Curve Matching, Time Warping, and Light Fields: New Algorithms for Computing Similarity between Curves

Alon Efrat
Department of Computer Science,
University of Arizona

Quanfu Fan
Department of Computer Science
University of Arizona

Suresh Venkatasubramanian
AT&T Labs – Research

Abstract

The problem of curve matching appears in many application domains, like time series analysis, shape matching, speech recognition, and signature verification, among others. Curve matching has been studied extensively by computational geometers, and many measures of similarity have been examined, among them being the Fréchet distance (sometimes referred in folklore as the "dog-man" distance).

A measure that is very closely related to the Fréchet distance but has never been studied in a geometric context is the *Dynamic Time Warping* measure (DTW), first used in the context of speech recognition. This measure is ubiquitous across different domains, a surprising fact because notions of similarity usually vary significantly depending on the application. However, this measure suffers from some drawbacks, most importantly the fact that it is defined between sequences of points rather than curves. Thus, the way in which a curve is sampled to yield such a sequence can dramatically affect the quality of the result. Some attempts have been made to generalize the DTW to continuous domains, but the resulting algorithms have exponential complexity.

In this paper we propose similarity measures that attempt to capture the "spirit" of dynamic time warping while being defined over continuous domains, and present efficient algorithms for computing them. Our formulation leads to a very interesting connection with finding short paths in a *combinatorial manifold* defined on the input chains, and in a deeper sense relates to the way light travels in a medium of variable refractivity.

1 Introduction

The problem of *curve matching* studies ways of measuring the similarity between two curves. Curve matching appears in a variety of different domains; analysis of stock market trends, protein shape matching, speech recognition, computer vision, *etc*. The main questions associated with curve matching in a specific domain are: **(1)** what is a good measure of similarity between curves? **(2)** how can we compute it (or some approximation of it) efficiently? Other questions that are often of interest are: given a database of curves and a candidate curve, can we find a nearest neighbor to this curve in the database? can we *cluster* curves with respect to a given measure of similarity?

Curve matching has been studied extensively in the domain of computational geometry. Here the curves are usually assumed to be represented as polygonal chains in the plane. The measures that have been used to compare them include the Hausdorff distance [ABB95], the *turning curve* distance [ACH$^+$91, CG97], and the Fréchet distance.

Dog-Man Measures The Fréchet distance [AERW03, BBW06, AG95, AKW01, BPRW05, CM04, EIV01, Ven99] has received much attention as a measure of curve similarity. It belongs to a general class of distance

measures that are sometimes called "dog-man" distances. This nickname arises for the following reason. Picture a man and a dog, each of whom is positioned at the start of one of the two given curves. The man holds the (elastic) leash that the dog is tied to. At time 0, the man and the dog start walking on their respective curves towards the respective endpoints. Neither of them can teleport i.e jump from one point to the next, and in most settings (see [EIV01] for an exception), both the man and dog are constrained to move (monotonically) forward along the curve, although they can move at arbitrary speeds relative to each other. We say that a motion of dog and man is *legal* if it satisfies the above constraints. The distance between the two curves is now defined as a function of the leash length (or the leash vector itself), typically minimized over all legal motions. For example, the Fréchet distance itself is the minimum (over all trajectories) of the maximum leash length needed for a fixed trajectory. The general intuition behind the dog-man measures is that since they are defined over continuous parametrizations, they preserve the notion of continuity along the curve and thus are well suited to measuring curve similarity.

Sum Measures and Dynamic Time Warping The Fréchet distance is a max measure; it is defined in terms of the maximum leash length over a parametrization. This dependence on the maximum value can often lead to non-robust behavior, where small variations in the input can distort the distance function by a large amount. Sum- (or average-based) measures are a way of smoothing such distortions, and this motivates our effort. We note that traditionally, it has been harder to compute such sum-measures than max-measures.

One sum-measure that has been in widespread use in various application areas is expressible as a "dog-man" distance. It is known as *Dynamic Time Warping* (DTW), and was first proposed in the 60s as a measure of speech signal similarity [RH93]. Since then, it has been used in a variety of contexts: in databases [CW99, GS00], in computer vision [SB94, MP99, MP03, RM02], in protein structure matching [WHSB98], and in time series clustering and data mining [OFC00, KP99, RK04]. Serra and Berthod [SB94] propose a continuous dynamic time warping technique for subpixel contour matching. Munich and Perona [MP99] explore the use of dynamic time warping as a measure of signature similarity and pose the question of whether continuous versions of these measures can be defined. However, the complexity of their algorithms grows combinatorially and heuristic constraints are needed to make the problem tractable. In the dog-man setting, the DTW distance between two curves (defined as sequences of points) is the *sum* of the leash lengths measured at each (discrete) position (minimized over all trajectories).

The DTW is very easy to compute. Given polygonal curves represented by the sequences of the vertices, of length m and n respectively, a simple dynamic programming algorithm yields the optimal solution in $O(mn)$ time. Given its wide usage in different domains, we would like to use the DTW as a measure of curve similarity. However, the DTW is defined over sequences of points, and is thus not immediately suitable for general curve matching. One could conceivably sample the input curves, and then compute the DTW. However It is not hard to construct examples of curves that are almost identical to each other but may appear quite different under the DTW because of an incorrect sampling of points on each curve (see Figure 1).

1.1 Our Work

Our goal in this paper is to present *continuous* dynamic time warping measures for computing curve similarity.

We describe exact and approximate algorithms for computing the continuous warping distance, and demonstrate their utility for practical applications.

Shortest Paths. The continuous DTW is strongly related to paths on 2D manifolds. We define a *universal* combinatorial manifold in terms of two input polygonal chains: we can represent all possible mappings between two curves as paths on the manifold. We define a class of continuous generalizations of the DTW, and show that

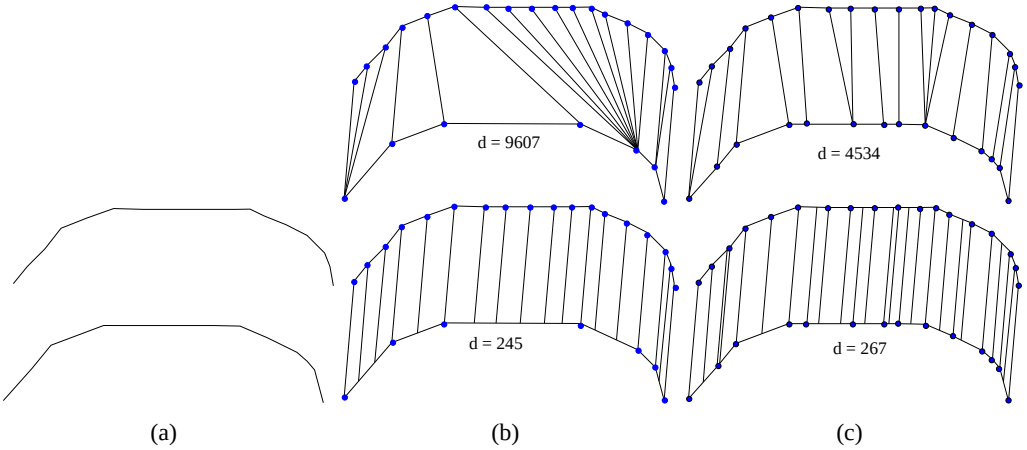


Figure 1: *Matching two curves using the discrete DTW (top) and matching the same curves using Continuous DTW (bottom). (a) shows two curves with a slight difference. (b) and (c) demonstrate the matching results under two different samplings of the curves. The numbers shown in the images are the warping distance of the matching. Note that the discrete DTW can lead to dramatically different results due to the sampling while the continuous DTW is not significantly affected by the sampling.*

computing any of these variants corresponds to computing an appropriately weighted geodesic on the universal manifold. In the special case of a uniform weight function, the problem reduces to computing ordinary shortest paths in a polyhedral surface, and we present an algorithm that computes this in time $O(mn(m+n)\log(m+n))$, where m and n are the number of segments in the two curves. We also present an approximation algorithm that runs in time $O(mnr^2)$, where r is a discretization parameter that controls the approximation ratio.

