolves around graphs of estimates of $\gamma^*$. Obviously we do not know $\gamma^*$, but can estimate it using DP solutions on coarser grids. However, before going into the specifics of our proposed algorithm, we will describe Procedure DP operating on generic $R$.

The set $R$ of labeled grid points can be any subset of the interior grid points plus the corner grid points $(1,1)$, $(N,M)$. Given any such $R$, we denote by $\Gamma(R)$ the set of elements of $\Gamma$ with all vertices in $R$. Accordingly, with the energy in (3) adjusted for $R$ (see below), given positive integer layrs (e.g., layrs = 5) which determines the size of certain neighborhoods to be searched (see below), then, based on DP, Procedure DP that follows, in $O(|\Gamma(R)|)$ time, will often (depending on layrs) compute optimal $\gamma^*$ in $\Gamma(R)$, $|R|$ the cardinality of $R$.

As the DP algorithm progresses over the indices $(i,j)$ in $R$, it examines function values on indices $(k,l)$ in a trailing neighborhood $N(i,j)$ of $(i,j)$ (see Figure 2 for a particular $R$ described below). In the full DP, we would be examining all $(k,l)$ in $R$, $1 \leq k < i, 1 \leq l < j$. This has high computational cost, and is not necessary for our applications. Using a much smaller square neighborhood $N(i,j)$ of $\omega$ points ($\omega = \text{layrs}$) per side gives satisfactory results. Thus, for each $(i,j)$ in $R$, we examine at most $\omega^2$ indices $(k,l)$ in the trailing neighborhood $N(i,j)$. Then the overall time complexity is $O(\omega^2|R|)$. We formally define $N(i,j)$ by

$$N(i,j) = \{(k,l) \in R : k \text{ is one of } \omega \text{ largest indices } i \text{ and } l \text{ is one of } \omega \text{ largest indices } j \}.$$ 

Note that in the unusual case $N(i,j)$ happens to be empty then a grid point $(k,l)$ in $R$, $k < i, l < j$, perhaps $(k,l) = (1,1)$, is identified and $N(i,j)$ is set to $\{(k,l)\}$.

The DP procedure follows. First, however, we clarify some implicit conventions in the procedure logic. The main loop in the DP procedure takes place over the single index $i$ (not the grid point $(i,j)$). We process index $i$ in increasing order of its values, and for each value, each occurrence of the value is processed before moving to the next one. Also in the procedure, pairs of indices $m_1, m_2$ are retrieved from an index set $M$, satisfying $m_1 < m_2$ with no other index in $M$ greater than $m_1$ and less than $m_2$.

**procedure DP**

$$E(1,1) = 0$$

for each $(i,j) \neq (1,1)$ in $R$ in increasing order of $i$

for each $(k,l) \in N(i,j)$

$\alpha = \text{linear function, } \alpha(t_k) = z_1, \alpha(t_i) = z_j$

$L = \text{slope of line segment } (k,l)(i,j)$

$M = \{m : k \leq m \leq i, \exists (m,n) \in R\}$

$$F_m = \sum_{m \in M}(m(t_m - t_i))(F_{m_2} + F_{m_1})$$

end for

$$E(i,j) = \min_{(k,l) \in \{k,l \in N(i,j)\}}(E(k,l) + F_{(i,j)})$$

end for

**end procedure**

The optimal solution $\gamma^*$ in $\Gamma(R)$ can then be obtained by backtracking from $(N,M)$ to $(1,1)$ with pointer $P$ above. Accordingly, Procedure opt-diffeom that follows, will produce $\gamma^*$ in the form $\gamma^*_m = \gamma^*_m(N)_{m=1}^N = \gamma^*_m(N)_{m=1}^N$:

