

Curve Shapes: Comparison and Alignment

John C. Femiani, Anshuman Razdan, Gerald Farin,

Abstract

We present a method for identifying a partial match for twice differentiable curves. The method uses equal arc-length interval correspondences when comparing points on one curve to another. We use the square of difference in curvatures at each point to compare two curves, and find it to be fast enough for querying a large database of curves.

Index Terms

curve matching, partial matching, curvatures.

John C. Femiani is a graduate assistant at PRISM, Department of Computer Science and Engineering, Arizona State University, MC8609, Tempe AZ 85287-8609. john.femiani@asu.edu

Anshuman Razdan is director of PRISM, Department of Computer Science and Engineering, Arizona State University, MC8609, Tempe AZ 85287-8609. razdan@asu.edu

Gerald Farin is a professor for the Department of Computer Science and Engineering, Arizona State University, MC8609, Tempe AZ 85287-8609. farin@asu.edu

I. INTRODUCTION

2D curves arise in a variety of applications, we list font design, topographic maps, sketches of object shapes, or handwriting analysis. If a collection of 2D curves is stored for a particular application, it is often necessary to distinguish between these curves or to identify a particular one. Consider the example of a collection of human face contours. If a new contour curve is supplied, we might want to identify the contour curve in the collection which is closest to the given contour. We refer to this process as *shape matching*.

It is necessary to define some terms before we present our curve matching method. First, let us define the terms *congruent* and *similar* [3] as they apply to curve matching. Two curves¹ \mathbf{x} and \mathbf{y} are congruent if and only if there exists an isometry (a distance-preserving transform) that transforms each point on \mathbf{x} to a corresponding point on \mathbf{y} , and vice versa. Valid isometric transforms include rotation, translations, and reflections, but not scaling². Congruence is a true/false concept, things are either congruent or they are not. If we consider congruence to be an equivalence relation on a set C of curves, then a function $d(\mathbf{x}, \mathbf{y})$ that would allow us to find the distance between any two curves is called a *metric*. To be a metric, the function $d(\mathbf{x}, \mathbf{y})$ must satisfy the following inequalities:

$$d(\mathbf{x}, \mathbf{y}) = 0 \quad \text{if } \mathbf{x} \cong \mathbf{y} \quad (1)$$

$$d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x}) \quad (2)$$

$$d(\mathbf{x}, \mathbf{z}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{y}) \quad (3)$$

A number of metrics exist for comparing entire curves. A *partial match* is a pairing between a curve and a region of another curve. A metric can not exist for partial matches, since the triangle inequality (3) can not hold. For example, suppose we have two short curves \mathbf{x} and \mathbf{y} and a longer curve \mathbf{z} . The curve \mathbf{x} can be congruent to a region of \mathbf{z} , and \mathbf{y} can be congruent to another region of \mathbf{z} while \mathbf{x} and \mathbf{y} do not have to be congruent at all.

Given a large database of twice-differentiable curves which we call *target* curves and a non-closed, twice differentiable curve which we call our *query* curve, our goal is to identify the best partial matches between the query curve and the target curves in the database. Our curves are allowed to self-intersect, (for example, they could trace the path of a pen on a paper) and the connectivity of points along the

¹We use bold-face roman letters to denote points, vectors, and curves

²If we wished to identify shapes that could be superimposed after uniform scaling, we would be concerned with *similarity*, but this paper focuses only on congruence.

curve must be respected as they are compared. This paper addresses the following issues regarding the comparison of curves:

- 1) Which points correlate between the query and the target curves?
- 2) How do we quantify the quality of a match?

When the provenance of curves is such that a meaningful parametrization (such as time) exists, then equal parameter-values on our query and target curve correspond. When data is purely geometric, we use an arc-length parametrization in order to compare them. The arc length heuristic for establishing a correspondence is always correct when one curve is congruent to the other, and it is generally quite good when the curves have similar shape.

We measure the difference between corresponding points on the query and target curves by comparing their curvatures since curvature is invariant under rigid body transforms. The best matching section of our target curve is identified by sliding our query curvature plot along our target curvature plot and measuring their difference, as illustrated in figure 1. This is not similar to measuring the average square of distances between points on the curve (called the mean squared error). If a curve x was perturbed slightly to \hat{x} , the two measurements would vary away from 0 differently.

One of the most popular methods for comparing and finding a correspondence between point sets is Iterative Closest Points (ICP) presented by Besl and McKay [2]. Some nice qualities about ICP are that it can be used to identify a best partial match, and it is not specific to curves. A drawback for the application to curve matching is that the correspondence it chooses is based solely on the proximity of points, regardless of parametrization or connectivity. ICP is nonlinear in nature, and does not find a globally optimal match, so it might have to be repeated several times with different initial poses in order to find a the best match over a whole curve. Veltkamp [9] presents a useful overview of a number of other shape similarity measures, including curvature scale space [8] and turning angles [1], [4].

The method of Curvature Scale Space (CSS) is used by the Shape Queries Using Image Databases (SQUID) project [7], and on-line demos are available for the interested reader. The method analyzes the zeroes of curvature after convolution with a filter. This method produces a fascinating signature plot for a curve by tracking the arc length distance s to each inflection point on the curve as the curve is convolved with a Gaussian kernel of width σ and plots a point (s, σ) for each inflection point. The inflection points move continuously along the contour as σ increases and eventually they meet in an *annihilation point* in the (s, σ) plane. Each annihilation point corresponds to either a concave or a convex region on the

