Self-Similarity Based Time Warping

Christopher J. Tralie†

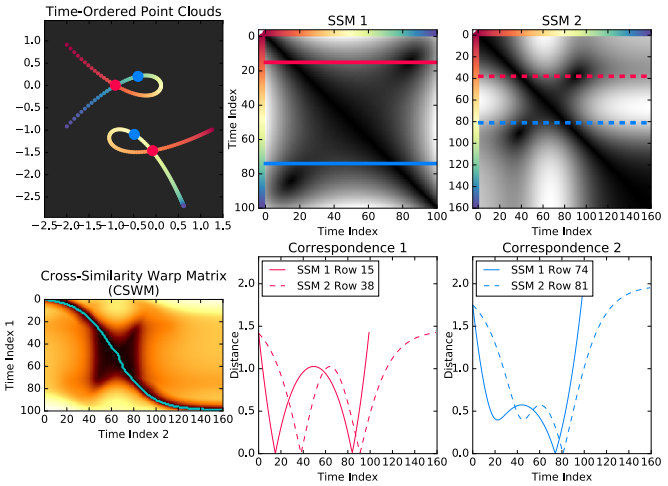


Fig. 1: A concept figure for our technique of aligning time-ordered point clouds which are rotated/translated/flipped and re-parameterized versions of each other. Rows of self-similarity matrices (SSMs) of points which are in correspondence are re-parameterized versions of each other, which reduces the global alignment problem to a series of 1D time warping problems. This observation forms the basis of our algorithm. The end result is a derived "warping path," drawn in cyan in the lower left plot of this figure, which informs how to synchronize the two point clouds in time.

***Abstract*—In this work, we explore the problem of aligning two time-ordered point clouds which are spatially transformed and re-parameterized versions of each other. Most other works that address this problem attempt to jointly uncover a spatial alignment and correspondences between the two point clouds, or to derive local invariants to spatial transformations such as curvature before computing correspondences. By contrast, we sidestep spatial alignment completely by using self-similarity matrices (SSMs) as a proxy to the time-ordered point clouds, since self-similarity matrices are *blind to isometries* and respect global geometry. Our algorithm, dubbed "Isometry Blind Dynamic Time Warping" (IBDTW), is simple and general, and we are able to apply it to cross modal time warping after carefully normalizing the SSMs. We also show that the dissimilarity measure associated with correspondences uncovered by the IBDTW algorithm lower bounds the Gromov-Hausdorff distance between the two point sets when restricted to warping paths. Finally, we present a local, partial alignment scheme based on the Smith Waterman algorithm which is also blind to isometries. This algorithm eliminates the need for tedious manual cropping of time series, which is ordinarily necessary for global alignment algorithms to function properly.**

***Index Terms*—Curve Alignment, Video Alignment, Self-Similarity, Time Warping, Dynamic Time Warping, Isometry Blind Time Warping, Cross-Modal Alignment**

† Department of Mathematics, Duke University,Durham, NC, USA. e-mail: ctralie@alumni.princeton.edu

I. Introduction

In this work, we address the problem of synchronizing sampled curves, which we refer to as "time-ordered point clouds" (TOPCs)[1]. The problem of synchronizing TOPCs which trace similar trajectories but which may be parameterized differently is usually approached with the Dynamic Time Warping (DTW) algorithm [37], [38]. Since sequential data can often be translated into a sequence of vectors in some feature space, this algorithm has found widespread use in applications such as spoken word synchronization [37], [38], gesture recognition [43], touch screen authentication [15], video contour shape sequence alignment [31], and general time series alignment [6], to name a few of the thousands of works that use it. The problem becomes substantially more difficult, however, when the point clouds undergo spatial transformations or dimensionality shifts in addition to re-parameterizations, which is more common *across* modalities. For instance, we might choose to summarize a sequence of 3D meshes representing a person's expression with a 3D shape histogram, while we might use a sequence 2D facial landmarks to summarize the same expression in a video (see Section V-C). There is no apparent correspondence between these spaces a priori. This problem even arises within modalities, such as aligning gestures from different people who reside in different spatial locations. Thus, when synchronizing sampled curves, it is important to address not only re-parameterizations, but also spatial transformations such as maps between spaces or rotations/translations/flips within the same space.

In our work, we avoid explicitly solving for spatial maps by using *self-similarity matrices* (SSMs). Figure 1 shows a sketch of the technique. Even if the curves have been rotated/translated/flipped and re-parameterized, rows of the SSMs which are in correspondence are re-parameterized versions of each other. With proper normalization (Section III-D), this technique can also address cross-modal alignment.

Our technique is incredibly simple both conceptually and in implementation, it is fully automatic, and it requires no supervision. We show favorable results on a number of benchmark datasets (Section V). We also show an extension of our basic technique to *partially* align time series across modalities (Section IV).

[1]There is mild overlap of this paper with Ch. 5 of Christopher Tralie's dissertation [45]

II. Background

A. Self-Similarity Matrices (SSMs)

We now review the main data structure we rely on in this work, the *self-similarity matrix*. Given a space curve parameterized by the unit interval $\gamma : [0,1] \to (\mathcal{M}, d)$, a *Self-Similarity Image* is a function $D : [0,1] \times [0,1] \to \mathbb{R}$ so that

$$D_\gamma(i,j) = d(\gamma(i), \gamma(j)) \tag{1}$$

If d is the $L2$ metric on $\mathbb{R}^d$, a common choice, then this is referred to as a *Euclidean Self-Similarity Image*. The self-similarity image completely summarizes the metric information of the curve, disregarding a fixed ambient space with a fixed position and orientation in that space. This means that self-similarity images are naturally *blind to isometries* of the underlying space curve; the self-similarity image does not change if the curve is rotated/translated/flipped. In practice, self-similarity images must be sampled and treated as discrete objects for time-ordered point clouds. That is, given a space curve $\gamma : [0,1] \to (\mathcal{M}, d)$, its corresponding self-similarity image D_γ and a set of time indices $t_1, t_2, ..., t_K$ corresponding to a time-ordered point cloud, so that $t_1 = 0$, $t_K = 1$, and $t_i < t_{i+1}$, the *Self-Similarity Matrix (SSM)* is an $N \times N$ matrix M_γ so that

$$M_\gamma[i,j] = D_\gamma(t_i, t_j) \tag{2}$$

SSMs have been used as a tool in pattern recognition across many domains, and we only review a few of them briefly here which use an explicitly geometric interpretation. SSMs have been applied to the problem of human activity recognition in video, where a video is treated as a time-ordered point cloud mapped into a feature space designed to summarize the frames of the video [24]. In this work, straightforward approaches using texture descriptors on SSMs to match the resulting curves have shown promising results for activity recognition. SSMs were also used as a tool to detect periodicity and symmetry in video motion [14]. In the music information retrieval community, SSMs have been used on audio summarized in feature spaces, starting with the pioneering work of [17] which used a descriptor related to curvature in these images to detect note boundaries, with much followup work for music structure understanding, segmentation, and cover song identification, including more recent work by [5], [27], [41] and [33]. In the dynamical systems community, Euclidean SSMs are used as a format for reconstructing trajectories up to isometry, from which recurrence statistics can be computed [34].

B. Warping Paths

The warping path is the basic primitive object we seek to synchronize two time-ordered point clouds. It is a discrete version of an orientation preserving homeomorphism of the unit interval used to re-parameterize curves. In layman's terms, it provides a way to step forward along both curves jointly in a continuous way without backtracking, so that they are optimally aligned over all steps.

More precisely, given two sets X and Y, a *correspondence* $\mathcal{C}$ between the two sets is such that $\mathcal{C} \subset X \times Y$ and $\forall x \in X \exists y \in y$ s.t. $(x,y) \in \mathcal{C}$ and $\forall y \in Y \exists x \in X$ s.t. $(x,y) \in \mathcal{C}$. In other words, a correspondence is a matching between two sets X and Y so that each element in X is matched to at least one element in Y, and each element of Y is matched to at least one element in X. Let X and Y be two sets whose elements are adorned with a time order: $X = \{x_1, x_2, ..., x_M\}$ and $Y = \{y_1, y_2, ..., y_N\}$. A *warping path*, between X and Y is a correspondence $\mathcal{W}$ which can be put into the sequence $\mathcal{W} = (c_1, c_2, ..., c_K)$ satisfying the following three properties

- *Monotonicity*: If $(x_i, y_j) \in \mathcal{W}$, then $(x_k, y_l) \notin \mathcal{W}$ for $k < i$, $l > j$
- *Boundary Conditions*: $(x_1, y_1), (x_M, y_N) \in \mathcal{W}$
- *Continuity*: $c_i - c_{i-1} \in \{(0,1), (1,0), (1,1)\}$

See [36] for more details.

C. Dynamic Time Warping

We now have the language necessary to define dynamic time warping, which is the basis for our algorithm. Suppose there are two time-ordered point clouds X and Y which both live in the same metric space $(\mathcal{M}, d)$. The *Dynamic Time Warping (DTW) Dissimilarity* [37], [38][2] between X and Y is

$$\text{DTW}(X,Y) = \min_{\mathcal{W} \in \Omega} \sum_{(i,j) \in \mathcal{W}} d(x_i, y_j)$$

where Ω is the set of all valid warping paths between X and Y. DTW satisfies the following *subsequence relation*

$$DTW_{ij} = \left\{ d(x_i, y_j) + \min \begin{matrix} DTW_{i-1,j-1} \\ DTW_{i-1,j} \\ DTW_{i,j-1} \end{matrix} \right\} \tag{3}$$

where DTW_{ij} is the DTW dissimilarity between $\{x_1, x_2, ..., x_i\}$ and $\{y_1, y_2, ..., y_j\}$. This makes it possible to solve DTW with a dynamic programming algorithm which takes $O(MN)$ time. Algorithm 1 shows the details. We sometimes refer to the matrix of all distances d_{ij} between X_i and Y_j as the "cross-similarity matrix" (CSM). In light of this, Algorithm 1 computes the cost of the shortest path from the upper left to the lower right of the CSM. The optimal warping path can be extracted by backtracing through the dynamic programming matrix D.

We also present a very simple modification of the algorithm that enforces constraints in the warping path, which will be useful later in our algorithms. The idea is to run the original DTW algorithm twice, once for point clouds $X_{1:i}, Y_{1:j}$ and once on $X_{i:M}$ and $Y_{j:N}$, and to add the costs. This is again exploiting the fact that the DTW algorithm must start on the first point and end on the last point, so we split it into two optimal sub-paths, one which ends on X_i, Y_j and one which starts on X_i, Y_j. Figure 2 shows an example.