**procedure opt-diffeom**

$$\gamma^*_N = 1$$

$$(i,j) = (N,M)$$

while $(i,j) \neq (1,1)$

$$(k,l) = P(i,j)$$

$$\gamma^*_k = z_l$$

for each integer $m, k < m < i$

$$\gamma^*_m = (t_i - t_m)/(t_i - t_k)z_1 + (t_m - t_k)/(t_i - t_k)z_j$$

end

$$(i,j) = (k,l)$$

end while

**end procedure**

The original $O(N^2)$ DP algorithm, which we call original-DP, was used in [1, 6] to compute elastic shape distances. It is essentially the same as Procedure DP above (for the proper instance of (2)) followed by Procedure opt-diffeom, using a uniform grid, $N = M$, and $R$ equal to all interior grid points plus the corner grid points $(1,1)$, $(N,M)$. Depending on layrs, it computes optimal $\gamma^*$ in $\Gamma$. In [2], a cheaper but still $O(N^2)$ version of original-DP called fast-DP, based on the Sakoe-Chiba Band [7], was presented for the same purpose. It is essentially original-DP with $R$ equal to the corner grid points $(1,1)$, $(N,M)$ plus interior grid points inside a strip $S$ of width $d$ along the diagonal of unit square from $(0,0)$ to $(1,1)$ (see Figure 2). Depending on layrs and $d$, it computes optimal $\gamma^*$ in $\Gamma$. 

1068
4. Dynamic Programming Restricted to an Adapting Strip

The DP algorithms original-DP and fast-DP used in [1, 9, 2, 3] are essentially the same as Procedure DP (for the proper instance of (2)), using a uniform grid, \( N = M \), and particular sets of grid points for \( R \). Clearly, under these conditions, the possibility is then precluded of using either one of them for elastic registration of curves whose defining point sets have been either refined or coarsened due, for example, to curvature considerations.

In what follows, working with partitions (not necessarily uniform) of \([0,1]\), \( \{t_i\}_{i=1}^N \), \( \{z_i\}_{i=1}^M \), as previously described, we present a linear algorithm which we call adapt-DP, based on DP restricted to an adapting strip, to compute optimal diffeomorphisms for elastic registration of curves. It has parameters \( \text{layers}, \text{lstrip} \), set to small positive integers, say 5, 30, respectively. Parameter \( \text{layers} \) is as previously described, while \( \text{lstrip} \) is an additional parameter that determines width of adapting strip (see below). Although adapt-DP is not guaranteed to be always successful, it has been observed to produce convincing results in our experiments. The original ideas for this algorithm are described in [5, 8] in the context of graph bisection and dynamic time warping.

As presented below, for a given instance of (2), adapt-DP is essentially an iterative process that restricts its search to the adapting strip around graphs of estimated solutions. Each iteration culminates with execution of Procedure DP for recursively projecting a diffeomorphism obtained from a lower resolution grid to one of higher resolution until full resolution is attained. For simplicity, we assume here \( N = M = 2^n + 1 \) for some positive integer \( n \). Extension of the algorithm to allow \( N, M \) to have any values is straightforward. Note we don’t assume partitions \( \{t_i\}, \{z_i\} \) are uniform. Finally, after last execution of Procedure DP in \( \text{adapt-DP} \), Procedure opt-diffeom is performed to obtain, depending on \( \text{layers} \) and \( \text{lstrip} \), optimal \( \gamma^* \) in \( \Gamma \). Algorithm \( \text{adapt-DP} \) follows:

algorithm \( \text{adapt-DP} \)

1. \( I(1) = J(1) = 1 \)
2. \( P(N, M) = (1, 1) \)
3. \( \text{for } r = 1 \text{ to } n \) \text{do} \( P(r, r) = \text{adapt-DP}(r, r) \) \text{for } \( r = 1 \text{ to } n \) \text{end for} \( P(r, r) = \text{adapt-DP}(r, r) \) \text{for } \( r = 1 \text{ to } n \) \text{end for}

4. \( \text{while } (i,j) \neq (1,1) \text{ do} \)
5. \( (i,j) = (N,M) \)
6. \( D = \emptyset \)

7. \( I(m+1) = m \cdot 2^{n-r} + 1 \)
8. \( J(m+1) = m \cdot 2^{n-r} + 1 \)
9. \( \text{end while} \)

