Average Curve of n smooth planar curves

Mukul Sati, Jarek Rossignac, School of Interactive Computing, Georgia Tech, USA
Raimund Seidel, Universität des Saarlandes, Germany
Brian Wyvill, University of Victoria, Canada
Suraj Musuvathy, Siemens Corporation, Corporate Technology, USA

Abstract

We define the *Average Curve* (AC) of a compatible set of two or more smooth and planar, Jordan curves. It is independent of their order and representation. We compare two variants: the *valley AC* (vAC), defined in terms of the valley of the field that sums the squared distances to the input curves, and the *zero AC* (zAC), defined as the zero set of the field that sums the signed distances to the input curves. Our formulation provides an orthogonal projection homeomorphism from the AC to each input curve. We use it to define compatibility. We propose a fast tracing algorithm for computing a polygonal approximation (PAC) of the AC and for testing compatibility. We provide a linear-cost implementation for tracing the PAC of polygonal approximations of smooth input curves. We also define the *inflation* of the AC and use it to visualize the local variability in the set of input curves. We argue that the AC and its inflation form a natural extension of the Medial Axis Transform to an arbitrary number of curves. We propose extensions to open curves and to weighted averages of curves, which can be used to design animations.

Keywords: Average of Curves, Medial Axis, Statistical Analysis of Shapes.

1. Introduction

Motivation: The concept of average of scalar values plays a central role in statistics [52] and hence in a broad variety of application areas. It also extends naturally to points [44]. But, extending the notion of an average to a set of $n > 2$ planar curves is non-trivial.

Theoretical contributions: We define the *Average Curve* (AC) of any "compatible" set of n smooth Jordan curves (i.e., simple loops) in the plane and propose it as the mathematical formulation of their average.

We define the *Gap* between the input curves and provide a formulation and also a sufficient condition for *compatibility* that is based on the *Gap Relative Projection* (GRP).

We define the *inflation* of the AC and show that it helps to convey the local variability in the input curves at any point along their AC. Fig.1 illustrates the use of the AC and inflation for statistical shape analysis and visualization. The thickness of the inflation is a more useful statistical indicator of local variability than the thickness of the gap.

We compare two variants: (1) the *valley AC* (vAC) defined in terms of the valley of the field that is the sum of the squared distances to the input curves and (2) the *zero AC* (zAC) defined in terms of the zero set of the field that is the sum of the signed distances to the input curves. They are nearly identical for the configuration of Fig.1.

Construction algorithm: Our *Trace* algorithm for computing the polygonal approximation (PAC) of the AC is independent of the order and representation of the curves. It produces consecutive samples along the AC incrementally, while traversing each curve only twice.

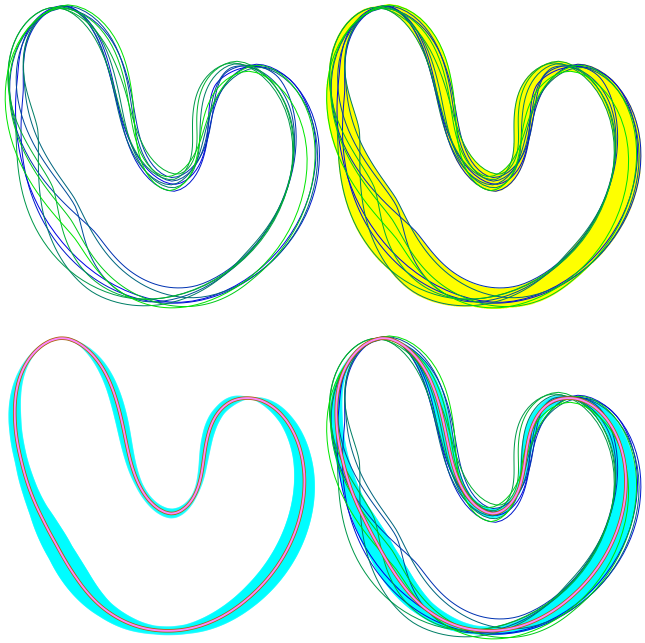

Figure 1: Top-left: Ten input curves. Top-right: Over their gap (yellow). Bottom-left: AC (pink vAC over a larger brown zAC) over inflation (cyan). Bottom-right: compared to the input curves.

Implementation: We present a linear-cost algorithm for computing the PAC from finely sampled polygonal approximations of smooth input curves. We use it to show the effectiveness and limitations of the AC and to compare its vAC and zAC variants.