[2]Note that DTW is not a metric, as it fails to satisfy the triangle inequality. For an example, see [36] section 4.1

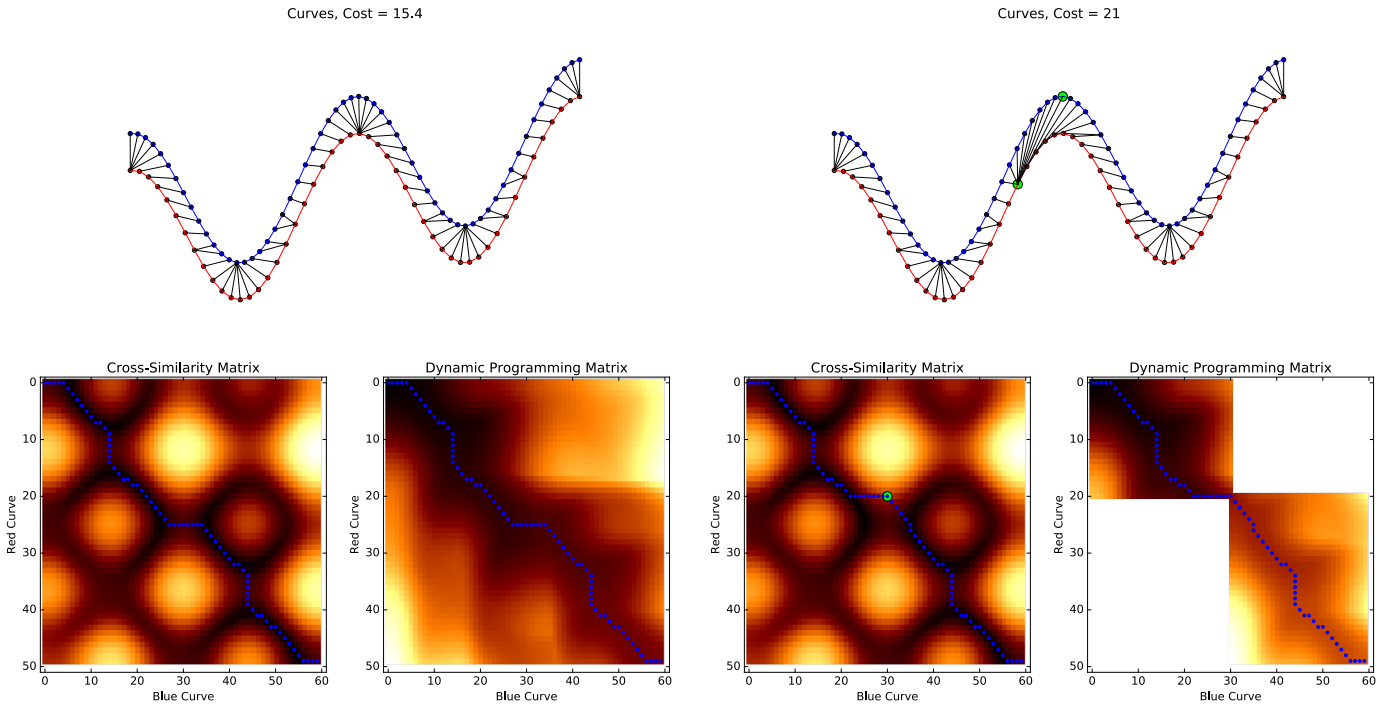


Fig. 2: An example of unconstrained DTW (left) compared to constrained DTW (right). The two points which are designated to be in correspondence are highlighted with a green dot in the figure on the right. This forces a slightly sub-optimal path with respect to the global alignment, but it is the optimal path with respect to the constraint.

Algorithm 1 Dynamic Time Warping Algorithm

procedure DTW(X, Y, d)
▷ TOPCs X and Y with M and N points, metric d.
▷ D array below holds a zero-indexed 2D dynamic programming matrix

$$\text{D} \leftarrow \underbrace{\left.\begin{bmatrix} 0 & \infty & \infty & \dots & \infty \\ \infty & 0 & 0 & \dots & 0 \\ \infty & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \infty & 0 & 0 & \dots & 0 \end{bmatrix}\right\}_{M+1}}_{N+1}$$

for $i = 1 : M$ **do**
  for $j = 1 : N$ **do**
    $$D[i,j] \leftarrow d(X_i, Y_i) + \min \left\{ \begin{matrix} D[i-1, j-1] \\ D[i-1, j] \\ D[i, j-1] \end{matrix} \right\}$$
  end for
end for
return D[M, N] ▷ Return cost of optimal warping path.
end procedure

Algorithm 2 Dynamic Time Warping with Constraints

procedure CONSTRAINEDDTW(X, Y, d, i, j)
▷ TOPCs X and Y with M and N points, compared with metric d. Return cost of optimal warping path with the constraint that x_i matches to y_j
$D1 \leftarrow$ DTW$([X_0, X_1, ...X_i], [Y_0, Y_1, ..., Y_j], d)$
$D2 \leftarrow$ DTW$([X_i, X_{i+1}, ...X_M], [Y_j, Y_{j+1}, ..., Y_N], d)$
▷ Now return the cost of the optimal constrained warping path, noting that $d(X_i, Y_j)$ was double counted
return $D1 + D2 - d(X_i, Y_j)$
end procedure

D. Cross-Modal Schemes for Dynamic Time Warping

In addition to synchronizing curves in the same ambient space which are approximately re-parameterizations of each other, there has also been some recent work attacking the more difficult problem of matching curves which live in different ambient spaces or which live in the same space but which may differ by a rigid rotation or translation, in addition to re-parameterization. The classes of techniques are as follows:

1) Procrustes-Based: A common approach to this problem is an "expectation/maximization" scheme that switches back and forth between a correspondence/time warping and a spatial alignment [53], [55]–[57]. One objective for spatial alignment could be to find the optimal *rigid transformation* taking one set of points to another. This leads to what's known as the *Procrustes distance* (or "Whaba's problem" [50]). Given two Euclidean point clouds $X, Y \in \mathbb{R}^d$, each with N points which are assumed to be in correspondence, the *Procrustes distance* is defined as

$$d_P(X,Y) = \min_{R_x, R_y, t_x, t_y} \sum_{i=1}^{N} ||R_x(x_i - t_x) - R_y(y_i - t_y)||_2^2 \quad (4)$$

where R_x and R_y are $d \times d$ rotation matrices, and t_x and t_y are d-dimensional translational offsets

The Procrustes distance can be solved with the Kabsch algorithm [26]. The optimal translation simply aligns the points by their center of mass, so $t_x = \overline{X}$ and $t_y = \overline{Y}$. Let $\hat{Y} = Y - t_y$ and $\hat{X} = X - t_x$, and let $\hat{X}$ and $\hat{Y}$ be $d \times N$ matrices holding the coordinates of each point in the corresponding columns. Then the optimal rotations are found by taking a singular value decomposition.

$$\hat{X}\hat{Y}^T = USV^T \tag{5}$$

The optimal rotations are then $R_x = U^T$ and $R_y = V^T$.

One issue with Procrustes is that not only do X and Y have to have the same number of points, but the correspondences must be known a priori. Often in practice, neither of these assumptions are true. To deal with this, there has been a long line of research on an algorithm known as Iterative Closest Points (ICP) [7], [11], usually in the context of 3D shape registration. If a subset of correspondences is known, they are used. For the rest, the algorithm proceeds as follows

1) Center the point clouds on their centers of mass
2) Find the nearest neighbors in Y to all points in X
3) Solve the Procrustes alignment based on these correspondences (possibly with duplication if two points in X have the same nearest neighbor in Y)
4) Rotate the points so that they are in alignment, and repeat steps 2-4 until convergence

This algorithm works well in practice if there is a good initial guess of alignment, but it can converge to a poor local minimum if not. It also has high sensitivity to outliers and missing data, which occur frequently in applications of 3D scanning (for example), though there has been some very promising recent work addressing this with norms other than L_2 which deal with outliers better [8]. In spite of these problems, the authors of [53] use a modified version of ICP, replacing step 3 with DTW instead of nearest neighbors. This ensures that the time order will be respected, which is not guaranteed with nearest neighbors only[3].

As an alternative to ICP and Procrustes, it is possible to rasterize point clouds to a grid and exhaustively search over all rotations using a modified version of the FFT [28]. This works well and is efficient for reasonable grid sizes of low dimensions, but it suffers from the curse of dimensionality since the number of grid cells explodes in higher dimensions.

2) CCA-Based: There are also techniques which use canonical correlation analysis instead of Procrustes analysis to perform a spatial alignment in an EM framework. These are very similar in spirit to Procrustes-based techniques, except spatial transformations are not restricted to rigid rotations, and the point clouds can be mapped into Euclidean spaces of a lower dimension. This is better-suited for cross-modal applications where scaling is involved. More precisely, given two point clouds represented by matrices $X \in \mathbb{R}^{d_1 \times N}$ and $Y \in \mathbb{R}^{d_2 \times N}$, each with N points arranged along columns assumed to be in correspondence between X and Y, CCA is defined as

$$d_{\text{CCA}} = \min_{V_x \in \mathbb{R}^{d_x \times b}, V_y} ||V_x^T X - V_y^T Y||_F^2 \tag{6}$$

for some chosen constant $b \leq \min(d_1, d_2)$, s.t.

$$V_x^T X X^T V_x = V_y^T Y Y^T V_y = I_b \tag{7}$$

In other words, find linear projections for each point cloud that map into the same space $\mathbb{R}^b$, where the sum of the squares is minimized in that space. This can also be solved with a singular value decomposition. Like Procrustes, this assumes that the correspondences are known a priori. To find the correspondences, the authors in [57] take the same iterative approach as that authors in [53] did with Procrustes, but they alternate back and forth between DTW and CCA instead of DTW and Procrustes. An updated version of this algorithm known as "generalized time warping" (GTW) [55], [56] was developed which aligns multiple sequences using a single optimization objective where the spatial alignment and time warping are coupled, but this still requires an an initial guess for a nonconvex optimization procedure. Hence, CTW and GTW are susceptible to the same problems getting stuck in a local min with a bad initial warping path. Finally, a recent work in [46], [47] takes a similar approach, but it replaces CCA by learning features in the projection stage with a deep neural network. Like all supervised learning approaches, however, this method requires training data with known correspondences.