10. \( \text{for } (m', n') \text{ in D do} \)
11. \( i_0 = \text{max}\{2, m' - \text{lstrip}\} \)
12. \( j_0 = \text{max}\{2, n' - \text{lstrip}\} \)
13. \( R = \emptyset \)
14. \( R = R \cup R_1 \cup R_2 \)

15. \( \text{end for} \)

16. \( \text{execute procedure DP on } R \)

17. \( \text{end for} \)

18. \( \text{execute procedure opt-diffeom to obtain } \gamma^* \)

end algorithm

In outline of \( \text{adapt-DP} \) above, we note in line 5, \( NI \) starts equal to 3 (for \( r = 1 \)) and then it is essentially doubled at each iteration \( r > 1 \) until it becomes equal to \( N \) at the \( n^{th} \) iteration. We note in line 2 and in line 7 inside for loop at line 6, range of \( I \) starts with 3 integers (for \( r = 1 \))
and then essentially doubles in size at each iteration \( r > 1 \), contains previous range of \( I \) from preceding iteration, and is evenly spread in the set \( \{1, 2, \ldots, N\} \) until it becomes this set. We note as well from the well-known sum of a geometric series that since \( N = 2^n + 1 \) then the sum of the \( NI \)'s, i.e., \( (2^1 + 1) + (2^2 + 1) + \ldots + (2^n + 1) \), is \( O(N) \). Clearly, all of the above applies to \( NJ, M \), and range of \( J \).

We note while loop at line 18 identifies certain cells in the Voronoi diagram [11] of the set of grid points \( R'd' = \{(i, j) : i = I(m'), j = J(n'), 1 < m' < NI, 1 < n' < NJ\} \) restricted to the unit square. Indeed bin \( B(m', n') \) as defined in lines 20-22, in terms of the computations in lines 8, 12, 14, 15, is exactly the Voronoi cell of \((I(m'), J(n'))\), and all such cells together partition the unit square. Accordingly, with \( \gamma^* \) encoded in \( P \) in line 3 \((r = 1)\) or in line 38 \((r > 1)\) through the execution of Procedure DP in the previous iteration \((r - 1)\), it must be that every point in the graph of \( \gamma^* \) is in some bin \( B(m', n') \). Thus, it then seems reasonable to say that a reliable region of influence of \( \gamma^* \) is the region around the graph of \( \gamma^* \) formed by the union of bins within a constant number of bins from the graph. Accordingly, to be precise, a bin \( B \) is part of this region if and only if there is a bin \( B' \), the interior of which the graph of \( \gamma^* \) intersects, \( B \) within a constant number \((lstrp)\) of bins from \( B' \), \( B \) directly below or to the left of \( B' \), or \( B \) equal to \( B' \) (see Figure 3). We note that identifying this region is essentially accomplished in while loop at line 18 and for loop at line 31, with the region understood to be the union of bins or Voronoi cells \( B(m', n') \) of grid points in \( R \) at the end of for loop. Clearly, the region contains the graph of \( \gamma^* \), and has the appearance of a strip whose shape evolves from one iteration to the next as it closely mimics the shape of the graph of \( \gamma^* \) (see Figure 3), thus it is referred to as a adapting strip. Finally, we note that at the end of for loop, \( \gamma^* \) in \( \Gamma(R) \subseteq \Gamma(R') \) encoded in \( P \) for current iteration is obtained in line 38 Procedure DP restricted to the region or adapting strip, a region that as just described depends on all previous \( \gamma^* \) functions from previous iterations. The last \( \gamma^* \) obtained is then, depending on \textit{layrs}, optimal in \( \Gamma(R) \), and, depending on \textit{layrs} and \textit{lstrp}, in \( \Gamma(R') \).