Extensions: We propose extensions of the AC to open curve segments and to the weighted average of curves, which we illustrate by showing patterns of curves or frames of curve animations.

Applications: We anticipate that the availability of a precise mathematical definition for the average of a set of curves (and ultimately of surfaces) will have benefits in a broad set of application areas.

For example, it may play a role in legal matters and in coastal studies, as a tool for providing precise definitions of shorelines in spite of their periodic fluctuations and for tracking the evolution of their average (rather than of their extrema) over time [29].

Shape averaging also has important industrial applications. For example, being able to compute the average shape of a set of machined parts is fundamental for assessing the statistical consistency of a manufacturing process [20]. Early efforts in developing formal models of statistical tolerances for CAD are discussed in [41].

Many aspects of medical research, surgery planning, and diagnosis are based on comparing the anatomy of a particular patient to the average anatomy of a population [46]. Hence, it is important to have a precise definition of that average anatomy and algorithms for computing it.

Finally, artists often prefer overdrawing (rather than erasing and redrawing or manipulating control points) to modify a curve locally. In digital approaches that support such an interface [13], it makes sense to define the final curve in terms of the (possibly weighted) average of the overdrawn strokes.

Other applications of shape averaging are illustrated in [43] (which proposes to define the average as a minimizer of an elastic deformation energy).

We illustrate applications of the AC to the statistic analysis of medical scans and of the weighted average to the design of animations of a smooth curve defined by several control curves.

2. Prior art and variations

We discuss prior approaches for computing averages of curves in terms of: (1) *correspondence*, which establishes a mapping between all input curves, and (2) *construction*, which defines a point of the average in terms of each set of corresponding points on the input curves and possibly of the corresponding normals. We also mention approaches that define the average in terms of fields or concatenations of input segments.

Arc-length or Landmark correspondence: Inferring correspondence from a normalized arc-length parametrization may lead to unacceptable artifacts [45]. Inferring correspondence from landmarks (salient features, extrema, inflections) [19, 34] either requires tedious user input or delicate heuristics for matching possibly incompatible sets of landmarks on different curves. The solution proposed here does not reply on arc-length parametrization or on landmark matching, but requires that the curves be smooth and compatible (i.e., reasonably parallel).

Generalized Fréchet correspondence: An $\mathcal{O}(n^2 \log n)$ algorithm for computing the Fréchet distance between two polygonal curves was proposed in [2], extended to more general curves in [40] and to more than two curves in [16]. An $\mathcal{O}(k^n)$ algorithm for computing a mean curve for a set of n polygonal curves, each of complexity $\mathcal{O}(k)$, was proposed in [24]. It minimizes its maximum Fréchet distance to each input curve. Unfortunately, small changes in the input curves can lead to large changes in the Fréchet metric (see Fig. 2.4 of [47]) and hence in the mean. To overcome this drawback, for two curves, measures that compute the summed and average Fréchet distance have also been proposed [47]. The solution proposed in this work has complexity $\mathcal{O}(nk)$, but assumes that the input curves are polygonal approximations of smooth and compatible input curves.

Normal or Radial correspondence: When one of the curves can be chosen as *base* so that each other curve is its *normal offset* [38] or its *radial offset* [38], the resulting correspondence (*ortho map,normal map* [10, 11]) is a homeomorphism. Unfortunately, the average obtained by using these correspondences is asymmetric: it depends on which curve is chosen to be the base. The solution proposed here is symmetric: it computes the AC as a new curve that is independent of the order of the input curves and establishes a normal map from the AC to each input curve.

Minkowski correspondence: The Minkowski average establishes a correspondence between points with the same normal. Realtime rendering approaches have been proposed for the weighted Minkowski average of two [31] and of more [9] polygonal solids. Unfortunately, it fails to satisfy many desirable properties of the average curve that AC has. For example, the Minkowski average of a non-convex curve C with itself is not C (their AC is).

Other approaches to correspondence: Correspondence between two curves can also be established by minimizing the integral of distance between corresponding points [21], by adapting the sampling step to the local curvature [15], or by minimizing the affine transformations that map corresponding pieces [49].