3) Isometry Invariant Surrogate Techniques: There have also been a number of techniques which derive a surrogate function which is invariant to isometries of the input time-ordered point cloud before performing time warping. "Derivative dynamic time warping" (DDTW) computes a coarse estimate of curvature, which is fed to the ordinary DTW algorithm [29]. The authors in [39] use a more refined version of curvature and length with edit distance instead of DTW to accomplish the alignment. There are also variants which use the fast marching method instead of dynamic time warping to align curvature profiles [18], and variants which use integrated curvature instead of curvature, which is scale invariant due to Gauss-Bonnet [13]. Some works use the triangle area between triples of points as an invariant [2], and some use the turning angle of the curve [12], which is related to curvature. These techniques all suffer from similar drawbacks of the numerical difficulty of computing some of the invariants, as well as the fact that most of the invariants are local, so small differences can cause the curves "drift" over time (e.g. as shown in [39] a U is similar to a 6 with local curvature).

4) Miscellaneous Schemes: There are a variety of other schemes that attack the problem of time series synchronization or curve matching in some form. The authors of [51] attack a more general case of cross-domain object matching (CDOM) with general correspondences and address warping paths as a special case. However, their problem reduces to the quadratic assignment problem, which is NP-hard, and their iterative approximation requires a good initial guess. The authors of

[3]This analogous to the difference between the Fréchet Distance [3] and the Hausdorff Distance between curves.

[48] attack a special case where curves form closed loops, using cohomology to find maps from point clouds to the circle, where they can be synchronized, but this provides a coarse alignment in practice and only works for periodic motions. Some attempt jointly align curves on manifolds [19], though this assumes that they lie on smooth manifolds, which must be found numerically. Finally, perhaps the most similar to our approach is the action recognition work of [25], which shows that small patches of SSMs can be put into sequence with a dynamic time warping algorithm to align time warped actions from different camera views. However, [25] does not make use of the entire self-similarity matrix as we do, relying only on pixels close to the diagonal of SSMs, and their scheme does not extend across modalities.

5) Our Approach: Unlike the EM-based and optimization approaches, our SSM-based approach avoids a spatial alignment/manifold alignment/initial guess. Unlike curvature-based surrogates and local self-similarity information, we preserve global properties of the curve. Finally, like some of the techniques (CCA-based, manifold alignment, integrated curvature), we have a normalization scheme to address scaling issues between modalities (Section III-D), so our technique is more general than isometric curve alignment.

E. Smith Waterman Local Alignment

So far, we have focused on schemes which require a global alignment, which requires the first points and last points in each sequence to be matched. However, this is a difficult constraint to satisfy in practice and often requires manually cropping the data to start and end in similar places. To migigate this, there is an algorithm for *local alignment* known as "Smith Waterman" [42], [49] which seeks the best contiguous subsequences in each time series which match each other. This algorithm was originally developed for gene sequence alignment, but it has been adapted to multimedia problems such as music alignment [40] and video copyright infringement detection [10]. In our work, we use Smith Waterman to extend our self-similarity based time warping to find the best contiguous subsequences of time-ordered point clouds.

Unlike dynamic time warping, Smith Waterman seeks to *maximize* an alignment *score*, but the alignment does not have to start on the first element of each sequence or end on the last element on each sequence. To solve this, the exact same dynamic programming algorithm can be used, except there is one extra "restart" condition if a local alignment has become sufficiently poor. The recurrence is as follows

$$SW_{ij} = \max \left\{ \begin{array}{c} SW_{i-1,j-1} + m(a,b) \\ SW_{i-1,j} + g \\ SW_{i,j-1} + g \\ 0 \end{array} \right\}, SW_{0,j} = SW_{i,0} = 0 \tag{8}$$

where $m(a,b)$ depicts the match/mismatch score between characters a and b (positive if matching, negative if mismatching), and g specifies a *gap penalty* which is incurred if a character is added/deleted from one of the sequences. Figure 3 shows the result of aligning the word "geography" to the word "geology" with alignment score $+2$, misalignment cost -2, and gap penalty -1. A directed acyclic graph (DAG) is drawn with arrows depicting the optimal step (add/delete/align) which is taken locally, which, as in dynamic time warping, can be back-traced to extract an optimal alignment. In this example, there are several different local alignments with the same optimal score of 6. One optimal local alignment is between "geo" and "geo." Another optimal alignment is between "geolog" and "geog," which can be accomplished two different ways; by aligning "geo," deleting "lo" from geology, and aligning "g," or by aligning "ge," deleting "ol" from geology, and then aligning "og." These two choices correspond to two different paths in the DAG between cells $(4,6)$ and $(0,0)$.

	–	g	e	o	l	o	g	y
–	0	0	0	0	0	0	0	0
g	0	2	1	0	0	0	2	1
e	0	1	4	3	2	1	1	0
o	0	0	3	**6**	5	4	3	2
g	0	2	2	5	4	3	**6**	5
r	0	1	1	4	3	2	5	4
a	0	0	0	3	2	1	4	3
p	0	0	0	2	1	0	3	2
h	0	0	0	1	0	0	2	1
y	0	0	0	0	0	0	1	4

Fig. 3: An example of the dynamic programming table that results by computing Smith Waterman between "geology" and "geography," using a matching cost of $+2$, a non-matching cost of -2, and a gap cost of -1. Back-pointers are shown to depict which alignments (addition/deletion/insertion) yield the best local alignment, and the cells corresponding to the ends of these alignments are bolded. Red arrows are drawn along optimal alignment paths in the resulting DAG.

Like DTW, we can modify Smith Waterman to return the best subsequence that matches the i^{th} character in the first sequence to the j^{th} character in the second sequence. This can be accomplished by running Smith Waterman between $\{X_1, X_2, ..., X_i\}$ and $Y_1, Y_2, ..., Y_j$, and then again between the reversed sequences $\{X_M, X_{M-1}, X_{M-2}, ..., X_i\}$ and $\{Y_N, Y_{N-1}, Y_{N-2}, ..., Y_1\}$.

III. Isometry Blind Dynamic Time Warping (IBDTW)

Most of the approaches we reviewed to align time series which have undergone linear transformations try to explicitly factor out those transformations before doing an alignment, but this is not necessary if we build our algorithm on top a self-similarity matrix between two point clouds, which is already blind to isometries. We note an additional advantage that we no longer need to restrict ourselves to Euclidean spaces.

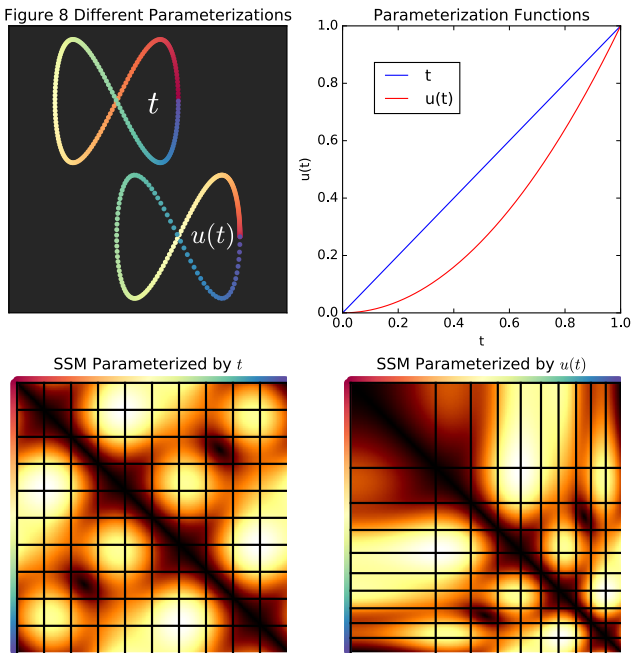


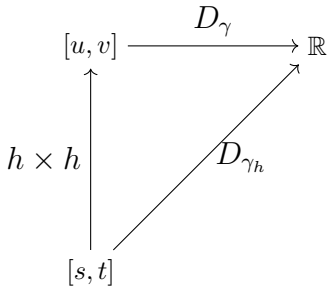
Fig. 4: Self-similarity images of different parameterizations of a Figure 8. Rectangles in one image map to rectangles in the other image, and lines in one image map to lines in the other image.

A. Induced Self-Similarity Warping Functions

To set the stage for our algorithms, we first study the maps that are induced between self-similarity images by re-parameterization functions, which will help in the algorithm design. For example, take the figure 8 curve, which we designate γ_8 (upper left, Figure 4)

$$\gamma_8(t) : [0,1] \to \begin{bmatrix} \cos(2\pi t) \\ \sin(4\pi t) \end{bmatrix} \tag{9}$$

The bottom left of Figure 4 shows the SSM of a linearly parameterized sampled version of this curve, while the bottom right of Figure 4 shows the SSM corresponding to a re-parameterized sampled version. Maps between the domains of the SSMs shown are always rectangles, and they are independent of underlying curve being parameterized (they only depend on the relationship between two parameterizations). This can be seen by starting with a space curve $\gamma : [0,1] \to \mathbb{R}^d$ and its resulting self-similarity image D_γ. Given a homeomorphism $h : [0,1] \to [0,1]$, which gives rise to a space curve $\gamma_h : [0,1] \to \mathbb{R}^d$ and a corresponding self-similarity image D_{γ_h}, there is an induced homeomorphism, $h \times h$ from the square to itself between the two domains of D_γ and D_{γ_h}. The commutative diagram below shows all of the maps that are involved



That is,

$$D_{\gamma_h} = D_\gamma(h(s), h(t)) \tag{10}$$

If we fix a correspondence $s \iff u = h(s)$, then this shows that row $h(s)$ of D_γ is a 1D re-parameterization of row s of D_{γ_h}, making rigorous the observation in Figure 1. Note that for a discrete version of these maps between time-ordered point clouds, one can replace the homeomorphism h with a warping path $\mathcal{W}$, and the relationships are otherwise the same.