With \( \gamma^* \) as above during the execution of while loop at line 18 for iteration \( r \), we note that since \( \gamma^* \) is in \( \Gamma(R) \) then the number of bins \( B(m', n') \) whose interiors the graph of \( \gamma^* \) intersects must be \( O(NI + NJ) \), which is also the time required to find them one linear component of the graph at a time. Since \( |R| \) at end of for loop at line 31 is then \( O(lstrp) \cdot O(NI + NJ) \), i.e., \( O(NI + NJ) \), complexity of Procedure DP at line 38 is then \( O(NI + NJ) \), and since as mentioned above the sum of the \( NI \)'s and \( NJ \)'s is \( O(N) \) and \( O(M) \), respectively, then the complexity of \( \textit{adapt-DP} \) must be \( O(N + M) \), implying \( \textit{adapt-DP} \) is linear.

5. Applications and Experiments

In this section, we illustrate the effectiveness of Algorithm \( \textit{adapt-DP} \) with several benchmarks and applications. We first compared it to \( \textit{original-DP} \) [1, 9] and \( \textit{fast-DP} \) [2] on synthetic applications. We examined both the computation times, and the accuracy of the solutions. Then we tested it on two important applications: elastic shape distances between 2d closed curves, and domain warping for alignment of functional data, specifically chromatograms. We report our findings in the respective subsections.

5.1. Synthetic Benchmarks

We evaluated \( \textit{adapt-DP} \) using five synthetic curves in Figure 4 and \( \gamma \) functions shown in Figure 5. The \( \gamma \) functions were chosen to be difficult, having either small or steep gradients or both. We compared the results with those from \( \textit{original-DP} \) and \( \textit{fast-DP} \). Given \( \gamma \) function, we reparametrized synthetic curve \( \beta_2 \) with it to obtain \( \beta_1 = \beta_2(\gamma) \), and then with each algorithm tried to recover the discrete solution \( \gamma \) from the shape functions \( q_1, q_2 \) of \( \beta_1, \beta_2 \), respectively (see Subsection 5.2 for definition of shape functions), using \( F(t, \gamma(t), \dot{\gamma}(t)) = \|q_1(t) - \sqrt{\gamma(t)}q_2(\gamma(t))\|^2 \) in (1). For various values of \( N \) (64, 128, 256, 512, 1024, 2048), and associated values of \textit{layrs} (64, 32, 16, 12, 12, respectively), with \textit{lstrp} = 20, we executed each algorithm, timed computations, and computed the \( L^2 \) error between the true solution \( \gamma^* \) and the computed \( \gamma \), i.e., \( \frac{1}{N-1}\sum_{i=1}^{N-1} (\gamma(t_i) - \gamma^*(t_i))^2 \). Algorithm \( \textit{adapt-DP} \) not only computed reasonable \( \gamma^* \) functions for all synthetic curves \( (L^2 \) errors were less than \( 10^{-3}) \), but was much faster than the other algorithms for \( N \geq 256 \) (see Figure 5 and Table 1).

5.2. Computation of Elastic Shape Distances

An elastic shape framework was introduced in [9] for finding geodesics in the shape space of closed curves and computing geodesic distances between elements of that space. Let closed curves \( \beta_i : [0, 1] \rightarrow \mathbb{R}^2, i = 1, 2 \) be of class \( C^2 \) and unit length. As each \( \beta_i \) is closed, it sat-
with DP, which is the most ext

\[ \hat{\gamma}(t) = \sqrt{\gamma(t)}q_2(\gamma(t)) \]

where \( t_0 \) is the optimal seed or starting point, \( R(\theta) \) is the \( 2 \times 2 \) rotation matrix defined by the optimal rotation angle \( \theta \), and \( \gamma \) is the optimal reparameterization function (see [3]). The triple \((t_0, \theta, \gamma)\) for optimal alignment is then obtained by minimizing the mismatch energy:

\[ E(t_0, \theta, \gamma) = \int_0^1 \|R(\theta)q_1(t + t_0) - \sqrt{\gamma(t)}q_2(\gamma(t))\|^2 dt. \]  

(4)

Note that, for fixed \( t_0, \theta \), (4) is in the same form as (1) for \( F(t, \gamma(t), \hat{\gamma}(t)) = \|R(\theta)q_1(t + t_0) - \sqrt{\gamma(t)}q_2(\gamma(t))\|^2 \). We use the trapezoidal rule to write a discretized version of the mismatch energy (4),

\[ E^h(t_0, \theta, \gamma) = h \sum_{l=1}^{N-1} \|R(\theta)q_1(t_l + t_0) - \sqrt{\gamma}q_2(\gamma(t_l))\|^2, \]

(5)

essentially adapting (2) to this particular case.