Light Fields. An interesting aspect of our approach is the connection it draws between dynamic time warping and light paths in a heterogenous medium. This analogy gives us a simple proof of the shortest path result described above, and in general provides an alternate method to computing generalized forms of dynamic time warping between curves.

Paper Outline. We define our continuous time warping measures in Section 3, and characterize them in terms of shortest paths along a combinatorial manifold. In Section 4 we establish the analogy between the continuous measures and light flow. Exact and approximate algorithms are presented in Section 5. We mention in Section 6 possible strategies for solving generalized time warping problems via the use of the *fast march* method of Sethian. Implementation details of our algorithms are described in Section 7. Finally in Section 8 we demonstrate the application of our measures to the problem of *signature verification*.

2 Definitions

We denote vectors as bold faced letters ($\mathbf{a}, \mathbf{b}, \ldots$). A vector $\mathbf{v}$ will have components v_x, v_y[1]. The derivative of a function $f(t)$ with respect to the parameter t will be denoted as $\dot{f}(t)$. We denote the standard Minkowski sum of two point sets A, B as the set $A \oplus B = \{\mathbf{a} + \mathbf{b} \mid \mathbf{a} \in A, \mathbf{b} \in B\}$, and define $A \ominus B = A \oplus (-B)$.

[1] In this paper, all vectors will be in $\mathbb{R}^2$.

Let $T = (t_1, t_2, \ldots t_n)$ be a sequence of time steps. Let $A = (\mathbf{a_i})$ and $B = (\mathbf{b_i})$ be two functions defined over T i.e $\mathbf{a_i} = A(t_i)$ and $\mathbf{b_i} = B(t_i)$. We assume that these are the vertices of polygonal paths, which approximate input curves. The Dynamic Time Warping (DTW) measure d_{DTW} is a distance defined on these functions A and B to be:

$$d_{DTW}(A, B) = \min_{\sigma} \sum_{\sigma_k=(i,j)\in\sigma} \|\mathbf{b_j} - \mathbf{a_i}\| \tag{1}$$

where $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_m)$, $\sigma_i \in [1 \ldots n]^2$ is a sequence satisfying:

Monotonicity: For all $(a, b), (c, d) \in \sigma$ where $a \leq c$, we have $b \leq d$.

Continuity: For $\sigma_i = (x_i, y_i), \sigma_{i+1} = (x_{i+1}, y_{i+1})$, we have $x_{i+1} - x_i \leq 1, y_{i+1} - y_i \leq 1$.

The underlying norm used is typically the ℓ_2 norm, although this is not essential.

A *translation-invariant* version of d_{DTW} [MP99] can be defined as follows. Let $\sigma_k = (i, j)$, and let $\mathbf{v_k} = \mathbf{b_j} - \mathbf{a_i}$. We can rewrite equation 1 as

$$d_{DTW}(A, B) = \min_{\sigma} \sum_{\sigma_k\in\sigma} \|\mathbf{v_k}\|$$

The measure $\tilde{d}_{DTW}$ is now defined as follows:

$$\tilde{d}_{DTW}(A, B) = \min_{\sigma} \sum_{0\leq k<m} \|\mathbf{v_{k+1}} - \mathbf{v_k}\|$$

3 A Universal Manifold

For the purpose of this paper, $\mathbf{A}$ and $\mathbf{B}$ will be polygonal chains. Let a polygonal chain $\mathbf{A} : [0, m] \to \mathbb{R}^2$ be a curve such that for each $i \in \{0, \ldots, m-1\}$, $\mathbf{A}_{|[\mathbf{i},\mathbf{i+1}]}$ is affine, i.e. $\mathbf{A}(\mathbf{i}+\lambda) = (1-\lambda)\mathbf{A}(\mathbf{i}) + \lambda\mathbf{A}(\mathbf{i}+\mathbf{1})$, $0 \leq \lambda \leq 1$. For such a chain $\mathbf{A}$, denote $|\mathbf{A}| = m$. Let $\mathbf{A_i}$ denote the segment $\mathbf{A}_{|[\mathbf{i},\mathbf{i+1}]}$, and $\mathbf{A_i}(\lambda)$ denote the point $(1-\lambda)\mathbf{A}(\mathbf{i}) + \lambda\mathbf{A}(\mathbf{i}+\mathbf{1})$.

We define a combinatorial manifold $\mathcal{M}(\mathbf{A}, \mathbf{B})$. Let $|\mathbf{A}| = m, |\mathbf{B}| = n$. For $1 \leq i \leq m, 1 \leq j \leq n$, let the *patch* $\mathcal{P}_{ij}$ be defined as the set $\mathbf{B_j} \ominus \mathbf{A_i}$ (see Figure 2). Patch $\mathcal{P}_{ij}$ shares edges with the three patches $\mathcal{P}_{i,j+1}$, $\mathcal{P}_{i+1,j}$, $\mathcal{P}_{i-1,j}$, and $\mathcal{P}_{i,j-1}$ (see Figure 3). Since we define a patch as a closed surface, diagonally adjacent patches $\mathcal{P}_{i,j}$ and $\mathcal{P}_{i+1,j+1}$ share a point. It is possible that a patch degenerates into a line (for example if $\mathbf{A_i}$ and $\mathbf{B_j}$ are isometric and parallel); this will not affect the construction of the surface (the reader may realize at this point that the manifold $\mathcal{M}(\mathbf{A}, \mathbf{B})$ is a PL 2-manifold [ACM97]). It is important to remember that this is not an ordinary manifold in that patches may intersect each other and be degenerate; however its local adjacency properties are all that we need for the algorithms we describe.

Each patch has its own local coordinate system. For a point $p = \mathbf{B_j}(\mu) - \mathbf{A_i}(\lambda)$, its coordinates on $\mathcal{P}_{ij}$ are (λ, μ). Thus any curve $\mathcal{C}$ on $\mathcal{P}_{ij}$ defines a parametrization α along the segments $\mathbf{A}_i, \mathbf{B}_j$, and monotonicity of α is equivalent to monotonicity of $\mathcal{C}$.

3.1 A Continuous Measure

We now define a class of continuous time warping measures on curves $\mathbf{A}, \mathbf{B}$. Let $\alpha : [0, m] \to [0, n]$ be a continuous monotone function, and let $\mathbf{v}_\alpha(\mathbf{t}) = \mathbf{B}(\alpha(\mathbf{t})) - \mathbf{A}(\mathbf{t})$. Note that $\mathbf{v}_\alpha(\mathbf{t})$ is a function from $[0, m]$ to $\mathcal{M}(\mathbf{A}, \mathbf{B})$. Let $f(\mathbf{v}) : \mathcal{M}(\mathbf{A}, \mathbf{B}) \to \mathbb{R}$ be a *weight* function. We define the measure $d_f(\mathbf{A}, \mathbf{B})$ as

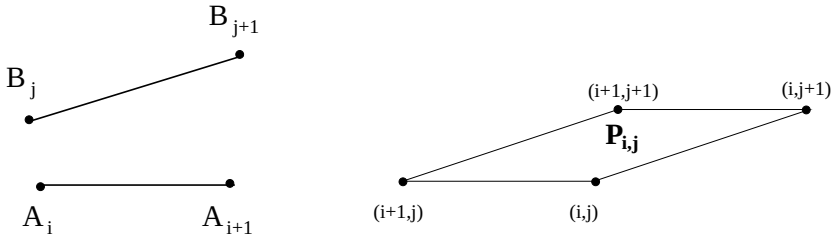


Figure 2: *A pair of segments* $\mathbf{A_i}$ *and* $\mathbf{B_j}$ *and the path* $\mathcal{P}_{i,j}$ *constructed from them. The four vertices of the patch are* $(i,j),(i,j+1),(i+1,j+1)$ *and* $(i+1,j)$.