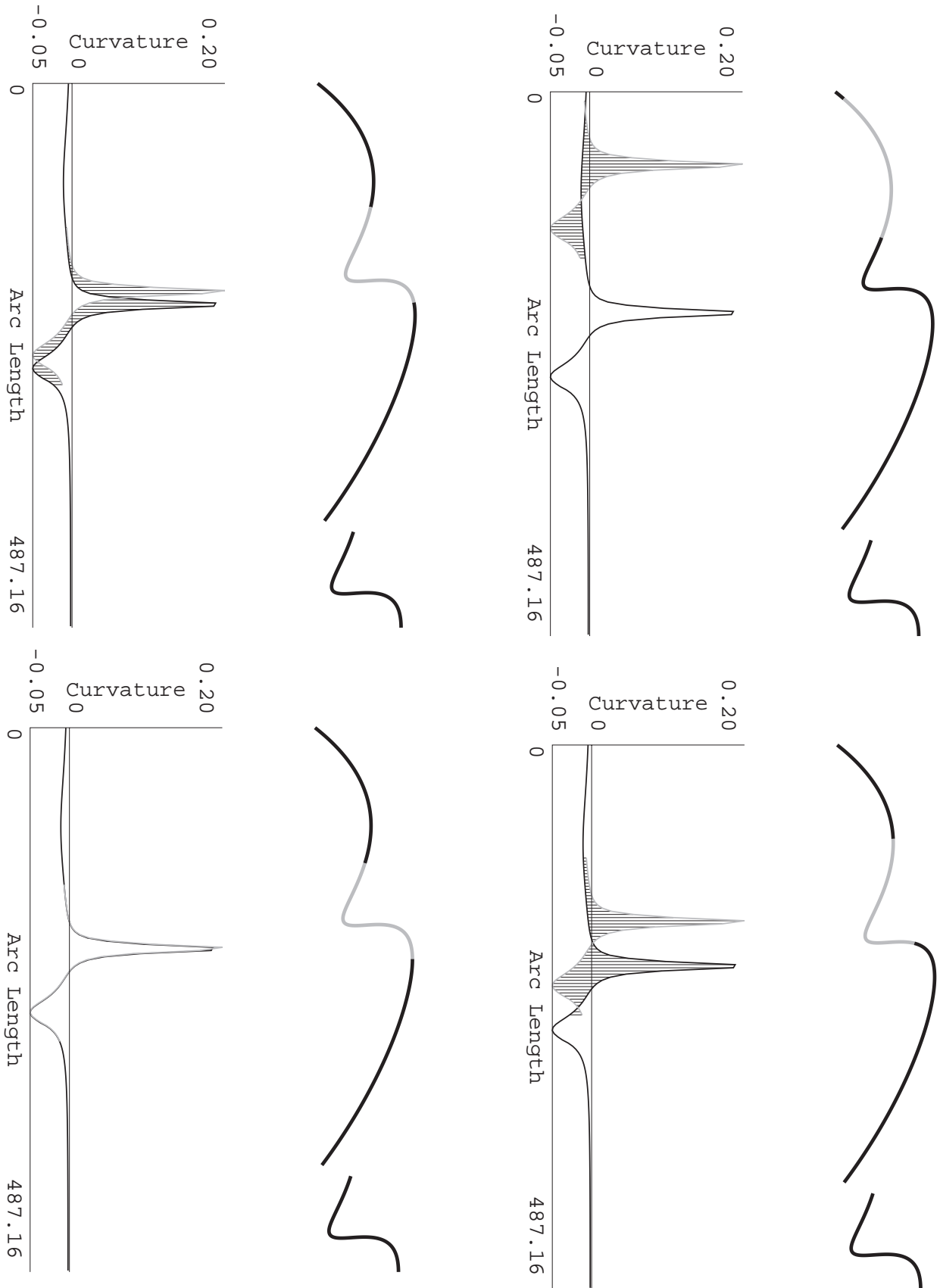


Fig. 1. An example of sliding two curvature plots to find a match

original curve. The minimum distance between the set of annihilation points after translation on the s axis can be used to compare curves. This method is not intended for partial matching or for open or self-intersecting curves. The CSS metric does not differentiate between convex curves, or even skewed version of the same curves, and Veltkamp's paper [9] shows an example of two species of hatchet fish with the same CSS representation.

Tangent angles, like curvature, provide a curve signature that is invariant under rigid body transforms. In fact, curvature is the derivative of the tangent angle with respect to arc length. The methods for comparing curves presented in [1] and for matching presented in [4] use graphs of tangent angle (which they call turning angle) versus arc length for polygonal curves. Determining the proper translation, rotation, and scaling for the two curves amounts to finding the right horizontal shift, vertical shift, and horizontal stretch that causes the two graphs to most closely resemble each other. The turning angle method presented by [4] is concerned with matching shapes in a scale invariant way, whereas we are mainly concerned with rigid body transforms (rotation and translation, but not scaling). Since they take scale into account, their method is $O(m^2n^2)$ where m and n are the number of sampled points on the two curves. By comparison our method is designed for smooth curves (twice differentiable) and is $O(mn)$ as theirs would be if scale-search part of their algorithm was eliminated.

II. CURVATURE MATCHING

In this section we outline an approach to partial curve matching. We take as input an an arc length parametrized query curve, which we call c , and target curve \hat{c} . We also use κ and $\hat{\kappa}$ do denote the curvatures of c and \hat{c} respectively, and $|\cdot|$ to denote the arc length of a curve. When the arc lengths are different, we choose \hat{c} to be the longer curve. We compare curves by measuring the difference between corresponding values on the curvature plot as shown in figure 1. This amounts to finding a parameter a such that

$$a = \arg \min_a \int_0^{|\hat{c}|} (\hat{\kappa}(s+a) - \kappa(s))^2 ds \quad (4)$$

We compare two curve segments by observing their signed curvature because it is an invariant property that completely captures the shape of a curve [10], [5]. We use the variable t to indicate that a curve uses a parametrization other than arc length, and s is used to indicate an arc length parametrized version of