In practice, the curves \( \beta \) are available as discrete sets of curve nodes. In order to obtain the continuous SRVFs \( q_i \), needed in (5), we first compute discrete derivatives \( \beta_i \) with centered finite differences, and then compute the corresponding \( q_i \), which we interpolate with cubic splines.

The minimization of (5) to obtain the optimal triple \((t_0, \theta, \gamma)\) is the most critical part of the shape distance computation. Although a globally optimal triple is required to compute the correct theoretical distance, a practical optimization algorithm to accomplish this goal is not available.

Instead, various local optimization approaches have been proposed. The approach in [9] is to loop through the starting point \( t_0 \) candidates, to compute for each \( t_0 \) candidate the optimal rotation angle \( \theta \) (assuming identity for initial \( \hat{\gamma} \)), and then to compute the optimal reparameterization \( \gamma \) for each fixed pair \((t_0, \theta)\) with DP, which is the most expensive step as it is \( O(N^2) \) for each pair. This optimization scheme is a direct search algorithm with total time complexity of \( O(N^3) \). Faster iterative algorithms were proposed in [4] and [2, 3]. In [4], Huang et al. used Riemannian optimization to achieve faster computation times and improved minimization results as compared to the direct search approach in [9]. In [2, 3], Dogan et al. proposed an alternating optimization algorithm that optimizes \((t_0, \theta)\) with FFT, and \( \gamma \) with an iterative solver based on constrained nonlinear optimization using the interior point method and initialized with fast-DP. They were able to demonstrate subquadratic running times in experiments. In this paper, we would like to demonstrate the efficiency gains from our new DP algorithm when used to compute elastic shape distances. For this purpose, we adopted the \( O(N^3) \) algorithm in [9], and replaced its \( O(N^2) \) original-DP step with our \( O(N) \) adapt-DP. We were able to show improvement by an order of magnitude in computation times, while still computing shape distances as good as the original algorithm.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|c|}
\hline
\( N \) & 64 & 128 & 256 & 512 & 1024 & 2048 \\
\hline
\hline
\( \text{layers} = \) & 64 & 32 & 16 & 12 & 12 & 12 \\
\hline
\hline
\( o\text{-DP} \) & all & \gamma & 0.49 & 1.26 & 1.00 & 2.07 & 8.48 & 34.3 \\
\hline
\hline
\( f\text{-DP} \) & all & \gamma & 0.24 & 0.66 & 0.51 & 1.04 & 4.20 & 17.0 \\
\hline
\hline
\( a\text{-DP} \) & \( \gamma^{\text{flat}} \) & \( \gamma^{\text{steep}} \) & \( \gamma^{\text{bumpy}} \) & 0.65 & 0.67 & 0.27 & 0.29 & 0.57 & 1.20 \\
\hline
\hline
\end{tabular}
\caption{Times (in seconds) for limaçon with original-DP \((o\text{-DP})\), fast-DP \((f\text{-DP})\) with \( d \approx 0.3\sqrt{2} \), and adapt-DP \((a\text{-DP})\). For \( o\text{-DP} \) and \( f\text{-DP} \), the times depend only on \( N \), not on \( \gamma \), whereas for \( a\text{-DP} \), the times depend on the shape of \( \gamma \) as well.}
\end{table}