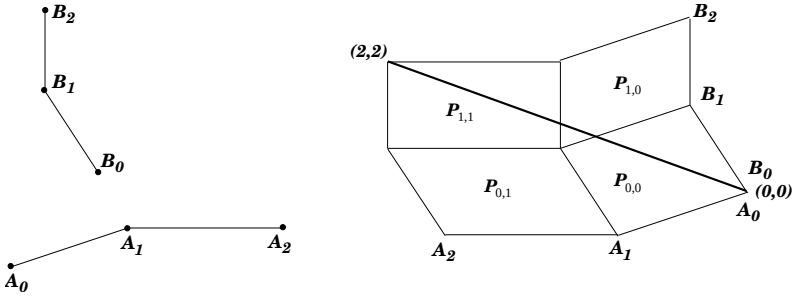


Figure 3: *A pair of curves and the corresponding manifold* $\mathcal{M}(\mathbf{A},\mathbf{B})$. *The bold line is the shortest path from* $(0,0)$ *to* $(2,2)$.

$$d_f(\mathbf{A},\mathbf{B}) = \min_{\alpha} \int_0^m f(\mathbf{v})\ \|\dot{\mathbf{v}}_\alpha(t)\|dt \tag{2}$$

For the special case $f \equiv 1$,

$$d_1(\mathbf{A},\mathbf{B}) = \min_{\alpha} \int_0^m \|\dot{\mathbf{v}}_\alpha(t)\|dt \tag{3}$$

It can be shown that both measures are metrics. The measure d_1 has the additional property of being *translation-invariant*: $d_1(\mathbf{A},\mathbf{B}) = d_1(\mathbf{A}+\mathbf{t},\mathbf{b})$, where t is an arbitrary vector. It can be thought of as the continuous analogue of $\tilde{d}_{DTW}$.

Intuitively, the key term in $d_f(\mathbf{A},\mathbf{B})$ is $\|\dot{\mathbf{v}}_\alpha(t)\|$. As we see in Figure 4, the rate of change of $\mathbf{v}_\alpha(t)$ captures the variation in the parametrization, and the weight function f allows us to penalize this variation non-uniformly.

Let $f(x,y) : \mathbb{R}^2 \to \mathbb{R}$ be a function on the plane, and let $\mathcal{C}$ be a curve in $\mathbb{R}^2$. The integral

$$\int_C f(x,y)ds$$

denotes the path integral of f over $\mathcal{C}$ where ds is the path element along $\mathcal{C}$. If $\mathcal{C}$ is written in terms of a parameter t as $\mathcal{C} = \mathbf{C}(t), t \in [0,m]$, the path element ds along C can be written as $ds = \|\dot{\mathbf{C}}(t)\|dt$ and the above path integral can be written as ([BL86])

$$\int f(C_x(t),C_y(t))\ \|\dot{\mathbf{C}}(t)\|dt$$

Given a parametrization α, the function $\mathbf{v}_\alpha(t)$ is a curve in $\mathcal{M}(\mathbf{A},\mathbf{B})$. Thus, $d_f(\mathbf{A},\mathbf{B})$ can be written as a path integral in $\mathcal{M}(\mathbf{A},\mathbf{B})$.

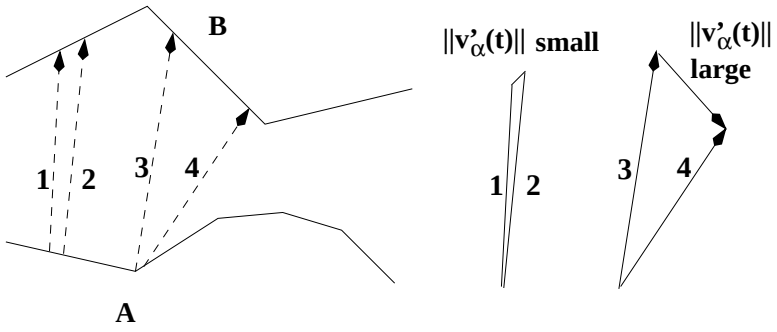


Figure 4: *The rate of change of* $\mathbf{v}_\alpha(\mathbf{t})$

3.2 Finding An Optimal Path

In order to evaluate d_1, we need to determine the nature of the optimal path in $\mathcal{M}(\mathbf{A}, \mathbf{B})$.

Lemma 3.1 *Let* $\chi_i : [0,1] \to A_i$ *be monotone parameterizations of the segments* $\mathbf{A_1}, \mathbf{A_2}$ *such that* $\chi_i(t) = \mathbf{A_i}(0) + (\mathbf{A_i}(1) - \mathbf{A_i}(0)) \int_0^t \alpha_i(u)du$ *where* $\alpha_i(t) \geq 0 \ \forall t \in [0,1]$ *and* $\int_0^1 \alpha_i(u)du = 1$. *Let* $\Phi = \int_0^1 \psi(\frac{d}{dt}(\chi_2(t) - \chi_1(t)))dt$ *where* $\psi : \mathbb{R}^2 \to \mathbb{R}$ *is a convex function. Then* Φ *is minimized when* $\alpha_1(t) = \alpha_2(t) = t$.

Proof: Since $\frac{d}{dt}\chi_i(t) = \alpha_i(t)[\mathbf{A_i}(1) - \mathbf{A_i}(0)]$, we have

$$\begin{aligned}
\Phi &= \int_0^1 \psi(\alpha_2(t)\mathbf{A_2}(1) - \mathbf{A_2}(0) - \alpha_1(t)\mathbf{A_1}(1) - \mathbf{A_1}(0))dt \\
&\geq \psi(\int_0^1 \alpha_2(t)(\mathbf{A_2}(1) - \mathbf{A_2}(0)) - \alpha_1(t)(\mathbf{A_1}(1) - \mathbf{A_1}(0))dt) \\
&= \psi((\mathbf{A_2}(1) - \mathbf{A_2}(0)) - (\mathbf{A_1}(1) - \mathbf{A_1}(0)))
\end{aligned}$$

where the inequality follows from Jensen's inequality. We can attain this minimum value by setting $\alpha_i(t) = t$. ■

The norm $\|\cdot\|$ is convex, and thus the conditions of Lemma 3.1 apply. Thus the optimal path is the uniform parametrization. Substituting α in Equation 3 yields the following result.

Lemma 3.2 *If* $\mathbf{A}, \mathbf{B}$ *consist of single segments, then*

$$d_1(\mathbf{A}, \mathbf{B}) = \|(\mathbf{B}(1) - \mathbf{A}(1)) - (\mathbf{B}(0) - \mathbf{A}(0))\|$$

But this is merely the shortest Euclidean distance between the points $(0,0)$ and $(1,1)$ on $\mathcal{M}(\mathbf{A}, \mathbf{B})$. Extending this over all patches, this yields the following theorem.

Theorem 3.1 $d_1(\mathbf{A}, \mathbf{B})$ *is equal to the length of the shortest monotone path in* $\mathcal{M}(\mathbf{A}, \mathbf{B})$ *from the point* $(0,0)$ *on* $\mathcal{P}_{0,0}$ *to the point* (m,n) *on* $\mathcal{P}_{m,n}$.

Proof: Define a path π on $\mathcal{M}$ to be *good* if it is monotone, connects the point $(0,0)$ on $\mathcal{P}_{0,0}$ to the point (m,n) on $\mathcal{P}_{m,n}$, and its intersection with every patch of $\mathcal{M}$ is, if not empty, consisting of a single edge. Note that every good path corresponds to a matching between **A** and **B**, (i.e., the function $\alpha()$ used in equation 3), and the const of the corresponding matching is, according to Lemma 3.1, equals to the length of π. In particular, there is path π^* that corresponds to the optimum matching. Clearly this is exactly the shortest good path. ■

4 Light Fields and the Eikonal Equation

In general, the form of the function f will determine our ability to compute $d_f(\mathbf{A},\mathbf{B})$. Traditional variational methods can be employed to solve the functional [BM91], but it is typically hard to do this for arbitrary functions.