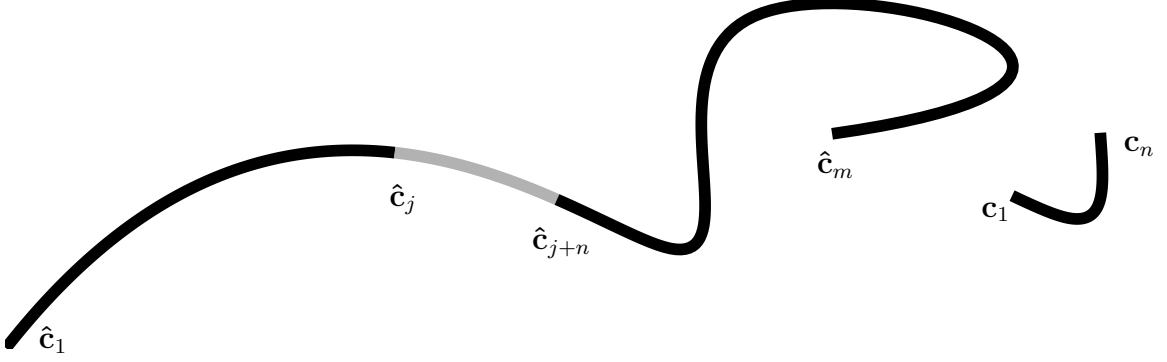


Fig. 2. An example of a query and target curve, and a region of the target curve

the curve. The curvature of a point on a curve can be obtained regardless of parametrization as follows:

$$\kappa(t) = \frac{\det \left[\frac{d}{dt} \mathbf{c}(t), \frac{d^2}{dt^2} \mathbf{c}(t) \right]}{\left\| \frac{d}{dt} \mathbf{c}(t) \right\|^3} \quad (5)$$

In order to evaluate integrals such as equation (4) we sample our curves at uniform increments δ of arc length chosen small enough to capture important aspects of the curve. This leaves us with $m = \left\lfloor \frac{|\hat{c}|}{\delta} \right\rfloor$ points on \hat{c} which we label $\hat{c}_1 \dots \hat{c}_m$, and $n = \left\lfloor \frac{|c|}{\delta} \right\rfloor$ points on c , which we label $c_1 \dots c_n$. Note that m is the floor of the arc-length of our target curve divided by the increment size. We have chosen \hat{c} to be the longer curve, so $m \geq n$ and there are at least $m - n + 1$ spans of n points in \hat{c} .

A curvature plot is a graph of curvature versus arc length (see figure 3). The vertical lines in figure 3 illustrate the difference in curvature at points on the two curvature plots. Our comparison of the two curves is based on the square of the differences between those plots, and our choice for a is based on a line-search for the minimum square of differences through our discretized curves.

$$\begin{aligned} a &= \left(\arg \min_j \sum_{i=1}^n (\hat{\kappa}_{j+i} - \kappa_i)^2 \right) \delta \\ &\approx \arg \min_a \int_0^{|c|} (\hat{\kappa}(s+a) - \kappa(s))^2 ds \end{aligned} \quad (6)$$

The asymptotic running time for curve matching is $O(mn)$, since $m - n + 1$ comparisons of n points must be made. Future work may address issues such as data storage and search strategies to make the search for a match and the process of comparison between curve segments much faster.

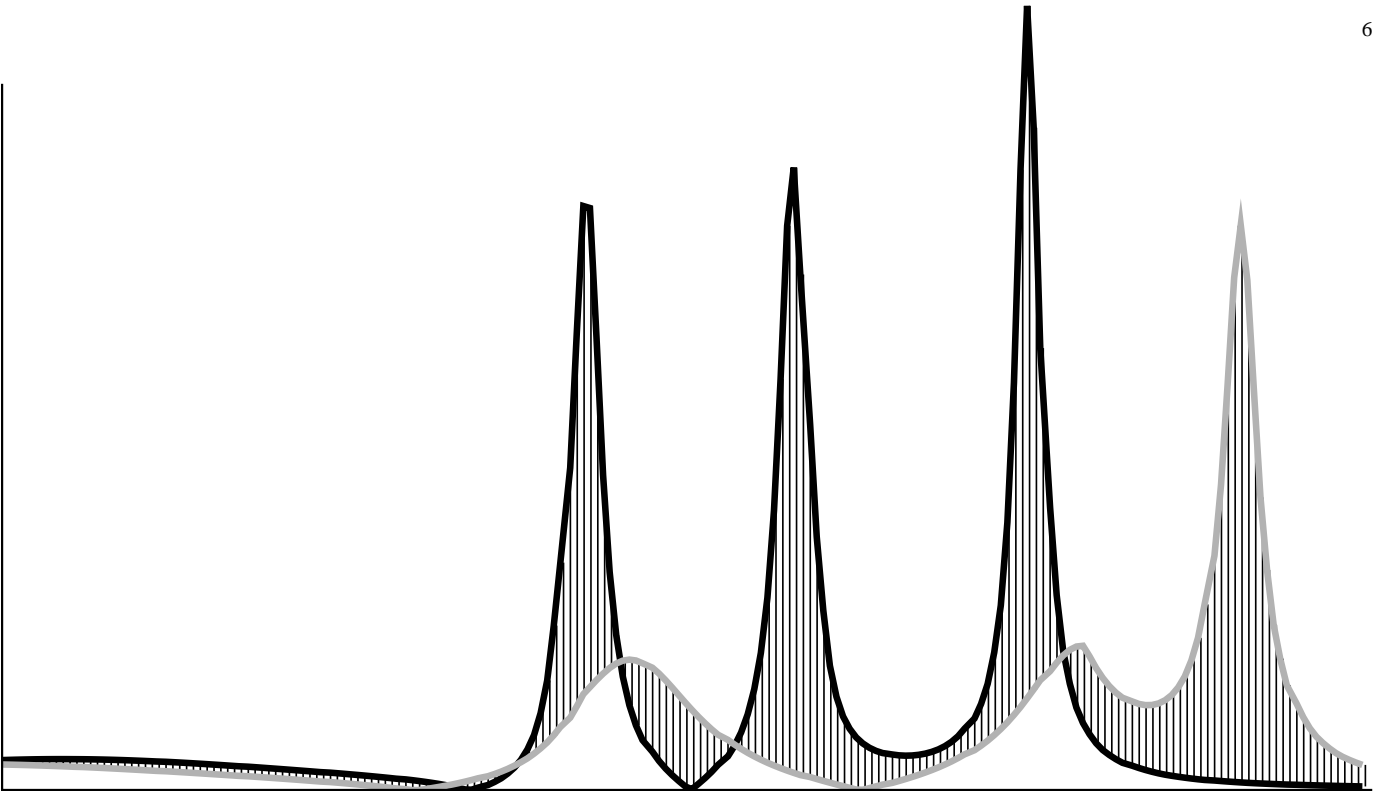


Fig. 3. An example of 2 curvature plots

III. ONE TO MANY COMPARISONS

Suppose we have a database consisting of N different curves, and we wish to search the database for a given input curve. For each curve in the database we have precomputed the curvature values and the arc lengths. We introduce C as our set of curves, and we use $c_{i,j}$ to refer to the j^{th} sampled point on the i^{th} curve in the database. We wish to find the top k curves³, where k is user defined. This means that whenever the summation in equation (6) exceeds the k^{th} smallest sum we can quit comparing that interval and continue sliding our curvature plots. We call this method a *thresholded search* since the k^{th} best match serves as a threshold allowing us to skip many computations.