B. Isometry Blind Dynamic Time Warping Algorithm

We now have the prerequisites necessary to define our main algorithm. The idea is quite simple. Based on our observations and Figure 1 and Figure 4, if we know that point i in a time-ordered point cloud (TOPC) X is in correspondence with a point j in TOPC Y, then we should match the i^{th} row of X's SSM to the j^{th} row of Y's SSM under the L_1 distance, enforcing the constraint that $(i,j) \in \mathcal{W}$. However, since we don't know ahead of time which rows should be in correspondence, we try every row i of SSM A against every row j in SSM B, and we create a *cross-similarity time warping matrix (CSWM)* C so that $C_{ij} = \text{constrainedDTW}(\text{SSMA}_i, \text{SSMB}_j)$ (Algorithm 2). Then, we can apply the ordinary DTW algorithm to C. Algorithm 3 summarizes this process.

Figure 5 shows an example of this algorithm on two rotated/translated/re-parameterized time-ordered point clouds in $\mathbb{R}^2$ (point clouds 1 and 2). As can be seen by the colors indicating correspondences, the algorithm was able to put the points into correspondence correctly even without first spatially aligning the curves. We also show alignment to a third time-ordered point cloud, which is metrically distorted in addition to being rotated/translated/re-parameterized. The returned warping degrades gracefully. We will explore this more rigorously in Section V-A.

C. Analysis: Lower Bounding L1 Metric Stress

All of our work in this section can be put into the *Gromov-Hausdorff Distance* framework, which describes how to "embed" one metric space into another. More formally, given two discrete metric spaces (X, d_X) and (Y, d_Y), and a correspondence $\mathcal{C}$ between X and Y, the *p-stress* is defined as

$$\mathcal{S}_p(X,Y,\mathcal{C}) = \left(\sum_{(x,y),(x',y') \in \mathcal{C}} (d_X(x,x') - d_Y(y,y'))^p \right)^{1/p} \tag{11}$$

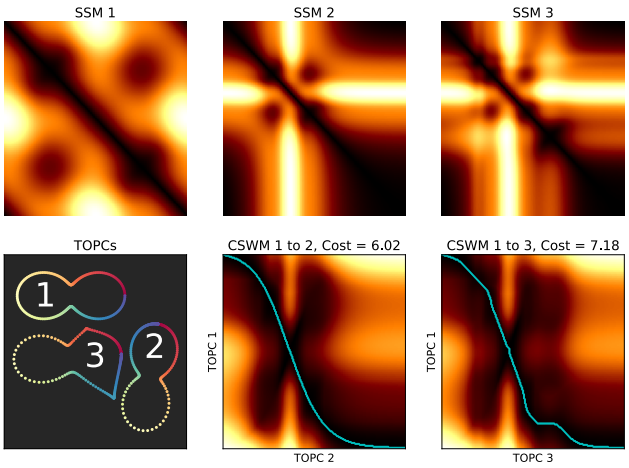


Fig. 5: An example of IBDTW between 3 different samplings of a pinched ellipse. The optimal warping path found by Algorithm 3 is drawn in cyan on top of the CSWM in each case. Based on this, points which are in correspondence are drawn with the same color in the upper right figure. Though time-ordered point cloud 2 has more points towards the beginning and fewer points towards the end than time-ordered point cloud 1, correct regions are put into correspondence with each other. Furthermore, in addition to being parameterized this way, the time-ordered point could 3 is also distorted geometrically, but the correspondences are still reasonable.

The L^∞ stress $\mathcal{S}_\infty(\mathcal{C})$, or the maximum stretching induced by a correspondence, is often referred to as the "distortion" of a correspondence. Intuitively, the p-stress measures how much one has to stretch one metric space when moving it to another. The Gromov-Hausdorff Distance is based off of the L_∞ stress specifically [22], given two discrete metric spaces (X, d_X) and (Y, d_Y), the *Gromov-Hausdorff Distance* $d_H(X, Y)$ between X and Y is

$$d_{\text{GH}}(X, Y) = \frac{1}{2} \inf_{\mathcal{C} \in \Pi} \mathcal{S}_\infty(X, Y, \mathcal{C}) \tag{12}$$

where Π is the set of all correspondences between X and Y. In other words, the Gromov-Hausdorff Distance measures the smallest possible *distortion* between a pair of points over all possible embeddings of one metric space into another. Unfortunately, the Gromov-Hausdorff Distance turns out to be NP-complete, and there is no known algorithm to even approximate it within a constant factor [1]. The authors in [9] develop a practical algorithm to approximate the 2-stress between two polygon mesh surfaces. But even in this special case, they pose a highly nonconvex quadratic optimization problem, which is only guaranteed to converge to a local minimum and hence which needs a good initial guess, though they showed success for applications of 3D face matching.

1) Lower Bound Proof: We now connect Algorithm 3 to the Gromov-Hausdorff Distance. In particular, we have the following lemma:

Lemma 1. *The cost returned by Algorithm 3 lower bounds* $\mathcal{S}_1(X, Y, \mathcal{W})$*, or the 1-stress* restricted to warping paths.

Algorithm 3 Isometry Blind Dynamic Time Warping (IBDTW)

1: **procedure** IBDTW(X, Y, d_X, d_Y)
2: ▷ TOPCs X and Y with M and N points, metrics d_X and d_Y, respectively
3:
4: ▷ Initialize cross-similarity warp matrix (CSWM)
5: C $\leftarrow$ $\underbrace{\begin{bmatrix} 0 & \dots & 0 \\ 0 & \dots & 0 \\ \vdots & \dots & \vdots \\ 0 & \dots & 0 \end{bmatrix}}_{N} \Big\} M$
6: **for** $i = 1 : M$ **do**
7: ▷i^{th} row of d_X
8: $A \leftarrow [d_X(x_i, x_1), d_X(x_i, x_2), \dots, d_X(x_i, x_M)]$
9: **for** $j = 1 : N$ **do**
10: ▷j^{th} row of d_Y
11: $B \leftarrow [d_Y(y_j, y_1), d_Y(y_j, y_2), \dots, d_Y(y_j, y_N)]$
12: Algorithm 2
13: $C_{ij} \leftarrow$ ConstrainedDTW(A, B, L_1, i, j)
14: **end for**
15: **end for**
16: ▷ Use the CSWM C as the distance in the ordinary DTW algorithm (Algorithm 1)
17: $D \leftarrow \frac{1}{2}\text{DTW}(X, Y, C)$
return (D, C) ▷ Return the cost and the CSWM
18: **end procedure**

Proof: Note that the optimal IBDTW warping path $\mathcal{W}^*$ has the following cost $c(\mathcal{W}^*)$

$$c(\mathcal{W}^*) = \sum_{(x_i, y_j), (x', y') \in \mathcal{W}^*} |d_X(x_i, x') - d_Y(y_j, y')| \tag{13}$$

which can be rewritten as

$$c(\mathcal{W}^*) = \frac{1}{2} \sum_{(x_i, y_j) \in \mathcal{W}^*} \sum_{(x', y') \in \mathcal{W}^*} |d_X(x_i, x'), -d_Y(y_j, y')| \tag{14}$$

since if $(x', y') = (x_i, y_j)$ then the cost is zero, all other terms counted twice. Now fix an x_i and y_j. Then the sum of the terms of the form $|d_X(x_i, x') - d_Y(y_j, y')|$ is simply the L_1 warping distance between 1D time series which are the i^{th} row of d_X, $d_X[i, :]$ and the j^{th} row of d_Y, $d_Y[j, :]$ under the warping $\mathcal{W}^*$. Note that the DTW Distance between $d_X[i, :]$ and $d_Y[j, :]$ is at most the L_1 warping distance under $\mathcal{W}^*$, and is potentially lower since we are computing them greedily only between x_i and y_j, ignoring all other constraints. Hence, the sum of the terms $|d_X(x_i, x') - d_Y(y_j, y')|$ is lower bounded by Line 13 in Algorithm 3. ■

One subtlety is that although we have shown that the cost $D[M, N]$ returned by Algorithm 3 lower bounds $c(\mathcal{W}^*)$, the cost of the warping path $\hat{\mathcal{W}}$ obtained by backtracing through D has a cost which is greater than or equal to the optimal warping path: $c(\hat{\mathcal{W}}) \geq c(\mathcal{W}^*)$. This is because D was computed in

a greedy manner only considering pairs of rows at any step of the algorithm, which could lead to an overall suboptimal warping path with respect to all constraints.

2) Approximating the Gromov-Hausdorff Distance: For a more direct analogy with DTW, Algorithm 3 was designed to lower bound the 1-stress restricted to warping paths, but a very similar technique could be used to lower bound the Gromov-Hausdorff Distance restricted to warping paths. The constrained DTW in the inner loop in Line 13 can be replaced instead with a constrained version of the discrete Fréchet Distance [16] to find the maximum distortion induced by putting two points in correspondence, and the discrete Fréchet Distance can be run on the final CSWM. In this work, however, we stick to the 1-stress, since it gives a more informative overall picture of the full metric space.

D. Cross-Modal Normalization via Histogram Matching

The scheme we have presented so far works well for point clouds sampled from isometric curves, but the isometry assumption does not usually hold in cross-modal applications. Not only can the scales be drastically different between modalities, but it is unlikely that a uniform re-scaling will fix the problem. For instance, consider a 1D oscillating bar of length B oscillating sinusoidally over the interval $[0, A]$ with a period of T. Then its center position can be measured as $c_t = (A/2) + (A/2)\cos(2\pi t/T)$. We call this type of measurement, which follows the object in question, a measurement in *Lagrangian coordinates*. By contrast, suppose we take a 1D video of the bar with A pixels, where each pixel in each frame measures occupancy by the bar at that frame in the video. Then pixel i in video X_t can be parameterized by time as

$$X_t[i] = \left\{ \begin{array}{cc} 1 & |c_t + B/2 - i| < B/2 \\ 0 & \text{otherwise} \end{array} \right\} \quad (15)$$

These pixel by pixel measurements at fixed positions are referred to as *Eulerian coordinates*. The top left and lower left plots in Figure 6 show these two different views into the same data. Let the Lagrangian SSM D_1 simply be the 1D metric between two different centers, $D_1[s,t] = |c_s - c_t|$, and let the Eulerian SSM D_2 be the Euclidean metric $D_2[s,t] = ||X_s - X_t||_2$ between each frame of the video. Then the second column of Figure 6 shows D_1 and D_2. Although they are measuring the same process, the SSMs have a locally different character. Rows of D_1 are perfect sinusoids, while rows of D_2 are more like square waves, since there are sharp transitions from foreground to background in Eulerian coordinates.