However, there is an alternative formulation of the above path integral. Consider a two-dimensional refractive medium whose refractive index at the point (x,y) is given by the function $f(x,y)$. Then the path $\mathcal{S}$ that a light beam will take to go from a point **a** to a point **b** in this medium is the curve $\mathcal{S}$ from **a** to **b** that minimizes the path integral[2] [BW80, §3.1]

$$\int_{\mathcal{S}} f(x,y)ds$$

Lemma 4.1 *Let* $\mathbf{A_i}$ *and* $\mathbf{B_j}$ *be two segments, and let* $\mathcal{P}_{ij}$ *be the associated patch. Assume that the path that light takes to travel from* (i,j) *to* $(i+h,j+k)$, $(h,k>0)$ *in* $\mathcal{P}_{ij}$ *is monotone with respect to the patch. Then* $d_f(\mathbf{A_i},\mathbf{B_j})$ *is equal to the length of the path that a light beam will take from* $(0,0)$ *to* $(1,1)$ *in a medium isometric to* $\mathcal{P}_{ij}$*, where the refractive index of any point* (x,y) *is* $f(x,y)$.

Light flow in nonhomogenous fields has been studied extensively [BW80], and working forwards from Maxwell's electromagnetic equations, the following equation can be derived [BW80, §3.1.1]:

$$\|\nabla\mathcal{F}\| = f \tag{4}$$

This equation is called the *eikonal* equation, and the function $\mathcal{F}$ is called the eikonal[3]. For isotropic media, $\mathcal{F}$ can be thought of as describing the geometric wavefronts that propagate the light rays; more formally, if **s** is a unit vector in the direction of the trajectory of a ray of light in this field, then $\mathbf{s} = \nabla\mathcal{F}/f$.

This formulation provides an alternate proof for Lemma 3.1 that does not rely directly on the calculus of variations. Setting $f \equiv 1$, Equation 4 can be written as

$$\|\nabla\mathcal{F}\| = 1$$

The solution to this equation consists of the family of curves

$$t^2 + u^2 = \text{const}$$

In other words, the curves orthogonal to the light paths are circles around the origin, implying that the light paths are straight lines emanating from the origin, hence in the optimal solution $\alpha_1(t) = \alpha_2(t) = t$, as already shown in Lemma 3.1.

[2] This is a version of Fermat's principle of the shortest optical length.

[3] According to [BW80], the word comes from the Greek word for 'image'.

5 Algorithms

In this section we discuss exact and approximation algorithms for computing $d_1(\mathbf{A}, \mathbf{B})$. For clarity, we refer to the traditional DTW method as discrete DTW (DDTW) and our continous measure as CDTW.

5.1 An exact algorithm

Mitchell, Mount and Papadimitriou [MMP87] presented an algorithm running in $O(k^2 \log k)$ time to compute s-t shortest paths on a general polyhedral surface with k triangles. This was subsequently improved by Chen and Han [CH96] to $O(k^2)$, and recently, a result by Kapoor [Kap99] showed a bound of $O(k \log^2 k)$. It is easy to verify that the algorithm of [MMP87] can be modified to compute a **monotone** shortest path. Also note that by the unfolding property of shortest paths, we only need the combinatorial structure of the polytope and the geometry of each face. Let $|A| = m$ and $|B| = n$. There are $O(mn)$ faces in the instance of the shortest path problem that we construct, and hence a naive bound on the running time of the algorithm of [MMP87] is $O(m^2n^2)$. A more careful look at the algorithm reveals that the actual running time is $O(Q \log Q)$, where Q is the total number of subintervals of edges of triangles of the manifold. These subintervals are obtained by splitting edges based on the combinatorial structure of shortest paths reaching points of this edge. We show that in our setting Q is only $O(mn(m+n))$. Thus the running time of the algorithm of Mitchell et al. [MMP87] in our setting is only $Q = O(mn(m+n)\log(mn))$.

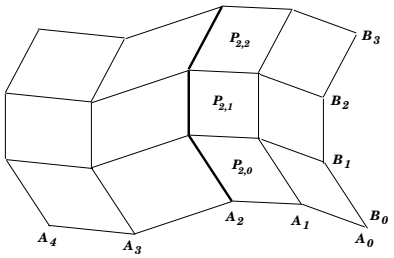


Figure 5: A vertical wall ($i = 2$) is shown as a bold line on the manifold $\mathcal{M}(\mathbf{A}, \mathbf{B})$. The wall is composed of the left edges of patch $\mathcal{P}_{2,0}$, $\mathcal{P}_{2,1}$, and $\mathcal{P}_{2,2}$.

Bounding Q. We define the *vertical wall* of the patch $\mathcal{P}_{i,j}$ to be the edge the patch shares with $\mathcal{P}_{i+1,j}$. We define the i-vertical wall ℓ_i of $\mathcal{M}(\mathbf{A}, \mathbf{B})$ to be the union of the left vertical wall of all pathes $\mathcal{P}_{i,j}$ (for $j = 0, \ldots n-1$). This is a connected polygonal path. See Figure 5 for an illustration. Fix $1 \le i \le n$ and $x \in \ell_i$, and consider the monotone shortest path from the origin to x. Let $\sigma(x)$ be the sequence of patches of $\mathcal{M}(\mathbf{A}, \mathbf{B})$ that this path meets, in the order that they appear along the path. We divide ℓ_i into maximally connected intervals that form equivalence classes for $\sigma(x)$, i.e into subintervals such that for all points x in a subinterval, $\sigma(x)$ is fixed.

Let $y_1, y_2, \ldots y_k$ be points along ℓ_i that have pairwise distinct patch sequences $\sigma(y_r)$ ($1 \le r \le k$). Note that to specify $\sigma(y_r)$, it suffices to specify above which corners of cells of $\{\mathcal{P}_{ij}\}$, $\sigma(y_r)$ pass, and below which corners. Observe (as in [MMP87]) that $\sigma(y_r)$ is disjoint to $\sigma(y_{r+1})$, and thus there must be a corner of a patch $\mathcal{P}_{i',j'}$ that $\sigma(y_r)$ passes below, and $\sigma(y_{r+1})$ passes above. Thus there must be at least k corners of cells of $\{\mathcal{P}_{ij}\}$ below $\sigma(y_k)$, implying that $k \le mn$. Summing this numbers for all m vertical walls $\{\ell_1 \ldots \ell_{m-2}\}$, this bound implies that the total number of maximally connected intervals is $O(mn(m+n))$. Since the running time of the algorithm is directly proportional to this quantity, we obtain the result (exchanging the rules of m and n if needed)

Theorem 5.1 *Given two polygonal chains* $\mathbf{A}, \mathbf{B}$*, where* $|\mathbf{A}| = m, |\mathbf{B}| = n$*, and* $m < n$ *we can compute* $d_1(\mathbf{A}, \mathbf{B})$ *in time* $O(nm^2 \log(n))$.

5.2 An Approximation Algorithm

The exact algorithm described above is slow in practice, mainly due to problems in scaling existing code ([KO00]) for computing shortest path on a terrain. It is not clear whether a careful implementation of the algorithm would resolve this problem. Thus, in this section we present a fast algorithm that runs in time linear in $m \cdot n$ and approximates d_1 to any desired factor.

Lanthier et al. [LMS97] compute an approximate weighted geodesic shortest path between two points on a polyhedron surface P by adding additional vertices (known as Steiner points) on the edges of P, breaking them into shorter edges. These points are connected via straight segments, and the Dijkstra's algorithm [Dij59] is then used to find the shortest path. Here, we employ a similar idea of adding Steiner points to our problem. However, the monotonicity requirement restricts the set of paths that we need to consider, and allows us to compute an approximat shortest path by using dynamic programming, which is much easier to implement and is faster than Dijkstra's method by a factor of $\log(mn)$.