Construction: Many approaches use a *construction* based on the centroid of the corresponding points. The medial axis [6] and its variations (as discussed in [54]) do not. Similarly to the medial axis construction, the solution proposed here constructs points on the AC as normal offsets of the corresponding (i.e., locally closest projection) points on the input curves. A variant that constructs the centroid of the corresponding points yields nearly identical results, but a different parametrization.

Averaging discrete points: The algorithm proposed in [26] computes a polygonal average C for an unordered set of discrete sample points by initially aligning C with the least-square fit line and then iteratively moving its vertices towards weighted averages of samples that have close projections on C. The authors of [26] point out that there is no guarantee of uniqueness or existence. Applying this solution to a sampling of all input curves will, in general, not yield the AC, but the duality between the two approaches is intriguing.

Our algorithm implements a fixed-point iterative scheme to converge to a point on the AC and this may be viewed as an extension of Moving Least Square techniques for computing a surface that interpolates a point cloud: set of input points or surfels (points associated with surface normals) [1, 4]. In a future 3D extension of the vAC, the desired "average" surface may be defined by several surfaces, each one defined by its cloud.

Zero set of a field: An average of two curves was defined in [14] as the zero set of a heat field. The result is similar to the Ball Morph [54]. A linear and cubic interpolation of density in medical scans was discussed in [27] as a mean to produce interpolating cross-sections, which were defined on each slice as zero-sets of a 3D field. This principle was exploited in [51] to define morphs between pairs of curves and between pairs of surfaces. When using a signed distance field, the mid-course curve of such a morph corresponds to the zAC of two curves. We extend this approach to more than two curves, propose the vAC variant, present the *Trace* algorithm and its efficient implementation for tracing the AC, and define the GRP to increase the set of configurations for which *Trace* produces the desired result.

Median trajectories: Given a set of polygonal trajectories, a median trajectory is constructed [7] by concatenating selected segments of input trajectories.

3. Theoretical definitions of AC and compatibility

We consider an input set $\{C_i\}$ of n smooth curves. We propose two formulations of their AC: vAC and zAC.

vAC: The arithmetic mean, x, of a set of numbers, x_i, is $argmin_x(x - x_i)^2$. Hence, we define the *Valley Average Curve* (vAC) as the valley of the scalar height field Q of the sum of squared distance. For any point P, $Q(P) = \sum_i P_iP^2$ where P_i is a "local closest projection" of P onto C_i. Its precise definition is discussed further below. Also, in our notation, N_i represents the outward normal to C_i at P_i. Fig. 2 shows the vAC for a compatible configuration and the associated height field.

zAC: The average, x, of a set of numbers, x_i, is the value that cancels the derivative $\sum_i(x-x_i)$ of $\sum_i(x-x_i)^2$. Hence, we define the *zero Average Curve* (zAC) as the zero set of the scalar height field D of signed distances, where $D(P) = \sum_i P_iP \cdot N_i$. Fig. 3 shows the zAC for the configuration of Fig. 2 and the associated height field.

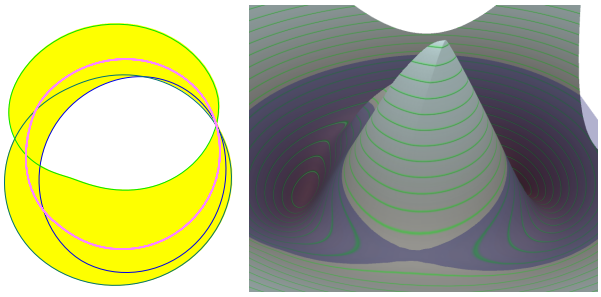

Figure 2: Left: vAC (pink) of 3 input curves (green-blue ramp). Right: Terrain with height $\sqrt{Q(P)}$. The vAC traces the valley that lies under the (transparent blue) water surface.

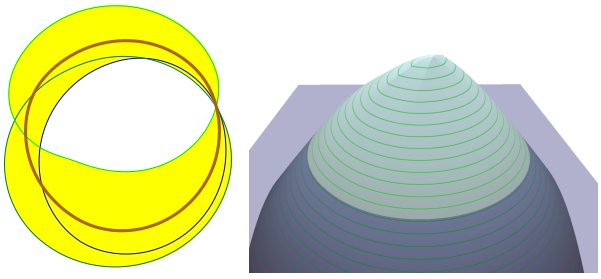

Figure 3: Left: zAC (brown). Right: Terrain with height $D(P)$. A semitransparent ($z = 0$) plane shows the zero-crossing.