Another issue arises when we compare to a large number of curves concerning arc length. If our query curve is *not* shorter than every other curve in C then the expression (4) would consider a longer curve as being poorer match (see figure 4). A solution is to divide the integral in equation (4) by the arc length of the shorter curve

$$a = \arg \min_a \frac{1}{|c|} \int_0^{|c|} (\hat{\kappa}(s+a) - \kappa(s))^2 ds.$$

However we feel that there is value in letting the length over which a match occurs influence its ranking. Particularly, we prefer to reward curves with a lower difference measurement if they match over a longer

³(i.e. after all curves are ranked according to the integral in equation (4) we select those k curves having the smallest shape difference)

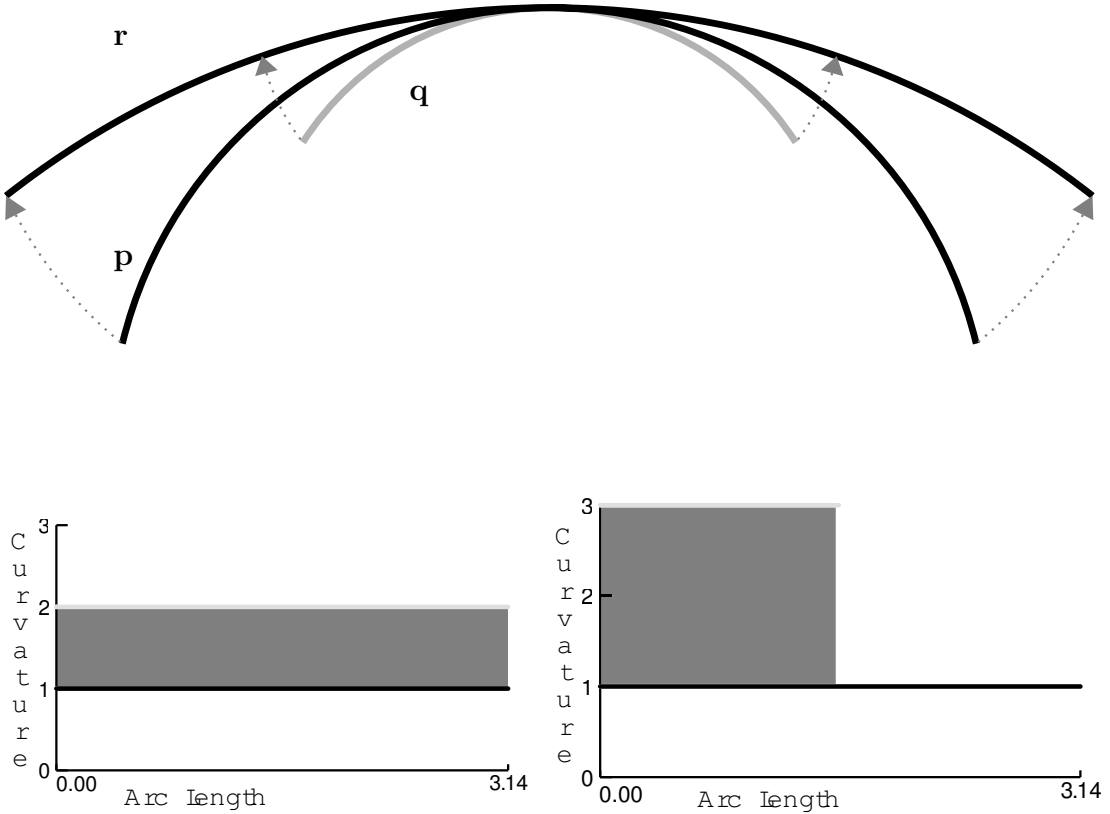


Fig. 4. Different curves with the same area between curvature plots

arc length. Figure 4 shows three arcs of a circle, p , q , and r . We consider curve p to be a much better match to r than q is. The modified version of equation (4) that we use is

$$a = \arg \min_a |c|^{-\alpha} \int_0^{|c|} (\hat{\kappa}(s+a) - \kappa(s))^2 ds \quad (7)$$

where α is a parameter chosen as follows. $\alpha = 1$ removes the effects of arc length, $\alpha > 1$ rewards a match over a longer arc length, and $\alpha < 1$ penalizes curves that match over a greater arc length. Choosing a proper α involves knowledge of what is in the database, and the choice should be guided by an expert on the kinds of shapes involved. For a general purpose database of curves, a choice of $\alpha = 1$ is likely the safest choice. Figure 5 shows the effects of α on a subset of a database of pottery profile curves.

IV. REGISTRATION

ICP as presented in [2] automatically aligns the point sets as the algorithm proceeds. Our algorithm does not require transforming the points at all, but we can find a rigid body transform efficiently after a match has been identified.