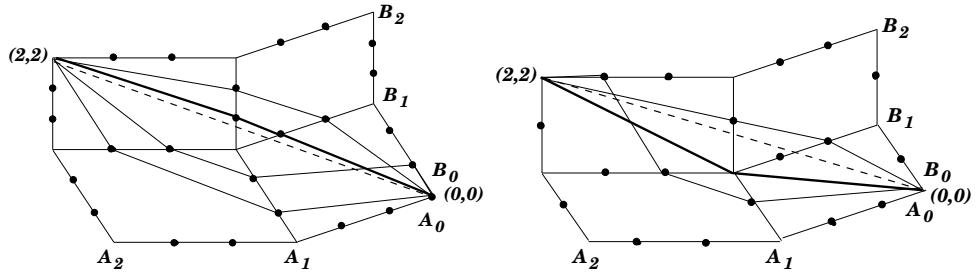


Figure 6: *Steiner points and* Γ*-paths on a manifold. a) UNIFORM placement scheme and b) LENGTH placement scheme. The dashed line is the optimal path* d_1 *and the bold line is the shortest* Γ*-path that approximates* d_1.

We start by placing Steiner points on the edges of the manifold $\mathcal{M}(\mathbf{A}, \mathbf{B})$ (See Figure 6). Let S be the set of Steiner points and $\Gamma = S \cup \mathbf{A} \cup \mathbf{B}$. A geodesic path is called a Γ-*path* if it is monotone and contains only vertices from Γ. Let $\Gamma(p)$ be a Γ-path from the origin $(0, 0)$ of $\mathcal{M}(\mathbf{A}, \mathbf{B})$ to a point $p \in \Gamma$. Observe that d_1 can now be approximated by the shortest Γ-path to the point (m, n) on $\mathcal{M}(\mathbf{A}, \mathbf{B})$. Let $\mathcal{P}$ be a patch of $\mathcal{M}(\mathbf{A}, \mathbf{B})$. We denote the four edges of $\mathcal{P}$ (left, right, bottom and top) as e_l, e_r, e_b, and e_t respectively (See Figure 7). We also denote the i^{th} point on an edge e_x ($x \in \{l, r, b, f\}$) as p_x^i. Monotonicity forces any Γ-path of a point p on e_l or e_t to pass through e_b or e_r. Thus if the shortest Γ-paths to all the points on e_r and e_b are computed, the shortest Γ-path to p can be easily found by considering all the possible paths from the points on e_b and e_r (See Fig 7(a)). Specifically, the shortest Γ-path to p can be computed as,

$$d_{min}(\Gamma(p)) = \arg\min_{p'}\{d_{min}(\Gamma(p')) + \overline{|pp'|}\} \tag{5}$$

where $p' \in e_b \cup e_r$ and $\overline{pp'}$ is locally monotone.

Dynamic programming can be used to compute the shortest Γ-paths to all the points on $\mathcal{M}(\mathbf{A}, \mathbf{B})$. The pseudo code is presented in Algorithm 1.

Algorithm 1 *Compute the CDTW distance between two curves* **A** *and* **B**

Require: Curve $\mathbf{A} = \{a_1, a_2, \ldots, a_n\}$
Require: Curve $\mathbf{B} = \{b_1, b_2, \ldots, a_m\}$
Require: s: Number of Steiner points per edge
  Construct the manifold $\mathcal{M}(\mathbf{A}, \mathbf{B})$;
  Place Steiner points on $\mathcal{M}(\mathbf{A}, \mathbf{B})$;
  for $i = 2$ to n **do**
    for $j = 1$ to m **do**
      Set values for points on the bottom and right edges of the patch $\mathcal{P}_{ij}$;
      for $k = 1$ to $s + 1$ **do**
        $d(p_l^k) = \min(\arg\min_{1 \le k' \le s+1}\{d(p_b^{k'}) + \overline{|p_l^k p_b^{k'}|}\}, \arg\min_{1 \le k' \le k}\{d(p_r^{k'}) + \overline{|p_l^k p_r^{k'}|}\})$
      end for
      for $k = 1$ to $s + 1$ **do**
        $d(p_t^k) = \min(\arg\min_{1 \le k' \le s+1}\{d(p_r^{k'}) + \overline{|p_t^k p_r^{k'}|}\}, \arg\min_{1 \le k' \le k}\{d(p_b^{k'}) + \overline{|p_t^k p_b^{k'}|}\})$
      end for
    end for
  end for
  return $d(p_t^{s+1})$ computed over the patch $\mathcal{P}_{mn}$

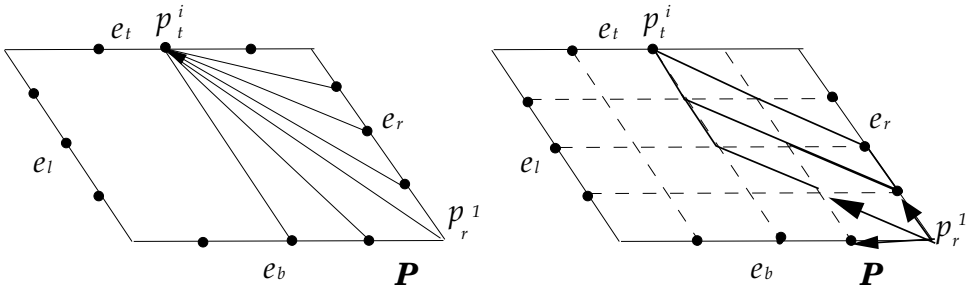


Figure 7: *(a) Possible paths to a point on the top edge of a patch $\mathcal{P}$ in the CDTW approach. (b) Possible paths to a point on the top edge of a patch $\mathcal{P}$ in the DDTW approach*

Lemma 5.1 *Given two polygonal chains* $\mathbf{A}$ *and* $\mathbf{B}$ *where* $|\mathbf{A}| = m$ *and* $|\mathbf{B}| = n$*, then* $d_1(\mathbf{A}, \mathbf{B})$ *can be approximated by a monotone shortest path in* $\mathcal{M}(\mathbf{A}, \mathbf{B})$ *and computed in* $O(r^2mn)$ *time where* r *is the number of Steiner points added on each edge in* $\mathcal{M}(\mathbf{A}, \mathbf{B})$*. The absolute difference in cost between the approximate and optimal paths is at most* $L(m + n)/r$*, where* L *is the length of the longest segment in either of the two chains.*

Proof: For each patch, we need to compute the shortest Γ paths for $2(r+1)$ points and for each point, we spend $O(r)$ time. Hence the time spent on each patch is $O(r^2)$. We have a total of $m \cdot n$ patches, so the running time bound holds. The approximation achieved by the algorithm follows the proof in [LMS97] directly. ■

The approximate algorithm proposed above has the same running time as the discrete DTW algorithm when the same number of Steiner points are added to the input segments.

Remark 5.1 *It is important to mention that although our approach is based on discretization, it discretizes the* manifold*, rather than the input segments. The discrete* DTW *approach would never yield an arbitrarily close approximation of the shortest path if the* discretization *or* resampling *is not done appropriately. Figure 7 illustrates such an example. The discrete approach restricts the warping paths to only three "directions": left, up and diagonal, so no path can go from* p_r^1 *to* p_t^i *directly, which is the shortest path from* p_r^1 *to* p_t^i *on the patch* $\mathcal{P}$*. Simply adding more points on the edges would not improve the approximation. In contrast, our continuous measure only requires that the warping paths are monotone. As shown by Lemma 5.1, as long as there are enough Steiner points added on the manifold, the approximation can be made arbitrarily close to the true answer.*

6 $d_f(\mathbf{A}, \mathbf{B})$ And The Fast-March Algorithm

Solutions to the eikonal equation can in general be non-differentiable (an easy example is the case of light travelling through an interface between media of different refractivity), and thus computing closed form solutions analytically can be hard in all but special cases. In addition, issues of numerical accuracy need to be addressed very carefully. Among the algorithmic solutions for weighted shortest path problems, the methods of [ALMS00] express running terms in terms of numerical properties of the input (such as the ratio of the longest edge to the shortest edge etc).

In this section, we discuss a technique first proposed by Sethian [Set99], and applied by Kimmel and Sethian [KS98] to solve the eikonal equation numerically in the context of computing weighted shortest paths[4]. The advantage of this method (called the *fast marching method*) are that the solution it provides converges monotonically to the exact answer as the error parameter tends to zero, and thus is a provably correct approximation scheme. In addition, the method itself works by imposing a uniform grid (or triangulation) on the surface, and runs in time independent of numerical parameters of the surface.