To address this kind of local rescaling between an SSM D_1 and an SSM D_2, we first divide each SSM by its respective max, and we quantize each to L levels evenly spaced in $[0, 1]$. We then apply a monotonic, one-to-one map f to each pixel in D_1 so that the CDF of D_1 approximately matches the CDF of D_2 (see, e.g., [20] ch. 3.3)[4]. Note that this process can be done from D_1 to D_2 or from D_2 to D_1, as shown in Figure 6 for the Eulerian/Lagrangian example. Since this process is not necessarily symmetric, we perform both sets of normalizations, and we choose the one which yields a lower alignment cost with the IBDTW algorithm.

[4]This is a constrained, simplified version of 1D optimal transport on the histograms

E. Computational Issues / Parallelization

The time complexity of this algorithm is $O(N^2M^2)$, since an $O(MN)$ dynamic programming algorithm is called for each pair $(i, j) \in (1, 2, \ldots M) \times (1, 2, \ldots, N)$. This quartic time complexity can be prohibitively expensive in practice. To mitigate this, we use a GPU algorithm that was designed by [54] for Smith Waterman on gene sequences. It uses what's known as a *linear systolic array* to parallelize computation. In particular, instead of processing the elements of the dynamic programming matrix in "raster order" (row by row from left to right across each row), it is possible to process them along diagonal lines that propagate from the upper left to the lower right. The dependencies are satisfied simultaneously for all points along a diagonal line, so these elements can be computed in parallel. This takes $O(M + N)$ computation time instead of $O(MN)$ computation time, since there are $N + M - 1$ red lines that need to be computed in sequence, but with enough parallel units the processing of each red line is $O(1)$. Furthermore, in the case of Algorithm 3, there are NM dynamic time warping problems that need to be computed in to obtain the CSWM C (all pairs of SSM rows), but these can also be computed in parallel. In CUDA, for instance, each pixel of C can be computed in its own block, and each element along a red line can be computed by a thread within the corresponding block. Therefore, with enough parallel units, Algorithm 3 is itself reduced to $O(M + N)$ from $O(M^2N^2)$, although in practice we usually don't have that many parallel units. On our machine, we witness a speedup of between 20-30x of a CUDA implementation of the linear systolic array over a C implementation of serial DTW between point clouds with several hundred points.

IV. Isometry Blind Partial Time Warping

One of the drawbacks of the technique we presented so far is that it requires a global alignment. However, if the sequences only partially overlap, forcing a global alignment leads to poor result, unless manual cropping is done to ensure that sequences start and end at the same place [57]. To automate this cropping, we design an isometry blind time warping algorithm that can do partial alignment. This algorithm is very similar to IBDTW, except constrained DTW is replaced with constrained Smith Waterman using a special match function. From this, we build a "partial cross-subsequence score matrix" (PCSSM) to hold best partial alignments between all pairs of rows, and we run a final Smith Waterman computation using some function of the PCSSM as match/mismatch cost. As with IBDTW, the SSMs can be histogram remapped before running the algorithm (Section III-D), and parallelization can be applied to make this an $O(N)$ algorithm (Section III-E).

More specifically, define a subroutine SMWat(A, B, m, g) which returns unconstrained Smith Waterman between two sequences A and be with a match/mismatch score $m(a, b)$ and a horizontal/vertical gap penalty g. Also define a subroutine ConstrainedSMWat(A, B, m', g', i, j) which returns the

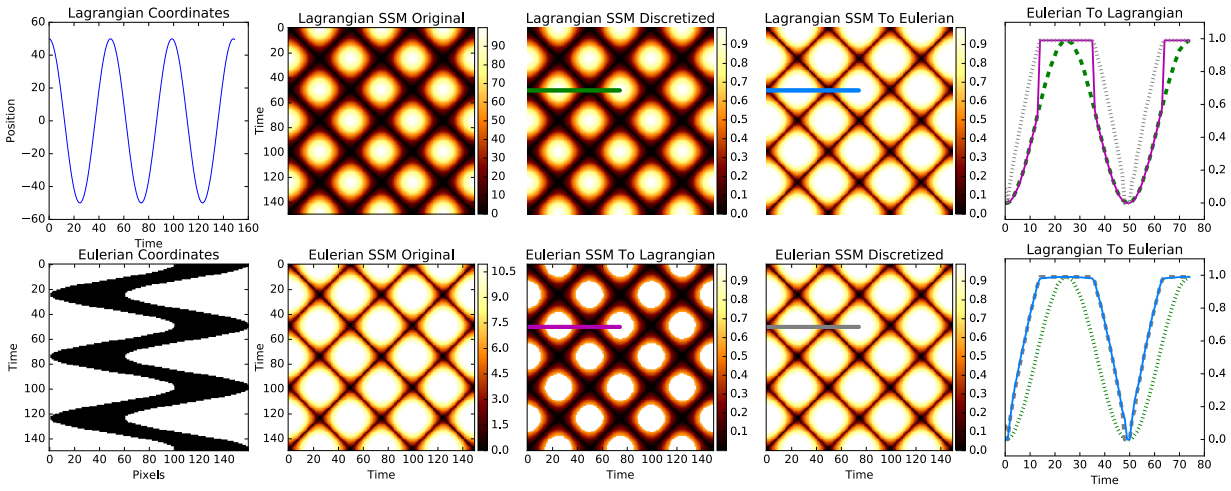


Fig. 6: An example of matching the SSM of an oscillating line segment captured with Lagrangian coordinates to an SSM of the oscillating line segment captured with Eulerian coordinates, and vice versa. The right column shows an example of a row from each of the matrices in different cases. The stipple line pattern shows the original row, the line segment shows the corresponding row from the SSM with the target distribution, and the solid row shows the remapped version of the original row. In this case, it is easier to remap the Lagrangian coordinates to Eulerian coordinates, though both remappings are closer to the target than the original.

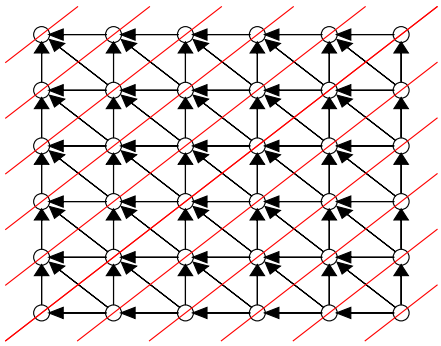


Fig. 7: A schematic of the linear systolic array used to compute DTW in parallel. Elements of the dynamic programming matrix are drawn as circles. Black arrows are drawn to show dependencies between elements in the dynamic programming matrix, and red lines show elements which can be computed in parallel if the processing occurs from the upper left $(0, 0)$ to the lower right (M, N).

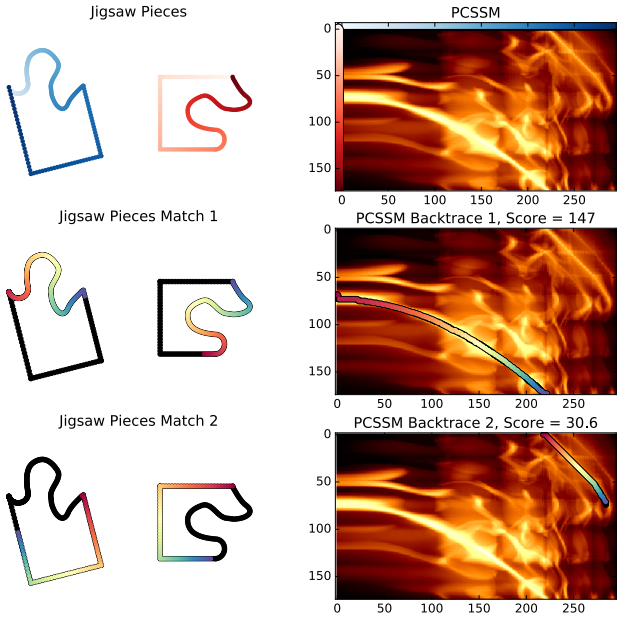


Fig. 8: An example of partial alignment with IBPTW on jigsaw puzzle pieces. The optimal partial alignment is shown on the middle row. The bottom row shows a locally optimally partial alignment with a lower score.

score of the optimal subsequence that matches character i in the sequence A to character j in the sequence B, using a matching/mismatching cost function $m'(a, b)$ and a fixed vertical/horizontal gap penalty of g'. Then, the Isometry Blind Partial Time Warping (IBPTW) algorithm is defined in Algorithm 4.

In practice, we define $m_1(a, b) = \exp(-|d(a, b)|/\sigma) - 0.6$, where in our experiments we usually choose $\sigma = 0.1$. If $d(a, b)$ is the L1 distance between elements of two histogram normalized SSM rows, then it ranges between 0 and 1. Thus, there is a positive matching score of up to 0.4 for the most similar pixels between the normalized SSMs and a negative matching score of -0.6 between the most dissimilar pixels. We also choose a gap penalty of -0.4 to promote diagonal matches[5]. For the outer loop which runs Smith Waterman

[5]Otherwise, the warping path maximizing the alignment score tends to have a preference for longer horizontal and vertical lines, leading to many undesirable pauses of one time series with respect to the other

Algorithm 4 Isometry Blind Partial Time Warping (IBPTW)

procedure IBPTW(X, Y, d_X, d_Y, m_1, g_1, m_2, g_2)
▷ TOPCs X and Y with M and N points, metrics d_X and d_Y, respectively, a match function m_1 to score elements in correspondence between rows in the inner loop, in inner loop horizontal/vertical gap cost g_1, and an outer loop match function m_2 and gap cost g_2.

▷ Initialize partial cross subsequence score matrix (PCSSM)

S $\leftarrow$ $\underbrace{\begin{bmatrix} 0 & \dots & 0 \\ 0 & \dots & 0 \\ \vdots & \dots & \vdots \\ 0 & \dots & 0 \end{bmatrix}}_{N}$ $\Big\}M$

for $i = 1 : M$ **do**
$A \leftarrow [d_X(x_i, x_1), d_X(x_i, x_2), \dots, d_X(x_i, x_M)]$
for $j = 1 : N$ **do**
$B \leftarrow [d_Y(y_j, y_1), d_Y(y_j, y_2), \dots, d_Y(y_j, y_N)]$
$S_{ij} \leftarrow$ ConstrainedSMWat(A, B, m_1, g_1, i, j)
end for
end for
▷ Use the PCSSM S as the distance in the ordinary Smith Waterman algorithm after applying the matching function to it
$\hat{S} \leftarrow m_2(S)$
$D \leftarrow$ SMWat$(A, B, \hat{S}, g_2)$
return (D, S) ▷ Return the cost and the PCSSM
end procedure

using the PCSSM S, we choose

$$m_2(a_i, b_j) = \frac{(S_{ij} - \text{md}(S))}{\max(|S - \text{md}(S)|)}$$

where md is the median operation. This will give a high score up to ≤ 1 to row pairs which have a high subsequence score in common, and a low score ≥ -1 to rows which do not have a good subsequence. Figure 8 shows an example of running this algorithm on two jigsaw puzzle pieces which should fit together with m_1 and m_2 defined as above, and $g_1, g_2 = -0.4$. The longest subsequence is indeed along the cutouts where they match together. It is also possible to backtrace from anywhere in the PCSSM to find other subsequences which match in common, so Figure 8 shows an example of suboptimal but good local alignment.