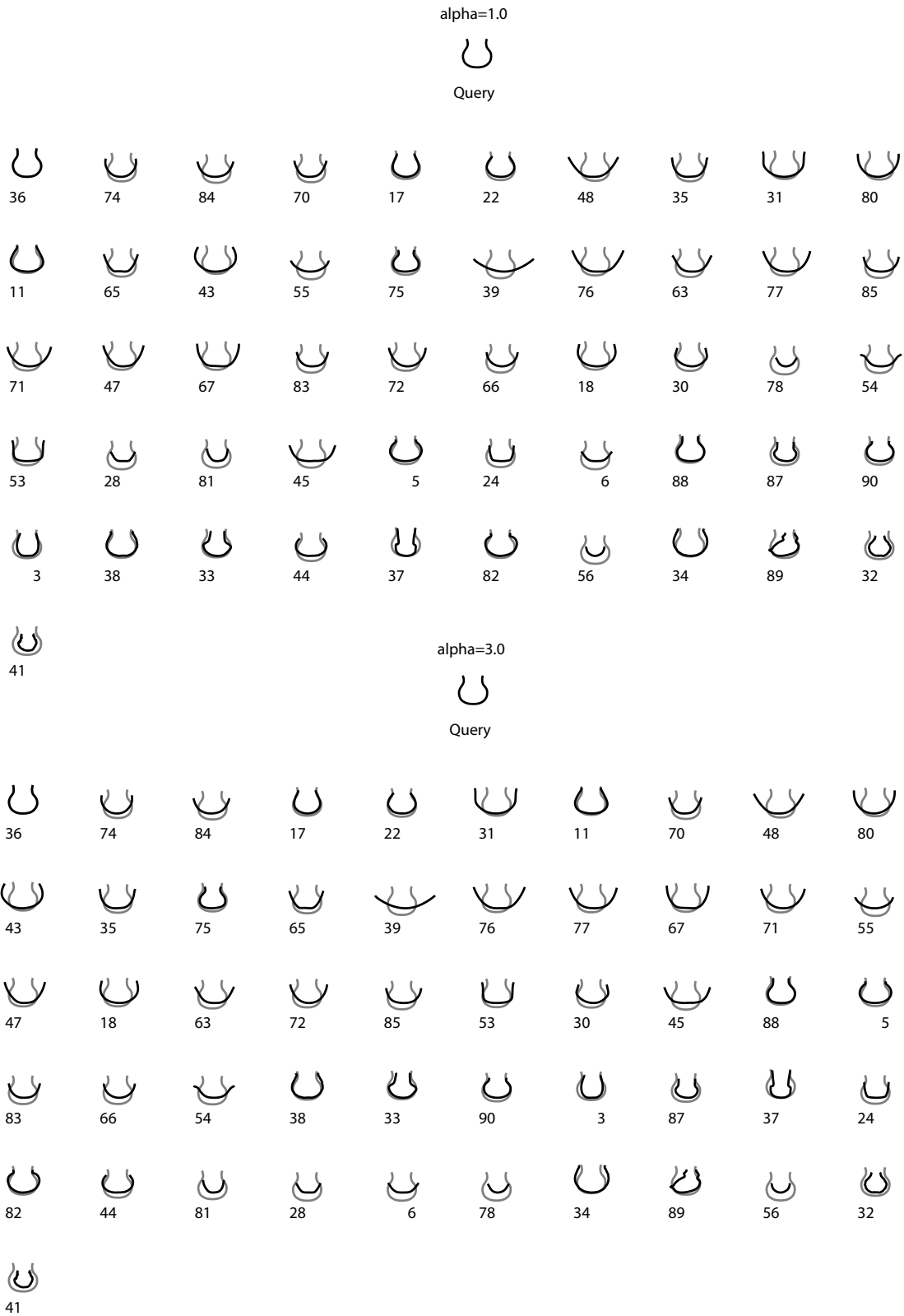


Fig. 5. Sorted results with $\alpha = 1$ (top) and $\alpha = 3$ (bottom)

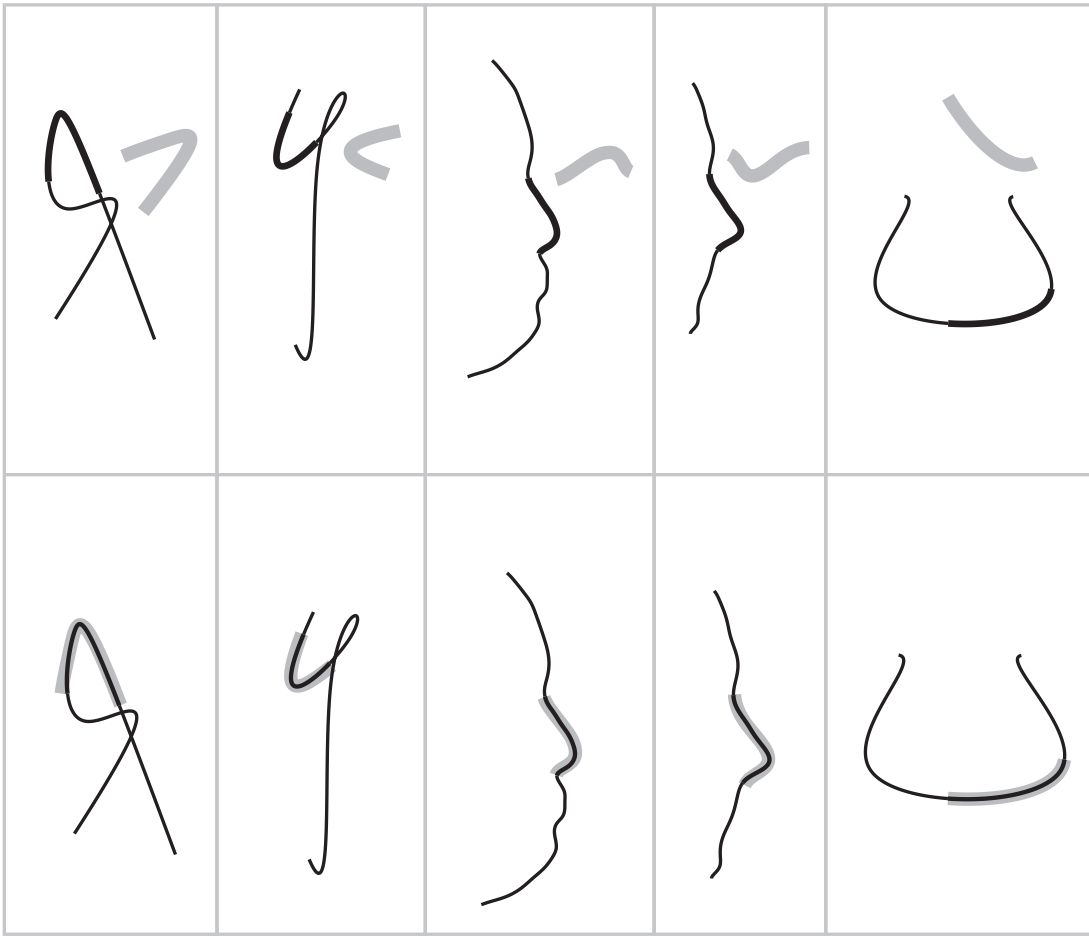


Fig. 6. 2D registration of matching points

ICP uses a quaternion based method presented by Horn [6] since it is intended to deal with 3D data, but Horn's paper also presents a 2D orientation method that we reiterate here.