The skeleton of the algorithm is a grid update procedure that expands out from the starting point in a fashion much like the Dijkstra [Dij59] shortest path algorithm on graphs. At each stage, the unvisited grid point with the smallest weight is visited, and all its neighbours are updated according the values at their neigbours.

The crucial difference between this technique and a standard Dijkstra-type algorithm is in the step where weights are updated. This issue is discussed at length in [Set99, §8.6.1]; the basic idea is that by using an update mechanism based on a second order finite difference operator, the method guarantees that the new distance computed is a smooth interpolation of the values inside a grid cell.

[4]This technique has been applied to a wide class of geometric problems, like motion planning, surface reconstruction, and computing shape differences. The interested reader is referred to [Set99] for further details.

The algorithm is extremely easy to implement, requiring only standard heap data structures, and runs in time $O(mn/\varepsilon^2)$, where ε is the length of the subdivision of each segment. Note that unlike other algorithms that make use of these so-called *Steiner* points, no careful placement of points is required. The field $\tilde{\mathcal{F}}$ computed can be shown to converge monotonically to the exact solution as the number of grid points increase, and the process is second order convergent. The error in the computation (for example the ℓ_2 distance between the approximate function $\tilde{\mathcal{F}}$ and the exact function $\mathcal{F}$) is linear in the length of the longest edge on the manifold.

7 Implementations

In order to implement the exact algorithm of Section 5.1, we chose to utilize Kaneva's implementation ([KO00]) of Chen and Han's shortest path algorithm ([CH96]). This is one of the few publicly released programs for computing shortest paths on polyhedra. We construct the combinatorial manifold $\mathcal{M}(\mathbf{A}, \mathbf{B})$ discussed in Section 3 for the curves A and B and use Kaneva's program to find the continuous dynamic time warping path (i.e the shortest path) on $\mathcal{M}(\mathbf{A}, \mathbf{B})$. This path is then used to match the two curves **A** and **B**. Figure 1 illustrates such a matching. Unfortunately, this implementation runs very slowly when the number of faces of the polyhedron exceeds a few thousands, and it was impractical to use it for large data sets. We therefore implement the approximate algorithm described in 5.2 and use it to conduct all the experiments below.

The approximate algorithm relies on placing Steiner points on the edges of the manifold. To see how the placement of Steiner points affects the approximation, we experimented with two schemes of placing Steiner points. One scheme assigns the same number of points to each edge and the other determines the number of points to be placed for an edge by its length. In both schemes, steiner points are uniformly spaced on edges. We call the former "UNIFORM" and the latter "LENGTH" (See Figure 6). We also include the discrete DTW algorithm with resampling for comparison.

We randomly generated a small data set of 50 curves. Each curve has 20 points and the maximum length of a segment on the curve is 50. For each pair of curves, we computed the approximate warping distance by using different average number of Steiner points per edge and compared it to the optimal distance. The optimal distance is approximated by the warping distance computed from the continuous DTW algorithm using 500 Steiner points per edge with the "LENGTH" placement scheme. Figure 8 shows the average relative difference between the approximate and the optimal warping distance for the two placement schemes. Note that the continuous measure can approximate the optimal distance much better than the discrete measure under both schemes. The continuous measure approaches the optimal distance slightly more quickly under the "UNIFORM" scheme than the "LENGTH" scheme. It is likely that the bound we obtain can be improved slightly using an expander technique as mentioned in the work of Lanthier et al. [LMS97].

The continuous measure can yield a very good approximation of the optimal warping path (above 99%) by using 5 Steiner points per edge. Unless specified, this number is used for both the continuous and discrete DTW algorithms in the experiments that follow.

8 Signature Verification

Dynamic time warping has been widely used in signature verification [MP99, MP03]. However, the quality of the DDTW measure relies heavily on the sampling of signatures. In practice, inappropriate or even poor sampling of signatures may occur from time to time for various reasons such as the use of different tracking devices or the change of signing behaviors (for example, signing heavily or softly or emotionally) of the signer. A possible solution for this problem is to resample the curves/signatures (for example, using spline interpola-

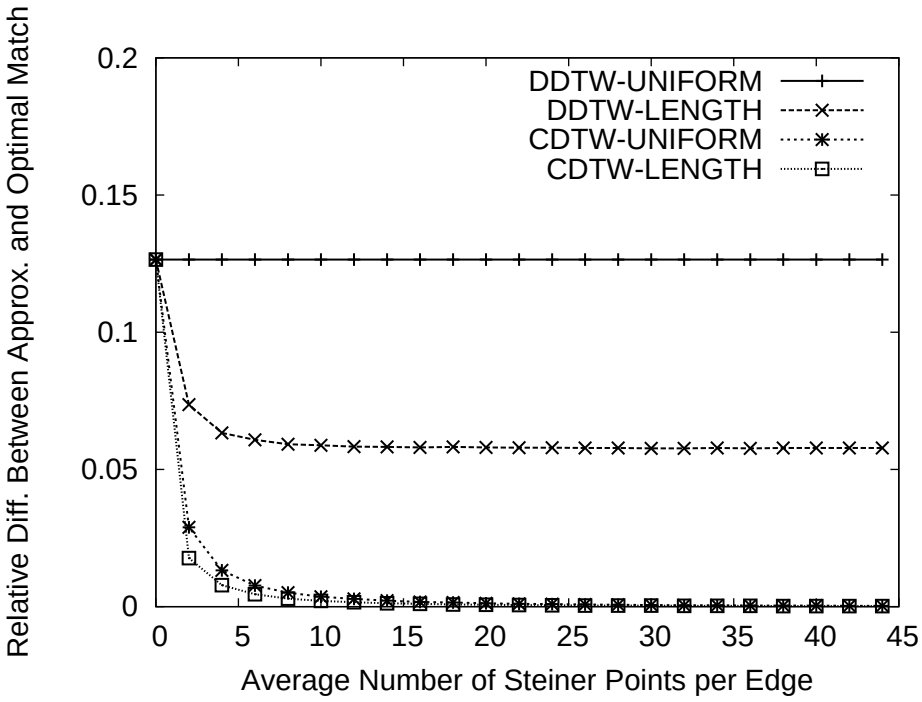


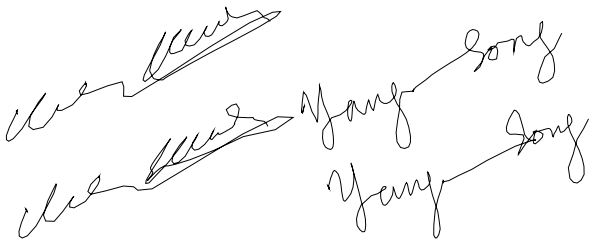
Figure 8: *Relative difference between the approximate and optimal matching of curves under the "UNIFORM" and "LENGTH" placement schemes.*

tion) before matching them. However, there is no general principle regarding how the resampling should be done to achieve desired results. Unlike DDTW, the CDTW measure can truly capture the similarity between two signatures regardless of the way of sampling, thus making itself a more robust and reliable measure for signature verification. As we demonstrate later, the continuous measure yields much more consistent performance than DDTW when the signatures are insufficiently sampled. We first briefly describe the data sets we use in the experiments.

Signature Database. The database was collected by Munich [Mun] and includes two sets of signatures captured by a camera-based tracking system. The first set consists of signatures from 56 subjects and each subject provides 25 signatures. The second set consists of signatures from 50 subjects and each subject provides 30 signatures. In addition, both sets include 10 skilled forgeries for each subject. Readers are referred to [Mun] for details.

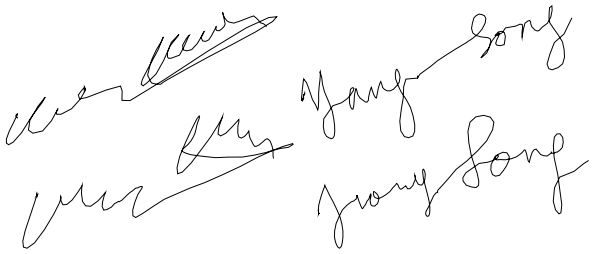
Figures 9−11 show some samples from the database. Table 1 summarizes the results of using our measure to compute the distance between the samples. There is a significant difference between the distance between similar signatures and the distance between eitherandomly chosen signatures or a signature and its forged version. Although these are only a few examples, they demonstrate that the continuous time warping measure effectively captures an intuitive notion of curve similarity. We now evaluate the performance of this measure in detail.