V. Experimental Results of Global Alignment

In this section, we will quantitatively compare the IBDTW algorithm for global alignment with several other techniques in the literature, including ordinary dynamic time warping (DTW), derivative dynamic time warping (DDTW) [29], canonical time warping (CTW) [57], Generalized Time Warping (GTW) [55], [56], and Iterative Motion Warping (IMW) [23] (a simpler version of CTW which is restricted to the same space). We use code from [57] and [55], [56] to compute all of these alignments[6]. We use the default parameters provided in this code, and, as in [57] and [55], [56], we use the results of DTW to initialize CTW and GTW. In all of our experiments, we show results both from IBDTW and IBDTW after SSM rank normalization, which we refer to as "IBDTWN." When we have access to a ground truth warping path, we report the alignment error as the area between the ground truth warping path and the computed warping path, divided by the total area of the cross-similarity matrix, as in [57]. Figure 9 shows an example of the different alignment techniques and their alignment errors on a synthetic TOPC.

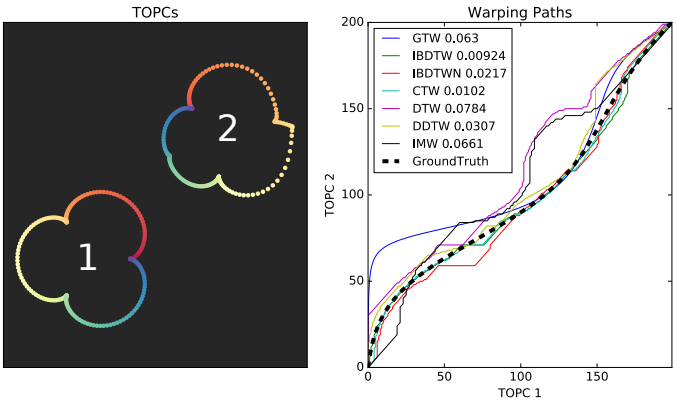


Fig. 9: Comparisons of different time warping techniques on a warped epicycloid point cloud. Alignment errors are displayed in the legend next to the technique name.

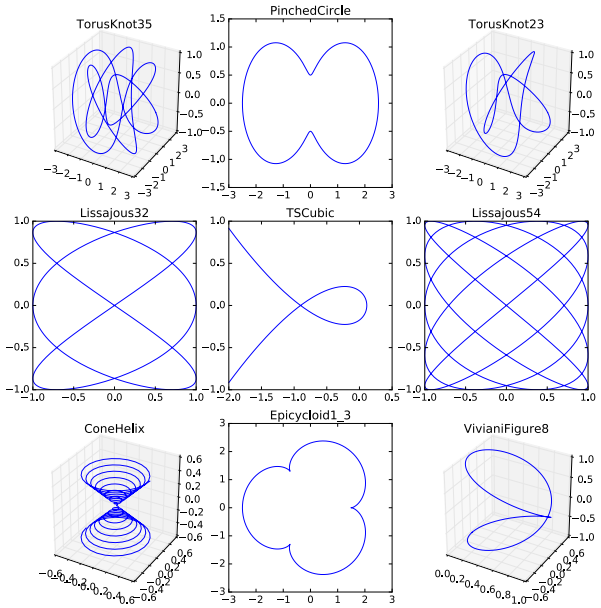


Fig. 10: The families of curves that we use in our synthetic global alignment experiment.

A. Synthetic Experiments

We first perform an experiment aligning rotated/translated/flipped and re-parameterized time-ordered

[6] http://www.f-zhou.com/ta_code.html

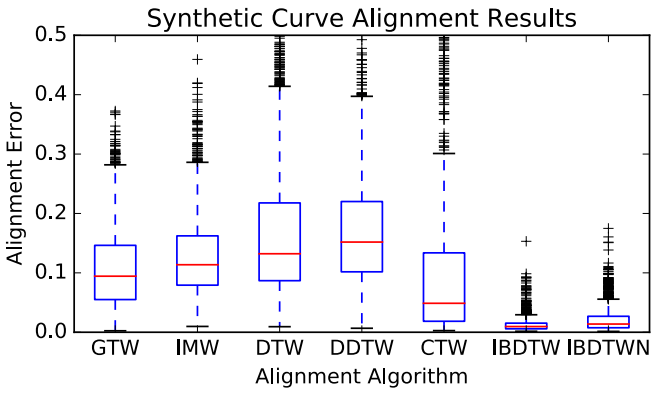


Fig. 11: Comparisons of alignment errors for different techniques on synthetic rotated/translated/re-paramterized/distorted 2D and 3D curves drawn from the classes in Figure 10.

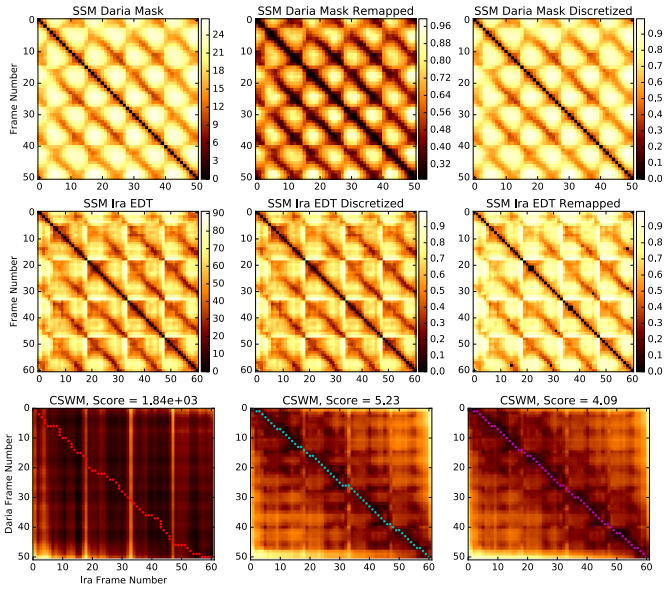


Fig. 12: Running IBDTW with and without histogram matching on binary mask features from "Daria" matched to EDT features from "Ira." CSWMs are shown on the bottom without normalization (left), and with histogram matching both ways (bottom center / bottom right), and optimal warping paths are superimposed. Figure 13 shows parts of the magenta warping path on the lower right.

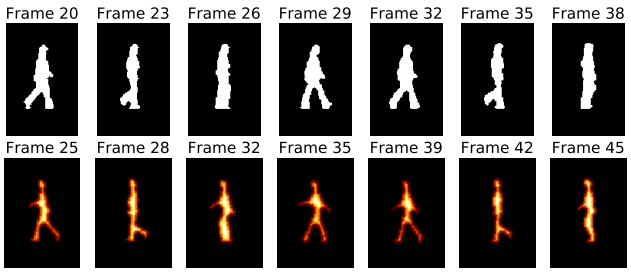


Fig. 13: An example of binary mask features from "Daria" walking (top) in the Weizmann dataset [21] aligned with Euclidean Distance Transform (EDT) features from "Ira" walking the other way (bottom), using IBDTW. Ira is walking faster, so she goes through more frames for the same motion under the optimal alignment.

point clouds (TOPCs) from a family of 9 curves in Figure 10. As in [56], we re-parameterize the curves with random linear combinations of polynomial, logarithmic, exponential, and hyperbolic tangent functions. To distort the curves, we move random control points in random directions after spatial transformation and re-parameterization. Let X be the TOPC before transformation/re-paramterization/distortion and let Y be the curve afterwards. The average ratio of $d_{GH}(X,Y)/\text{diam}(X)$, where diam is the "diameter" of X (the maximum inter point distance), is 0.18. Figure 11 shows the results. IBDTW performs the best, while doing the normalization for IBDTWN only degrades the results slightly in this example. Unsurprisingly, CTW and GTW are the next best techniques.

B. Weizmann Walking Videos

As our next experiment, we compare to the benchmark presented in [57], which aligns 4 videos of people walking from the Weizmann dataset [21] cropped to 4 walking cycles each. As in [57], we use different features for the two videos undergoing alignment. On one video, we use the binary mask of the foreground object of the person walking, where every pixel is a dimension, and every frame is a point in the TOPC. On the second video, we use the Euclidean distance transform (EDT) [32]. This is the first time we need to apply normalization to real data, so we show the original and normalized SSMs and CSWMs for an example in Figure 12. Normalization is clearly necessary due to the different scales. There is a slightly lower alignment score when remapping Ira's SSM to Daria's SSM, so that is the alignment path taken, part of which is shown in Figure 13.

To assess performance more rigorously, we create a more controlled experiment where the second video is the same as the first video after a time warp and applying EDT, so that we have access to the ground truth[7]. Figure 14 shows the results. CTW performs slightly better than IBDTWN, but not appreciably. It is also clear why the normalization is needed in cross-modal applications, as IBTW without normalization has the worst average alignment error.

C. BU4D Face Expressions Dataset

We now perform experiments on a much more difficult cross-modal application which is meant to "stress test" the different techniques. We synchronize expressions drawn from the BU 4D face dataset [52], which includes RGB videos of people making different facial expressions, a 3D triangle mesh corresponding to each RGB frame. An example sequence is shown in Figure 15. For our RGB features, we simply

[7]This differs from the experiment in [57], who use DTW between two different subjects as ground truth, but we believe the ground truth in our experiment is a better control.