We wish to find a vector \mathbf{t} and rotation matrix $R(\phi)$ that rotates points by the angle ϕ such that $\sum \|\hat{\mathbf{c}}_{i+j} - R(\phi)\mathbf{c}_i\|^2$ is minimal. If we define the points \mathbf{o} and $\hat{\mathbf{o}}$ as the centroids of the two point sets $\mathbf{c}_1 \dots \mathbf{c}_n$ and $\hat{\mathbf{c}}_1 \dots \hat{\mathbf{c}}_{j+n}$ then Horn shows that the $\mathbf{t} = \hat{\mathbf{o}} - R(\phi)\mathbf{o}$ is the optimal translation vector. For any two points \mathbf{c}_i and $\hat{\mathbf{c}}_{j+i}$ we can define the vectors

$$\mathbf{v}_i = \mathbf{c}_i - \mathbf{o}$$

$$\hat{\mathbf{v}}_i = \hat{\mathbf{c}}_{j+i} - \hat{\mathbf{o}}$$

and the angle $\gamma_i = \angle(\mathbf{v}_i, \hat{\mathbf{v}}_i)$ as illustrated in figure 7. We call $R(\phi)\mathbf{v}_i$ the vector \mathbf{v}_i rotated by an angle

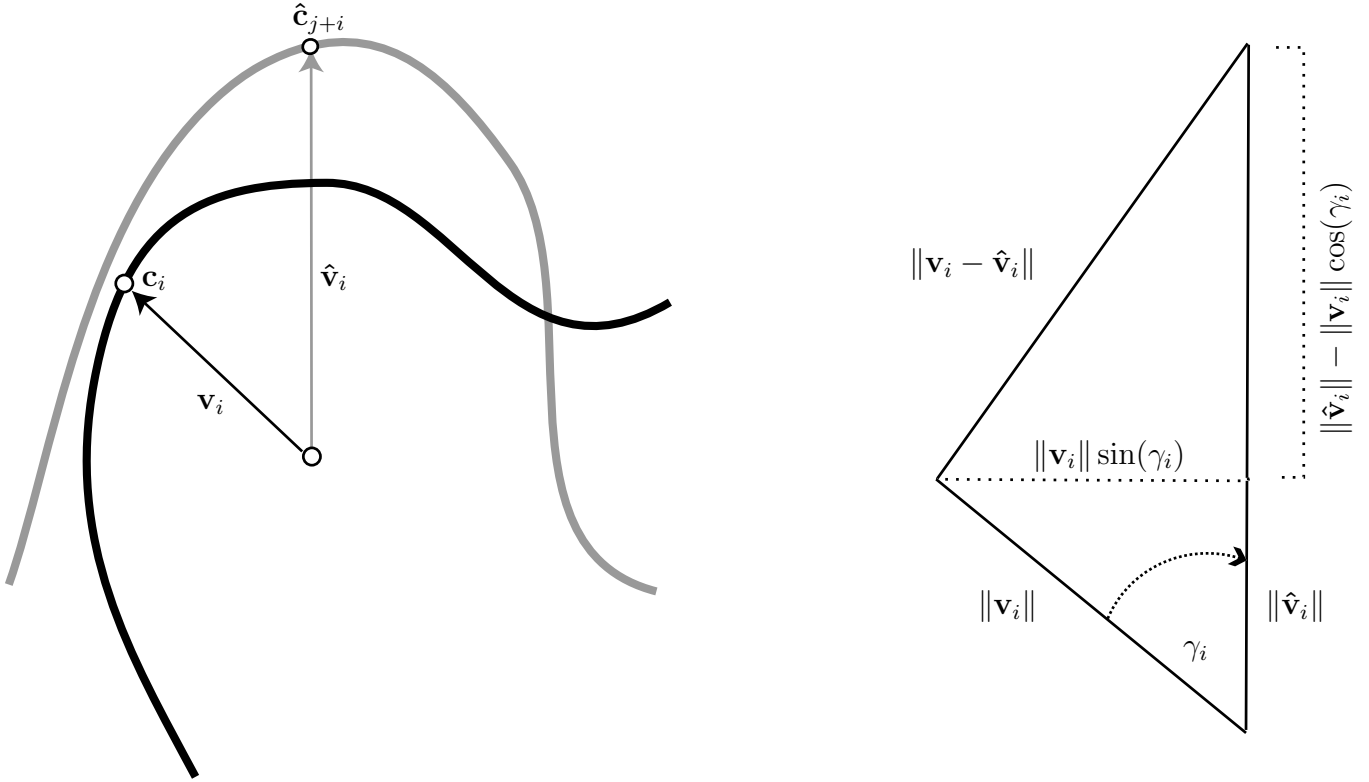


Fig. 7. Registration

ϕ , and

$$\|\hat{\mathbf{v}}_i - R(\phi)\mathbf{v}_i\|^2 = (\|\mathbf{v}_i\| \sin(\gamma - \phi))^2 + (\|\hat{\mathbf{v}}_i\| - \|\mathbf{v}_i\| \cos(\gamma - \phi))^2 \quad (8)$$

$$\begin{aligned} &= \|\mathbf{v}_i\|^2 \sin^2(\gamma - \phi) + \|\hat{\mathbf{v}}_i\|^2 \\ &\quad - 2\|\mathbf{v}_i\| \|\hat{\mathbf{v}}_i\| \cos(\gamma - \phi) + \|\mathbf{v}_i\|^2 \cos^2(\gamma - \phi) \end{aligned} \quad (9)$$

$$= \|\mathbf{v}_i\|^2 + \|\hat{\mathbf{v}}_i\|^2 - 2\|\mathbf{v}_i\| \|\hat{\mathbf{v}}_i\| \cos(\gamma - \phi) \quad (10)$$