Data Simplification. Our purpose is to compare the quality of the DDTW and the CDTW measures on insufficiently sampled data. We approximate the signatures in the database with fewer points through curve simplification. Such signatures are called *simplified signatures*. There are many efficient and effective approaches proposed for simplifying curves or approximating time series [AV00, KCHP01]. As our focus here is to demonstrate the quality of our measure for data insufficiently sampled, we simply choose the Douglas-



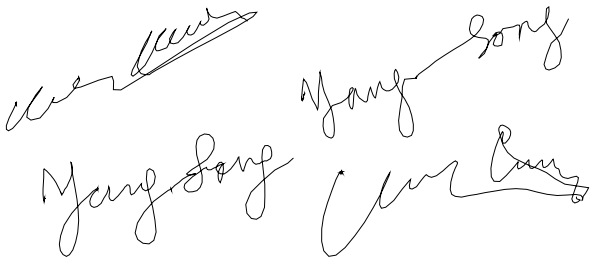
(a) (b)

Figure 9: *Two examples of signature data. In each picture, two signatures made by the same person are depicted.*



(a) (b)

Figure 10: *The genuine signatures compared with forgeries.*



(a) (b)

Figure 11: *The genuine signatures compared with random signatures.*

Pair being compared	Num. Steiner points per edge		
	0	5	10
Same (Figure 9(a))	119.41	87.96	81.163
Forgery (Figure 10(a))	218.67	136.85	131.54
Different (Figure 11(a))	413.08	313.86	306.09
Same (Figure 9(b))	131.52	98.62	94.56
Forgery (Figure 10(b))	213.42	160.27	157.52
Different (Figure 11(b))	315.07	266.50	264.61

Table 1: *Approximation of of the optimal distance* d_1 *between pairs.*

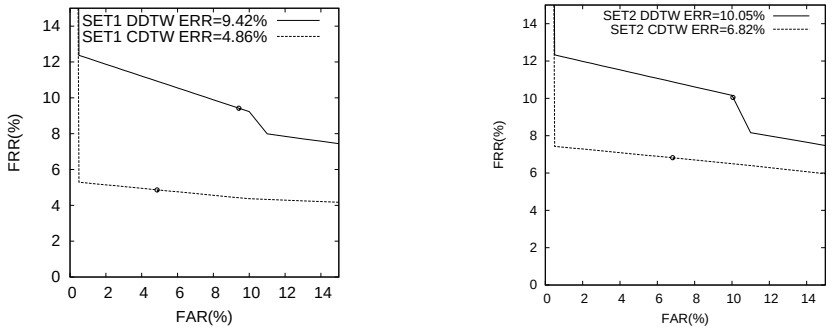
Forgery	Skilled				Random			
Algorithm	DDTW		CDTW		DDTW		CDTW	
Data	Set 1	Set 2	Set 1	Set 2	Set 1	Set 2	Set 1	Set 2
Original Sig.	6.1	8.8	5.3	8.0	1.6	3.0	1.6	2.8
Simplified Sig.	6.5	9.2	5.8	8.2	1.7	3.3	1.6	2.9

Table 2: Equal Error Rate *(%) computed from the original signatures and the simplified signatures.* 5 *Steiner points per edge are used in both algorithms.*

Peuker algorithm [DP73], one of the most popular curve simplification algorithms. Note that the simplification also brings other extra benefits such as eliminating small discontinuities or movement introduced by the measurement and saving storage that could be a critical issue for some verification systems with a lot of users. This technique is also often used when indexing very large time series database.[KCHP01, PKC01]. In our experiments, when a tolerance of 0.2 is used, the simplification reduces the number of points for a signature by $30\% - 50\%$ and only slightly changes the shape of the signature. (The minimum width and height of the signatures in the database are 22 and 14, respectively. The average width and height are 82 and 54.).

Training and Testing. We choose the first 10 true signatures of each person as the training data. Similar to [MP03], we do pairwise alignments to pick a *reference signature* that yields the minimum average alignment cost with the other signatures. We consider both *random* and *skilled forgeries* in our experiments. A *random forgery* is a signature from a subject other than the subject that the signature to be verified belongs to. The database can provide each subject up to $2,000$ random forgeries, but only 10 skilled forgeries.

When evaluating the performance of a system, one indicator that is often used in signature verification literatures is the *error tradeoff curve* (See Figure 12). This parameter depicts the false acceptance rate *(FAR)* as a function of the false rejection rate *(FRR)*. Here, FAR measures the number of forgeries being accepted as genuine ones by the system while FRR refers to the number of genuine signatures being recognized as forgeries. The error tradeoff curve is traditionally characterized by its *equal error rate* (EER), the error rate at which the FAR is equal to FRR. A lower EER represents better performance.



(a) (b)

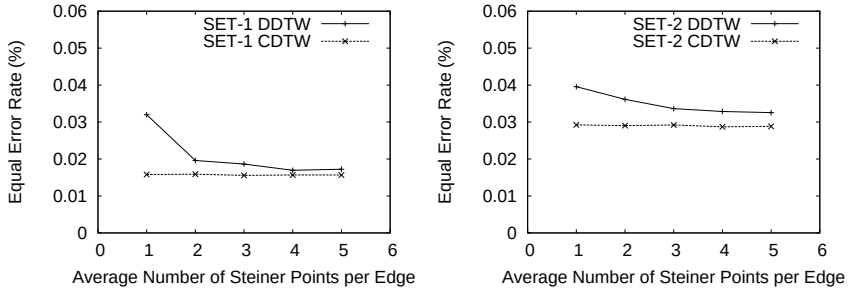
Figure 12: *Error tradeoff curves for DDTW and CDTW using skilled forgeries. The EERs are marked as circles in the figure.*

Table 2 shows the EERs computed from the original signatures and the simplified signatures by using 5 Steiner points per edge. As we can see, the CDTW measure performs only slightly better than the DDTW measure on the original data, which suggests that the signatures in the original data are well sampled. As expected, the skilled forgeries are shown to be more difficult to be verified than the random ones in both of

the two measures. In addition, both measures yield similar results on the original and the simplified data, demonstrating that signature simplification is a feasible way in practice.

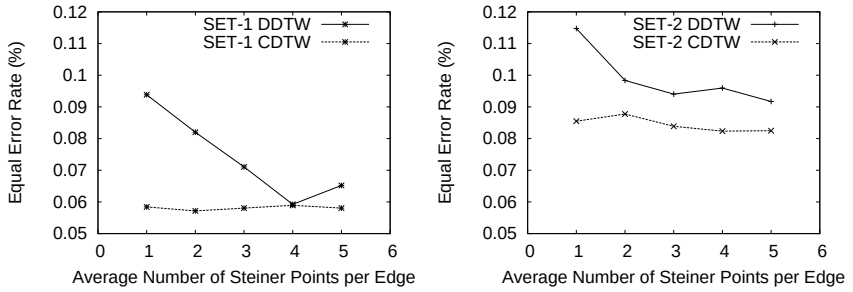
Figure 13 and Figure 14 show how the EERs of using random and skilled forgeries vary with the number of Steiner points per edge added in the DDTW and CDTW algorithms. We observe that although the performance difference between DDTW and CDTW decreases as the number of points per edge added increases, CDTW demonstrates more consistent results than DDTW. This indicates that the CDTW measure is less sensitive to the sampling of the signatures and hence a more robust measure in practice.

We have thus demonstrated that the CDTW is a more robust measure for insufficiently sampled data than the traditional DTW measure. In addition, it allows us to use simplified *reference signatures* for signature verification, which are favored by some verification systems in which storage is a critical resource.