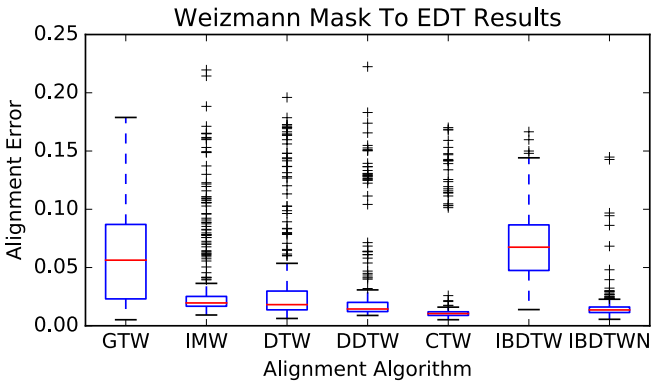


Fig. 14: Comparisons of different time warping techniques on walking videos from the Weizmann action dataset [21].

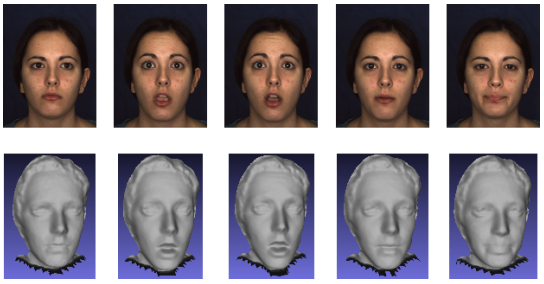


Fig. 15: Subject 5 in the BU dynamic 3D face dataset [52], performing the "surprised" expression. 2D RGB frames are on the top and the corresponding 3D meshes are on the bottom.

take each channel and each pixel to be a dimension, so a video with $W \times H$ pixels lives in $\mathbb{R}^{3WH}$. For the 3D mesh features, we use shape histograms [4] after mean centering and RMS normalizing each mesh. Our shape histograms have 20 radial shells, each with 66 sectors per shell with centers equally distributed across the sphere, so the 3D features live in $\mathbb{R}^{1320}$. Both of these feature spaces suffer from the curse of dimensionality and other problems such as "bin hopping." As a result, as shown in the example in Figure 17, most of the alignment techniques fail outright, and IBDTW is the only one that gets close. Similarly to the Weizmann dataset, we perform

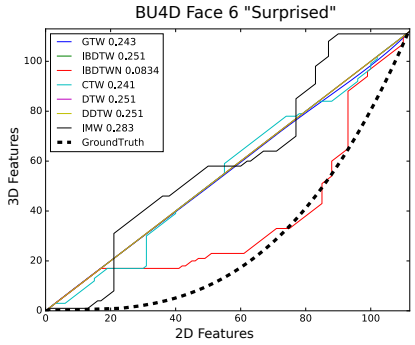


Fig. 16: An example of warping paths computed between 2D and 3D features of subject 6 in the BU4D dataset forming a "surprised" expression.

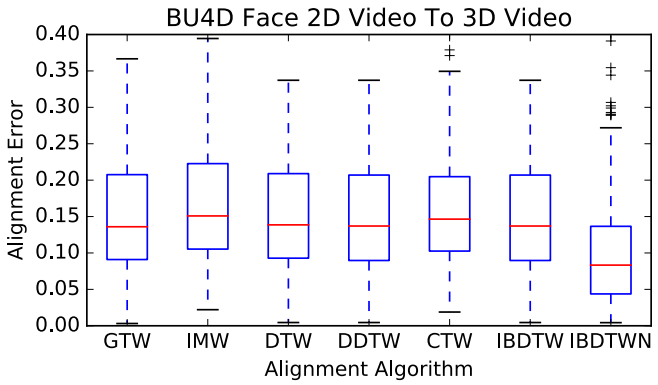


Fig. 17: Alignment errors on the BU4D face expressions dataset.

an experiment where we take the 2D features from a face and the 3D features of the same face which has been warped in time, so that we have access to ground truth. We perform 10 such warpings and alignments for 9 faces from 6 different expression types, for a total of 560 experiments. Figure 17 shows the results. The performance is strikingly worse than the Weizmann dataset, with most techniques performing equally poorly, though the normalized IBDTW has about half of the alignment error of other techniques.

D. Discussion

As we have shown by our experiments, IBDTW performs excellently when aligning nearly isometric sampled curves, and normalized IBDTW is competitive with state of the art for cross-modal applications. Furthermore, our technique requires no parameters, while, in our experience, CTW and GTW require many parameters that make or break performance. We also noticed that CTW is very slow without an aggressive dimension reduction, while our technique is dimension independent by nature of using SSMs.

VI. Experimental Results of Partial Alignment

We now evaluate IBPTW on real data to complement our IBDTW experiments.

A. Weizmann Alignment

First, we follow up on an experiment we did in Section V-B. In that section, like the authors in [57], [56], and [46], we had to manually crop the videos so that they started and ended at the same place. With our partial alignment algorithm, on the other hand, no such preprocessing is necessary. We demonstrate this applying IBPTW to uncropped videos in which one person walks more strides than the other person. Figure 18 shows the results. Denis takes longer strides, so he walks across the field of view in fewer steps. His also starts a couple of frames later into his stride than Daria. The partial warping path extracted in Figure 18 correctly accounts for both of these facts, and it properly locally the frames (please refer to the supplementary video).

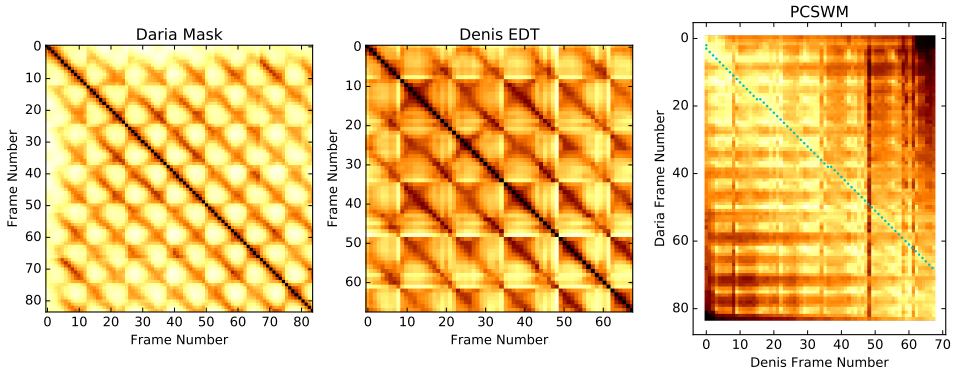


Fig. 18: An example of running IBPTW to partially align binary mask features from an uncropped video of Daria walking with EDT features from an uncropped video of Denis walking. SSMs are shown for each in the left two columns, while the PCSWM is shown with the optimal warping path superimposed in the right column.

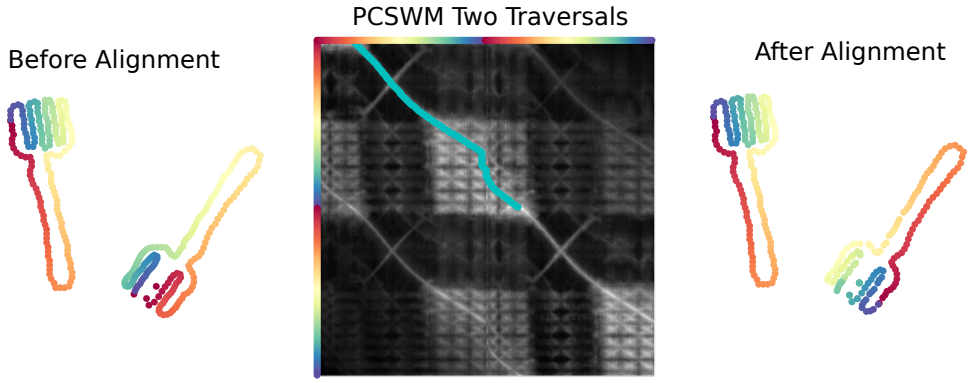


Fig. 19: An example of aligning two closed loops in the shape of a fork which have been rotated/translated/re-parameterized, in addition to starting at different points. The left plot shows the forks before alignment. The center plot shows the PCSWM resulting from aligning both point clouds each repeating themselves twice, as shown by the colors along the rows (left fork) and columns (right fork), which correspond to the colors in the left plot. The optimal partial warping path truncated to the first repetition of the left fork is superimposed in cyan. The forks in the right plot are put into correct correspondence by this truncated partial warping path.

B. Isometry Blind Loop Alignment

We now return to the realm of 2D curve alignment once more to showcase our IBPDTW algorithm. This time, we focus on time-ordered point clouds sampled from *loops*. A 2D loop is a curve $\gamma : [0, 1] \to \mathbb{R}^2$ so that $\gamma(0) = \gamma(1)$. In other words, a loop is a curve that ends up where it started. It is possible to define a geometrically equivalent curve which is parameterized starting at a different point in the interior of the first loop: $\gamma'(t) = \gamma(t - \tau(\text{mod}1))$. As in our other examples, we may also apply a time warping to this this curve: $\gamma'_h(t) = \gamma'(h(t))$ for some orientation preserving homeomorphism $h : [0, 1] \to [0, 1]$. Finally, $\gamma'_h(t)$ may be transformed spatially.

The analysis of these transformed and re-parameterized loops is important in recognizing boundaries of foreground objects in video, for instance. To demonstrate how our algorithm is able to align such curves in the presence of spatial transformations, time warping, and different starting positions, we use an example of a fork object (Figure 19) from the MPEG-7 dataset of 2D contours [30]. To uncover the starting point and parameterization between the two point clouds, we first repeat each loop twice by concatenating the point clouds to themselves. Assuming that one of the point clouds starts T samples later than the other one, there will then be a partial warping path which starts at sample 1 for the first concatenated point cloud and sample T for the second concatenated point cloud which loops around each one once. We can use PIBDTW to uncover this path[8]. Figure 19 shows an example where to forks are successfully put into correspondence this way. Unlike most works which use curvature or other invariants to factor out spatial transformations in quantifying 2D closed contours (e.g. [35], [44]), we are once again able to use more global structure by relying on SSMs.

VII. Conclusions / Future Work

We have shown in this work that it is possible to design end to end curve alignment and cross modal time series synchronization schemes relying solely on self-similarity matrices (SSMs) as an input. As such, we avoid complicated spatial

[8]We found in this example that it was helpful to set σ in the Smith Waterman matching cost to be 0.01 instead of 0.1

alignments, and we avoid computing invariants to spatial transformations which may be too local or lossy. We also provided some theoretical justification for our algorithm, in addition to empirically demonstrating good performance on benchmark datasets. Finally, we made some modifications to our global isometry blind time warping algorithm so that it can be used to synchronize point clouds for which only a partial correspondence is appropriate.