isfies \( \beta_i(0) = \beta_i(1), \hat{\beta}_i(0) = \hat{\beta}_i(1) \). We define \( q_i(t) = \beta_i(t)/\|\beta_i(t)\|^{1/2} \) to be the shape function or square-root velocity function (SRVF) of \( \beta_i \). Then the elastic shape distance between \( \beta_1 \) and \( \beta_2 \) is defined as the \( L^2 \) angle between the optimally aligned SRVFs \( \hat{q}_1, \hat{q}_2 \).
In order to examine scalability with respect to $N$, we computed with algorithm in [9] the shape distance between two synthetic curves, hippopede and bumps (see Figure 4), using original-DP and adapt-DP. Results are given in Table 2. The shape distances from both approaches agree very well. But we see that computation times with original-DP grow cubically, whereas with adapt-DP grow quadratically. For $N = 256, 512$ we then computed $10 \times 10$ pairwise shape distance matrices for cell boundary curves in Figure 4 as well, and observed the same efficiency gains. Algorithm in [9] with adapt-DP had total computation time that is a fraction of what it had with original-DP: 17 min, 1 hr by adapt-DP vs 50 min, 7 hrs by original-DP for $N = 256, 512$ respectively. A $5 \times 5$ submatrix of the distance matrix for $N = 512$ is shown in Table 3.

Additionally, we verified that adapt-DP indeed handles nonuniform discretizations correctly and produces results as good as the uniform case. For this, we took two cell boundary curves in Figure 4, $\beta_1$, the rightmost curve in middle row, and $\beta_2$, the rightmost curve in bottom row. We resampled them uniformly with equal numbers of nodes $N_1 = N_2 = 257$, and with $F(t, \gamma(t), \dot{\gamma}(t)) = \|q_1(t) - \sqrt{\gamma(t)}q_2(\gamma(t))\|^2$ in (1), computed $\dot{\gamma}$ with adapt-DP for optimal matching, and $E(\gamma)$, the value of mismatch energy for $\dot{\gamma}$. We repeated this experiment for the nonuniform discretization of these curves obtained with the two-step procedure in Section 2. Optimal energy values and computation times are given below. Numerical results in the nonuniform case are as good as those of the uniform case at half the cost of computation time.

<table>
<thead>
<tr>
<th></th>
<th>$N_1$</th>
<th>$N_2$</th>
<th>$E(\gamma)$</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>257</td>
<td>257</td>
<td>0.3786</td>
<td>0.291s</td>
</tr>
<tr>
<td>Nonuniform</td>
<td>163</td>
<td>149</td>
<td>0.3628</td>
<td>0.138s</td>
</tr>
</tbody>
</table>

### 5.3. Function Alignment by Warping

In the context of elastic functional data analysis, a framework was introduced in [10] for domain warping of functions in order to align them optimally by matching critical features, such as peaks.

For $i = 1, 2$, let $f_i : [0, 1] \to \mathbb{R}$ be functions in an appropriate space of functions (e.g., the space of absolutely continuous functions), and let $q_i : [0, 1] \to \mathbb{R}$ be the square-root slope function (SRSF) of $f_i$: $q_i(t) = \text{sign}(\dot{f}_i(t))\sqrt{|\dot{f}_i(t)|}$, $t \in [0, 1]$ (see [10]). Note the SRSF is the form the SRVF takes as $f_i$ is real-valued.

As established in [10] the warping function $\gamma$ that makes $f_2(\gamma)$ the optimal alignment of $f_2$ to $f_1$ is obtained by minimizing the following energy with respect to $\gamma$:

$$E(\gamma) = \int_0^1 (q_1(t) - \sqrt{\gamma(t)}q_2(\gamma(t)))^2 dt,$$

where as before $\gamma$ is a diffeomorphism of $[0, 1]$ onto itself with $\gamma(0) = 0, \gamma(1) = 1, \dot{\gamma}(t) > 0$. Note that (6) is in the same form as (1) for $F(t, \gamma(t), \dot{\gamma}(t)) = (q_1(t) - \sqrt{\gamma(t)}q_2(\gamma(t)))^2$.