(a) (b)

Figure 13: *EERs under different number of Steiner points used (Random forgery)*



(a) (b)

Figure 14: *EERs under different number of Steiner points used (Skilled forgery)*

9 Conclusions

Our goal in this paper was to study measures of curve similarity in the "dog-man" setting. Our study of "continuous" dynamic time warping was motivated by the drawback of other methods (e.g. Fréchet distance and dynamic time warping) and the resulting formulations in terms of weighted shortest paths demonstrate that sum-based measures can have rich structure, as well as efficient algorithms.

The general nature of $\mathrm{d_f}(\mathbf{A}, \mathbf{B})$ (as modulated by the weight function f), suggests that this measure might be utilized in a wide range of applications. We have shown the quality of this measure in signature verification.

One interesting future direction would be to further evaluate the performance of the CDTW measure on other data sets such as handwritten recognition data [RM03],

Another promising direction is in the context of protein backbone matching, in which we wish to compare the similarity of proteins represented as chains (in three dimensions) of carbon atoms (the C_α atoms). It would be interesting to explore the efficacy of this measure in capturing the similarity of such backbones.

10 Acknowledgements

We would like to thank Jim Reeds, Jeff Lagarias, Shankar Krishnan, Pankaj Agarwal and Joe Mitchell for many discussions on the topic of curve matching measures, vector calculus and differential geometry. We also would like to thanks the anonymous reviewers for very fruit ful advices.

References

[AV00] P. K. Agarwal, K. Varadarajan. Efficient algorithms for approximating polygonal chains *Discrete Comput. Geom.* 23 (2000), 273–291.

[ABB95] H. Alt, B. Behrends, and J. Blömer. Approximate matching of polygonal shapes. *Ann. Math. Artif. Intell.*, 13:251–266, 1995.

[BBW06] K. Buchin, M. Buchin and C. Wenk, Computing the Frechet Distance Between Simple Polygons in Polynomial Time, *Proc. 22th ACM Symp. on Computational Geometry (SoCG)*, 80–88, 2006.

[AG95] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International J. of Computational Geometry and Applications*, 5:75–91, 1995.

[AKW01] H. Alt, C. Knauer, and C. Wenk. Matching polygonal curves with respect to the Fréchet distance. In *Symp. on Theoretical Aspects of Computer Science (STACS)*, 63–74, 2001.

[AERW03] H. Alt, A. Efrat, G. Rote and C. Wenk. matching planar maps, *J. Alg.* , 49 (2003) 262–283.

[ALMS00] L. Aleksandrov, M. Lanthier, A. Maheshwari, and J.-R. Sack. Approximation algorithms for geometric shortest path problems. In *32nd ACM Symposium on Theory of Computing*, 2000.

[ACM97] J. Ambjørn, M. Carfora, and A. Marzuoli. *The Geometry of Dynamical Triangulations*, volume m50 of *Lecture Notes in Physics*. Springer-Verlag, 1997.

[ACH+91] E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and Joseph S. B. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(3):209–216, 1991.

[BL86] P. Baxandall and H. Liebeck. *Vector Calculus*. Oxford University Press, 1986.

[BM91] U. Brechtken-Manderschied. *Introduction to the Calculus of Variations*. Chapman and Hall, 1991.

[BW80] M. Born and E. Wolf. *Principles of Optics: Electromagnetic theory of propagation, interference and diffraction of light*. Cambridge University Press, 6th edition, 1980.

[BPRW05] S. Brakatsoulas, D. Pfoser, R. Salas and C. Wenk, On Map-Matching Vehicle Tracking Data, in *Very Large Data Bases (VLDB)*, 853–864, 2005.

[CM04] M. Clausen, A. Mosig, Approximately Matching Polygonal Curves with Respect to the Frechet Distance, . of *Computational Geometry: Theory and Applications.*, (Special Issue on the 19th Europ. Workshop on Comp. Geom) to appear.

[CG97] S. D. Cohen and L. Guibas. Partial matching of planar polylines under similarity transformations. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, 777–786, 1997.

[CH96] J. Chen and Y. Han. Shortest paths on a polyhedron. *Internat. J. Comput. Geom. Appl.*, 6:127–144, 1996.

[CW99] K. K. W. Chu and M. H. Wong. Fast time-series searching with scaling and shifting. In *Symposium on Principles of Database Systems*, 237–248, 1999.

[Dij59] E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematic*, 1:269–271, 1959.

[DP73] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Canadian Cartographer*, 10(2):112–122, 1973.

[EIV01] A. Efrat, P. Indyk, and S. Venkatasubramanian. Pattern matching with sets of segments. In *Proc. 12th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 295–304, 2001.

[GS00] Xianping Ge and Padhraic Smyth. Deformable markov model templates for time-series pattern matching. In *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 81–90, 2000.

[Kap99] S. Kapoor. Efficient computation of geodesic shortest paths. In *31st ACM Symposium on the Theory of Computing (STOC)*, 770–779, 1999.

[KO00] B. Kaneva and J. O'Rourke. An implementation of chen and han's shortest paths algorithm. In *Proc. 12th Canad. Conf. Comput. Geom. (CCCG)*, 139–146, 2000.

[KP99] E. Keogh, and M. Pazzani. Scaling up dynamic time warping to massive datasets. In *Proc. 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases (KDD)* 1–11, 1999.

[KCHP01] E. Keogh, S. Chu, D. Hart, and M. Pazzani. An Online Algorithm for Segmenting Time Series. In *IEEE International Conference on Data. Mining (ICDM)* 289–296, 2001.

[KS98] R. Kimmel and J. Sethian. Fast marching methods on triangulated domains. *Proc. National Academy of Sciences*, 95:8431–8435, 1998.

[LMS97] M. Lanthier, A. Maheshwari, and J. Sack. Approximating weighted shortest paths on polyhedral surfaces. In *Proc. 13th ACM Symp. on Computational Geometry (SoCG)*, 274–283, 1997.

[MMP87] Joseph S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou. The discrete geodesic problem. *SIAM J. Comput.*, 16:647–668, 1987.

[MP99] M. Munich and P. Perona. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In *International Conference on Computer Vision (ICCV)*, 108–115, 1999.

[MP03] M.E. Munich and P. Perona. Visual identification by signature tracking. *IEEE Transactions PAMI*, 25(2):200–217, 2003.

[Mun] Mario Munich. Signature data. `http://www.vision.caltech.edu/mariomu/research/data/signDist1.0.tgz`.

[OFC00] T. Oates, L. Firoiu, and P. Cohen. *Using Dynamic Time Warping to Bootstrap HMM-Based Clustering of Time Series*, chapter Sequence Learning: Paradigms, Algorithms and Applications. Springer-Verlag, 2000.

[PKC01] S. Park, S.W. Kim, and W. W. Chu. Segment-based approach for subsequence searches in sequence databases. *In Proc. of the Sixteenth acm Symposium on Applied Computing*, 248–252, 2001.

[RH93] L. R. Rabiner and B. H. Huang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[RK04] C. A. Ratanamahatana, and E. Keogh. Everything you know about Dynamic Time Warping is Wrong. *Third Workshop on Mining Temporal and Sequential Data*, in conjunction with the *Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004)*, 22–25, 2004.

[RM02] T. M. Rath and R. Manmatha. Lower-Bounding of Dynamic Time Warping Distances for Multivariate Time Series. *Technical Report MM-40*, Center for Intelligent Information Retrieval, University of Massachusetts Amherst, 2002.

[RM03] T. M. Rath and R. Manmatha. Word Image Matching Using Dynamic Time Warping. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 521–527, 2003.

[Set99] J. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 2nd edition, 1999.

[SB94] B. Serra and M. Berthod. Subpixel contour matching using continuous dynamic programming. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 202–207, 1994.

[Ven99] S. Venkatasubramanian. *Geometric Shape Matching and Drug Design*. PhD thesis, Department of Computer Science, Stanford University, August 1999.

[WHSB98] T. Wu, T. Hastie, S. Schmidler, and D. Brutlag. Regression analysis of multiple protein structures. In *2nd Annual Conference on Research in Computational Biology*, 585–595, 1998.