$$\begin{aligned} &= \|\mathbf{v}_i\|^2 + \|\hat{\mathbf{v}}_i\|^2 \\ &\quad - 2\|\mathbf{v}_i\| \|\hat{\mathbf{v}}_i\| (\cos(\gamma) \cos(\phi) - \sin(\gamma) \sin(\phi)) \end{aligned} \quad (11)$$

$$\begin{aligned} &= \mathbf{v}_i^T \mathbf{v}_i + \hat{\mathbf{v}}_i^T \hat{\mathbf{v}}_i \\ &\quad - 2(\mathbf{v}_i^T \hat{\mathbf{v}}_i \cos(\phi) - \det(\mathbf{v}_i, \hat{\mathbf{v}}_i) \sin(\phi)) \end{aligned} \quad (12)$$

is the square of the distance between $\hat{\mathbf{c}}_{j+i}$ and $R(\phi)\mathbf{c}_i + \mathbf{t}$. We define the error measure parametrized by

an angle θ

$$E(\theta) = \sum_{i=1}^n \|\hat{\mathbf{v}}_i - R(\theta)\mathbf{v}_i\|^2 \quad (13)$$

$$= \sum_{i=1}^n \mathbf{v}_i^T \mathbf{v}_i + \sum_{i=1}^n \hat{\mathbf{v}}_i^T \hat{\mathbf{v}}_i \quad (14)$$

$$-2 \left(\cos(\theta) \sum_{i=1}^n \mathbf{v}_i^T \hat{\mathbf{v}}_i - \sin(\theta) \sum_{i=1}^n \det(\mathbf{v}_i, \hat{\mathbf{v}}_i) \right) \quad (15)$$

$$(16)$$

Since $E(\phi)$ is a minimum, the derivative of $\frac{d}{d\theta}E(\phi) = 0$. The derivative can be expressed in terms of $\sin \phi$ and $\cos \phi$

$$\frac{d}{d\theta}E = 2 \sin(\phi) \sum_{i=1}^n \mathbf{v}_i^T \hat{\mathbf{v}}_i + 2 \cos(\phi) \sum_{i=1}^n \det(\mathbf{v}_i, \hat{\mathbf{v}}_i) \quad (17)$$

In order to solve for $\sin(\phi)$ and $\cos(\phi)$ we introduce the variables σ , τ , and μ as follows

$$\sigma = \sum_{i=1}^n \mathbf{v}_i^T \hat{\mathbf{v}}_i$$

$$\tau = \sum_{i=1}^n \det([\mathbf{v}_i, \hat{\mathbf{v}}_i])$$

$$\mu = \sqrt{\sigma^2 + \tau^2}$$

$$(18)$$

In order for equation (17) to be 0 the value of ϕ must satisfy

$$\begin{bmatrix} \sin(\phi) \\ \cos(\phi) \end{bmatrix} = \pm \frac{1}{\mu} \begin{bmatrix} -\tau \\ \sigma \end{bmatrix}.$$

Recall that we actually want a rotation matrix $\mathbf{R}(\phi)$

$$\begin{aligned} R(\phi) &= \begin{bmatrix} \cos \phi & \sin \phi \\ -\sin \phi & \cos \phi \end{bmatrix} \\ &= \pm \frac{1}{\mu} \begin{bmatrix} \sigma & -\tau \\ \tau & \sigma \end{bmatrix} \end{aligned} \quad (19)$$

Now the value of $E(\phi)$ can efficiently be obtained without actually applying the rotation $R(\phi)$ by plugging

the values (IV) into equation (12) to give us the formula for $E(\phi)$ as follows

$$E(\phi) = \sum_{i=1}^n \mathbf{v}^T \mathbf{v} + \sum_{i=1}^n \hat{\mathbf{v}}^T \hat{\mathbf{v}} \mp 2 \left(\frac{\sigma^2 + \tau^2}{\mu} \right) \quad (20)$$

$$= \sum_{i=1}^n \mathbf{v}^T \mathbf{v} + \sum_{i=1}^n \hat{\mathbf{v}}^T \hat{\mathbf{v}} \mp 2\mu \quad (21)$$

In figure 6 we show the results of alignment for several curves.

V. RESULTS AND CONCLUSIONS

a) Comparison with ICP: We compare our method with ICP [2], although our method searches for a global minimum under the constraint that points at equal arc length distances correspond whereas ICP finds a local minimum with no constraint on the correspondence. We base our comparison on the curve matching method suggested in [2] using a standard version of ICP run several times at different starting positions. We generated 100 uniform cubic B-spline curves with ten randomly positioned control points. We extracted a section one quarter the length of each B-spline to use as a query curve. We always sampled 200 points on the longer curve, and 50 points on the shorter curve. In order to search for the closest points in ICP, we did a basic search that always compared $200 \cdot 50 = 10000$ points. Initial registrations were found by taking ten equally spaced spans of sampled points from our target curve and using Horn's quaternion method [6] to register them. The average running time over 100 trials was 0.52688s. We repeated the same experiment with our curve matching algorithm, fully evaluating the summation in equation (4) and not using the thresholding speedup mentioned in section III. We identified the correct match in 0.0046s on average, and were able to register the matching sections in 0.0006s on average. Our running times include computation of the curvature arrays. If they were pre-computed then our method would run in 0.0017s on average. These experiments were run on a Pentium IV 2GHz processor with 523 Mb of RAM. Figures 8 and 9 show starting and ending positions of an ICP-based curve matching that used 6 initial registrations.