In [12] the alignment of chromatograms using the framework in [10] described above was demonstrated on liquid chromatography-mass spectrometry data for a chromatographically complex metabolomic reference sample. Computational results were presented in [12] from aligning two chromatograms in this manner. The chromatograms were taken in immediate succession under the conventional high performance liquid chromatography (HPLC) protocol described in [12]. As reported there, for chromatograms having 1,000 points, the aligning took 10 seconds on a desktop computer. In this work, we addressed much larger chromatograms (19,000+ points) for which original-DP was impractical. We aligned such chromatograms in a few minutes using adapt-DP with 1strip = 150 and lagps = 12 (see Figure 6). Timings for these experiments are given below:

<table>
<thead>
<tr>
<th></th>
<th>Chromatogram 1</th>
<th>Chromatogram 2</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pair 1</td>
<td>19,713 pts</td>
<td>19,759 pts</td>
<td>180s</td>
</tr>
<tr>
<td>Pair 2</td>
<td>19,759 pts</td>
<td>26,474 pts</td>
<td>270s</td>
</tr>
<tr>
<td>Pair 3</td>
<td>19,693 pts</td>
<td>19,763 pts</td>
<td>172s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Timings for $\text{Dist}$(hippopede, bumps)</th>
<th>Timings for $\text{Dist}$(bumps, hippopede)</th>
<th>Values for $\text{Dist}$(hippopede, bumps)</th>
<th>Values for $\text{Dist}$(bumps, hippopede)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$N = 64$</td>
<td>128</td>
<td>256</td>
<td>512</td>
</tr>
<tr>
<td>original-DP</td>
<td>0.50</td>
<td>3.53</td>
<td>28.7</td>
<td>227</td>
</tr>
<tr>
<td>adapt-DP</td>
<td>0.57</td>
<td>2.35</td>
<td>9.7</td>
<td>40</td>
</tr>
</tbody>
</table>

Table 2. The numerical values and running times (in seconds) for the elastic shape distance between the two synthetic curves: hippopede and bumps for increasing $N$.

Table 3. Matrix of pairwise shape distances of type A (rows) and type B (columns) cells. The first and second values of a pair computed using original-DP and adapt-DP, respectively.
We note that as currently implemented, \( \text{adapt-DP} \) uses only \( F(t, \gamma(t), \dot{\gamma}(t)) = \|q_1(t) - \sqrt{\gamma(t)} q_2(\gamma(t))\|^2 \) in (1), \( q_1, q_2 : [0, 1] \rightarrow \mathbb{R}^d, d = 1 \) or 2.

6. Conclusions

In this paper, we propose a fast linear Dynamic Programming (DP) algorithm to compute optimal diffeomorphisms for elastic registration of curves. Although we cannot guarantee that it will always compute a globally optimal solution, we have observed very convincing results in our experiments. This algorithm which we call \( \text{adapt-DP} \) is based on ideas in [5, 8] in the context of graph bisection and dynamic time warping. We achieve considerable savings in computations and very favorable run times by restricting its search to thin strips around graphs of estimated solutions. It is essentially an iterative process that starts with a diffeomorphism computed at a very low resolution grid, projects at each iteration current diffeomorphism to one of double resolution using DP, and ends when a diffeomorphism of full resolution is obtained. This process runs with linear asymptotic time complexity with respect to the number of nodes on the given curves. We note, furthermore, \( \text{adapt-DP} \) has been implemented to allow for curves of possibly unequal and nonuniform discretized domains of definition. We use this flexibility to our advantage, to achieve further savings in computations, by not working with uniformly discretized curves, but with nonuniformly discretized curves of fewer nodes, as we concentrate nodes on parts with high curvature, and not so much on flat parts. We demonstrate the efficiency of \( \text{adapt-DP} \) with several examples. We achieve an order of magnitude gain in speed when we perform elastic shape analysis proposed in [9] with \( \text{adapt-DP} \). We achieve even larger speed gains when we use \( \text{adapt-DP} \) for the alignment of chromatograms with large numbers of sample points. In particular, for chromatograms of 20,000 points, we show this can be done in approximately 3 minutes.

A copy of \( \text{adapt-DP} \) with example data files and usage instructions can be obtained from the link: http://math.nist.gov/~JBernal/Fast_Dynamic_Programming.zip

References