One under-utilized asset of our approach is the fact that since it relies on SSMs, it is coordinate free. This means that the technique can apply to any metric space. While we focused mostly on Euclidean feature spaces in this work, as others have, our algorithms would be uniquely suited to tackle synchronizing time evolving processes that are poorly approximated as Euclidean or for which it is impossible to find a Euclidean embedding. For instance, given a metric between graphs, perhaps we could use this to synchronize evolutionary patterns in social networks.

ACKNOWLEDGMENTS

Christopher Tralie was partially supported by an NSF Graduate Fellowship NSF under grant DGF-1106401 and an NSF big data grant DKA-1447491. Paul Bendich is thanked for the suggestion to compare the technique to the Gromov-Hausdorff Distance, and for helpful comments on an initial manuscript.

REFERENCES

[1] Pankaj K Agarwal, Kyle Fox, Abhinandan Nath, Anastasios Sidiropoulos, and Yusu Wang. Computing the gromov-hausdorff distance for metric trees. In *International Symposium on Algorithms and Computation*, pages 529–540. Springer, 2015.

[2] Naif Alajlan, Ibrahim El Rube, Mohamed S Kamel, and George Freeman. Shape retrieval using triangle-area representation and dynamic space warping. *Pattern Recognition*, 40(7):1911–1920, 2007.

[3] Helmut Alt and Michael Godau. Computing the fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(01n02):75–91, 1995.

[4] Mihael Ankerst, Gabi Kastenmüller, Hans-Peter Kriegel, and Thomas Seidl. 3d shape histograms for similarity search and classification in spatial databases. In *International Symposium on Spatial Databases*, pages 207–226. Springer, 1999.

[5] Juan P. Bello. Grouping recorded music by structural similarity. *Int. Conf. Music Inf. Retrieval (ISMIR-09)*, 2009.

[6] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[7] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

[8] Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. Sparse iterative closest point. In *Computer graphics forum*, volume 32, pages 113–123. Wiley Online Library, 2013.

[9] Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. Generalized multidimensional scaling: a framework for isometry-invariant partial surface matching. *Proceedings of the National Academy of Sciences*, 103(5):1168–1172, 2006.

[10] Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. The video genome. *arXiv preprint arXiv:1003.5320*, 2010.

[11] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

[12] Scott D Cohen and Leonidas J Guibas. Partial matching of planar polylines under similarity transformations. In *SODA*, pages 777–786, 1997.

[13] Ming Cui, John Femiani, Jiuxiang Hu, Peter Wonka, and Anshuman Razdan. Curve matching for open 2d curves. *Pattern Recognition Letters*, 30(1):1–10, 2009.

[14] Ross Cutler and Larry S. Davis. Robust real-time periodic motion detection, analysis, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):781–796, 2000.

[15] Alexander De Luca, Alina Hang, Frederik Brudy, Christian Lindner, and Heinrich Hussmann. Touch me once and i know it's you!: implicit authentication based on touch screen patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 987–996. ACM, 2012.

[16] Thomas Eiter and Heikki Mannila. Computing discrete fréchet distance. Technical report, Citeseer, 1994.

[17] Jonathan Foote. Automatic audio segmentation using a measure of audio novelty. In *Multimedia and Expo, 2000. ICME 2000. 2000 IEEE International Conference on*, volume 1, pages 452–455. IEEE, 2000.

[18] Max Frenkel and Ronen Basri. Curve matching using the fast marching method. In *EMMCVPR*, pages 35–51. Springer, 2003.

[19] Dian Gong and Gerard Medioni. Dynamic manifold warping for view invariant action recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 571–578. IEEE, 2011.

[20] R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. Addison-Wesley world student series. Addison-Wesley, 1992.

[21] Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. *Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, December 2007.

[22] Mikhail Gromov. *Metric structures for Riemannian and non-Riemannian spaces*. Springer Science & Business Media, 2007.

[23] Eugene Hsu, Kari Pulli, and Jovan Popović. Style translation for human motion. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 1082–1089. ACM, 2005.

[24] Imran N Junejo, Emilie Dexter, Ivan Laptev, and Patrick Pérez. Cross-view action recognition from temporal self-similarities. In *Proceedings of the 10th European Conference on Computer Vision: Part II*, pages 293–306. Springer-Verlag, 2008.

[25] Imran N Junejo, Emilie Dexter, Ivan Laptev, and Patrick Perez. View-independent action recognition from temporal self-similarities. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):172–185, 2011.

[26] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

[27] Florian Kaiser and Thomas Sikora. Music structure discovery in popular music using non-negative matrix factorization. In *ISMIR*, pages 429–434, 2010.

[28] Michael Kazhdan. An approximate and efficient method for optimal rotation alignment of 3d models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7), 2007.

[29] Eamonn J Keogh and Michael J Pazzani. Derivative dynamic time warping. In *Proceedings of the 2001 SIAM International Conference on Data Mining*, pages 1–11. SIAM, 2001.

[30] LJ Latecki. Shape data for the mpeg-7 core experiment ce-shape-1. *Web Page: http://www. cis. temple. edu/˜ latecki/TestData/mpeg7shapeB. tar. g z*, 2002.

[31] Pierre Maurel and Guillermo Sapiro. Dynamic shapes average. 2003.

[32] Calvin R Maurer, Rensheng Qi, and Vijay Raghavan. A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):265–270, 2003.

[33] Brian McFee and Daniel PW Ellis. Analyzing song structure with spectral clustering. In *15th International Society for Music Information Retrieval (ISMIR) Conference*, 2014.

[34] Gary McGuire, Nabeel B Azar, and Mark Shelhamer. Recurrence matrices and the preservation of dynamical properties. *Physics Letters A*, 237(1-2):43–47, 1997.

[35] Farzin Mokhtarian and Alan Mackworth. Scale-based description and recognition of planar curves and two-dimensional shapes. *IEEE transactions on pattern analysis and machine intelligence*, (1):34–43, 1986.

[36] Meinard Müller. *Information retrieval for music and motion*, volume 2. Springer, 2007.

[37] Hiroaki Sakoe and Seibi Chiba. A similarity evaluation of speech patterns by dynamic programming. In *Nat. Meeting of Institute of Electronic Communications Engineers of Japan*, page 136, 1970.

[38] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

[39] Thomas B. Sebastian, Philip N. Klein, and Benjamin B. Kimia. On aligning curves. *IEEE transactions on pattern analysis and machine intelligence*, 25(1):116–125, 2003.

[40] Joan Serra, Emilia Gómez, Perfecto Herrera, and Xavier Serra. Chroma binary similarity and local alignment applied to cover song identification. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(6):1138–1151, 2008.

[41] Joan Serra, Meinard Müller, Peter Grosche, and Josep Lluis Arcos. Unsupervised detection of music boundaries by time series structure features. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[42] Temple F Smith and Michael S Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[43] Gineke A ten Holt, Marcel JT Reinders, and EA Hendriks. Multi-dimensional dynamic time warping for gesture recognition. In *Thirteenth annual conference of the Advanced School for Computing and Imaging*, volume 300, 2007.

[44] Ninad Thakoor, Jean Gao, and Sungyong Jung. Hidden markov model-based weighted likelihood discriminant for 2-d shape classification. *IEEE Transactions on Image Processing*, 16(11):2707–2719, 2007.

[45] Christopher John Tralie. *Geometric Multimedia Time Series*. PhD thesis, Duke University Department of Electrical And Computer Engineering, 2017.

[46] George Trigeorgis, Mihalis Nicolaou, Stefanos Zafeiriou, and Bjoern Schuller. Deep canonical time warping for simultaneous alignment and representation learning of sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[47] George Trigeorgis, Mihalis A Nicolaou, Stefanos Zafeiriou, and Björn W Schuller. Deep canonical time warping. 2016.

[48] Mikael Vejdemo-Johansson, Florian T Pokorny, Primoz Skraba, and Danica Kragic. Cohomological learning of periodic motion. *Applicable Algebra in Engineering, Communication and Computing*, 26(1-2):5–26, 2015.

[49] Michael S Waterman. *Introduction to computational biology: maps, sequences and genomes*. CRC Press, 1995.

[50] G Whaba. A least squares estimate of spacecraft attitude. *SIAM Review*, 7(3):409, 1965.

[51] Makoto Yamada, Leonid Sigal, Michalis Raptis, Machiko Toyoda, Yi Chang, and Masashi Sugiyama. Cross-domain matching with squared-loss mutual information. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1764–1776, 2015.

[52] Lijun Yin, Xiaochen Chen, Yi Sun, Tony Worm, and Michael Reale. A high-resolution 3d dynamic facial expression database. In *Automatic Face & Gesture Recognition, 2008. FG'08. 8th IEEE International Conference on*, pages 1–6. IEEE, 2008.

[53] Rex Ying, Jiangwei Pan, Kyle Fox, and Pankaj K. Agarwal. A simple efficient approximation algorithm for dynamic time warping. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '16, pages 21:1–21:10, New York, NY, USA, 2016. ACM.

[54] Chi Wai Yu, KH Kwong, Kin-Hong Lee, and Philip Heng Wai Leong. A smith-waterman systolic cell. In *New Algorithms, Architectures and Applications for Reconfigurable Computing*, pages 291–300. Springer, 2005.

[55] Feng Zhou and Fernando De la Torre. Generalized time warping for multi-modal alignment of human motion. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1282–1289. IEEE, 2012.

[56] Feng Zhou and Fernando De la Torre. Generalized canonical time warping. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):279–294, 2016.

[57] Feng Zhou and Fernando Torre. Canonical time warping for alignment of human behavior. In *Advances in neural information processing systems*, pages 2286–2294, 2009.

Christopher J. Tralie is an algorithmic coder working in applied geometry and geometric signal processing. His work spans shape-based music structure analysis and cover song identification, video analysis, multimodal time series analysis, and geometry-aided data visualization. He received a B.S.E. from Princeton University in EE in 2011 (cum laude), a master's in ECE at Duke University in 2013, and a Ph.D. in ECE at Duke University in 2017. His Ph.D. was primarily supported by an NSF Graduate Fellowship, and his dissertation is entitled "Geometric Multimedia Time Series." He recently began a postdoctoral fellowship at the Information Initiative at Duke, sponsored by the department of mathematics.