A separate experiment was run to evaluate the speedup gained by only partially evaluating the summation in equation (6). For this experiment we used a different class of curves than B-splines, instead we generated the curvature plots by convolving an array of computer-generated pseudo-random numbers. The basic curvature matching routine was run on a database of 1,000,000 curvature plots, sampled at 800...1600 points with a query curve sampled at 100 points. The top match was identified in 440.17s (7 minutes) using the default method and in just 71s using the thresholded method. This experiment was run on a

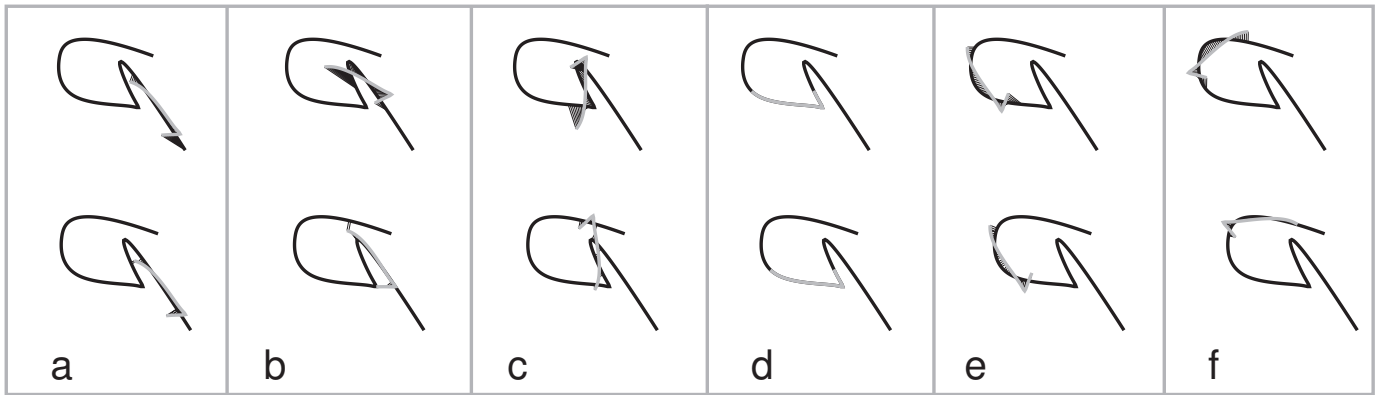


Fig. 8. Initial (upper) and final (lower) registrations for ICP based partial curve matching. d is a match.

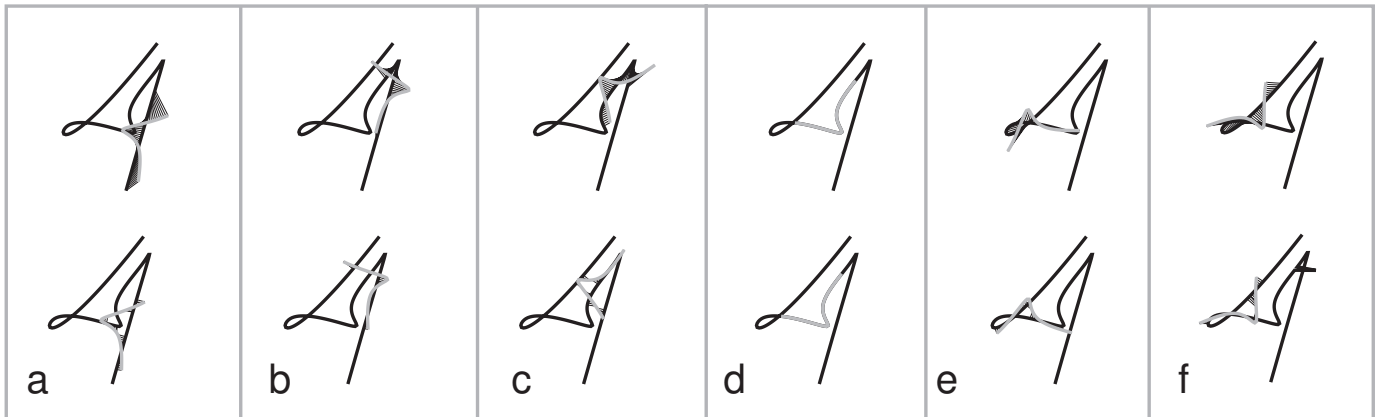


Fig. 9. Initial (upper) and final (lower) registrations for ICP based partial curve matching. d is a match.

Pentium IV 3.2GHz with 2GB RAM.

In conclusion, have presented an $O(mn)$ method to identify the best partial match for a query curve c within a target curve \hat{c} by comparing differences in curvature between the two curves. This is an order of magnitude better than ICP, which would have been $O(kmn)$, where k is the number of initial registrations. We use arc length parametrization to establish a correspondence between our query curve and portions of our target curve since it is much faster than searching for closest points. We also present a straightforward method due to [6] for simultaneously identifying a rigid body transform and the mean squared error between two curve segments.

Future directions include the following:

- 1) Our method can be extended to compare 3D curves by measuring the magnitude of the difference in the vectors $\mathbf{v} = [\kappa, \tau]^T$, where τ represents the torsion of the curve.
- 2) We can use the arc length heuristic, but use equation (21) to compare by mean squared error rather than curvature, or use a weighted combination. This expression lends itself to a thresholded

comparison like the one we use for curvature.

- 3) We might be able to quickly threshold out entire regions of our target curve if we reorder the summation in equation (6) in order to process curvature values in descending order of magnitude.
- 4) Any heuristic that increases our likelihood of finding the best matches sooner might allow us to increase performance by letting us threshold out larger portions of the database.

REFERENCES

- [1] Esther M. Arkin, L. Paul Chew, Daniel P. Huttenlocher, Klara Kedem, and Joseph S. B. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):209–216, March 1991.
- [2] Paul J. Besl and Niel D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2), 2 1992.
- [3] Manfredo P. Do Carmo. *Differential Geometry of Curves and surfaces*. Prentice Hall, 1976.
- [4] Scott D. Cohen and Leonidas J. Guibas. Partial matching of planar polylines under similarity transformations. *Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 777–786, January 1997.
- [5] Gerald Farin. *Curves and Surfaces for CAGD: A Practical Guide*. Morgan Kaufmann, fifth edition, 2002.
- [6] Berthold K.P. Horn. Closed-form solution of absolute orientation using unit quaternions. *A Journal of the Optical Society of America*, 4:629, 4 1987.
- [7] Farzin Mokhtarian. Shape queries using image databases.
- [8] Farzin Mokhtarian. Silhouette-based isolated object recognition through curvature scale space. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 17(5):539–544, May 1995.
- [9] Remco C. Veltkamp. Shape matching: Similarity measures and algorithms. *International Conference on Shape Modeling and Applications*, pages 128–201, 2001.
- [10] Robert C. Yates. *A Handbook on Curves and Their Properties*, pages 123–126. J.W. Edwards, 1952.