To discuss the proper definition of the "local closest projection", P_i, we use the following two concepts.

Face: F_i denotes the *face* (bounded and open set) that has C_i as boundary. By our convention, F_i abuts C_i from the right.

Gap: Extending (to more than two curves) the definitions proposed in [53, 8], we define the *gap* of $\{C_i\}$ as $(\cup F_i)^- - (\cap F_i)^\circ$, where H^- and H° denote respectively the topological closure and interior [1] of set H [50].

Inadequacy of using global closest projection: Defining P_i as the closest point on C_i to P is adequate for simple configurations, such as the one of Fig. 2, but may fail in more complex ones.

Fig. 4 shows an example of a configuration where the vAC contains components outside of the gap. Note that zAC does not have this problem, since, for any point P outside of the gap, the components $P_iP \cdot N_i$ all have the same sign.

Fig. 5 shows a configuration where both vAC and zAC yield inadequate results. To understand why a segment of the left-most portion of the AC in the gap is missing in the vAC and lacking the expected curvature in the zAC, consider a point P on that missing segment. Its closest projection P_0 onto the outer-most (green) curve C_0 is not on the left-most part of C_0, but on a different portion of C_0.

[1] The closure and interior operators are necessary to deal with cases where the gap includes lower-dimensional portions. For example, when all input curves are identical, the gap is one-dimensional and is identical to these curves.

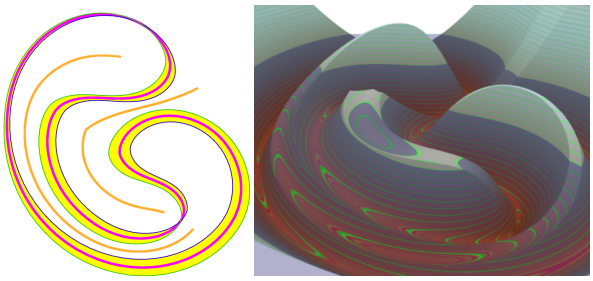

Figure 4: Left: The vAC of two curves (green and blue) has 3 components: one in the gap (pink) and two outside (orange). Right: Height field proportional to $\sqrt{Q(P)}$.

One can verify in this example that such erroneous closest projections cannot be characterized by $P_iP \cdot N_i < 0$. So, to prevent them, we define P_i as the GRP of P.

GRP: The *Gap Relative Projection* (GRP) defines P_i as a point of C_i such that the line segment (P_i, P) lies in the gap and is orthogonal to C_i at P_i.

Definition of compatibility: We say that the input curves are *compatible* if there exists a curve, C, such that the GRP map from point P to its GRP, P_i, is a homeomorphism[2] for all i.

The problem with this definition is that it can be tested only if a suitable curve C is found. Hence, we define below a sufficient, although not necessary, condition for compatibility (which we call *strong compatibility*) that can be verified without computing C.

Ball and normal compatibility of two curves: Two curves are compatible if and only if they are *ball compatible* (BC) [8] (i.e.,when the symmetric map between them, called the *ball map*, [8] is a homeomorphism.) A sufficient condition for ball compatibility is that the minimum of their *local feature size* (LFS) exceeds their Hausdorff distance [8]. The *normal map* [11] (i.e. the closest point map) from C_0 to C_1 associates with each point P_0 of C_0 the set of points P_1 of C_1 at which the distance to P_0 passes through a local minimum. If the normal map from each curve to the other is a homeomorphism, the two curves are said to be *normal compatible* [11] (NC). The equivalent and more descriptive term of *projection homeomorphic* was used in [10]. Two curves are NC if the minimum of their LFS [3] exceeds their *Hausdorff* distance [42] divided by $(2-\sqrt{2})$. (See [11] for a formal proof and intuition.) NC is a more constraining condition than BC [8].

Gap compatibility and strong compatibility: We say that the input curves are *gap compatible* if the GRP, P_i, is unique for all i and for all points P in the gap.

We say that the input curves are *strongly compatible* if each pair is normal-compatible.

Spike: Consider any one of the input curves: wlog C_0. For each point P_0 of C_0, we consider the line L that is orthogonal to C_0 at P_0. We consider the set $\{S\}$ of segments of the intersection of L with the gap. We define the *spike* of C_0 at P_0 as the segment of $\{S\}$ that contains P_0.

[2]In other words, the GRPs establish continuous and bijective correspondences between the AC and the input curves.

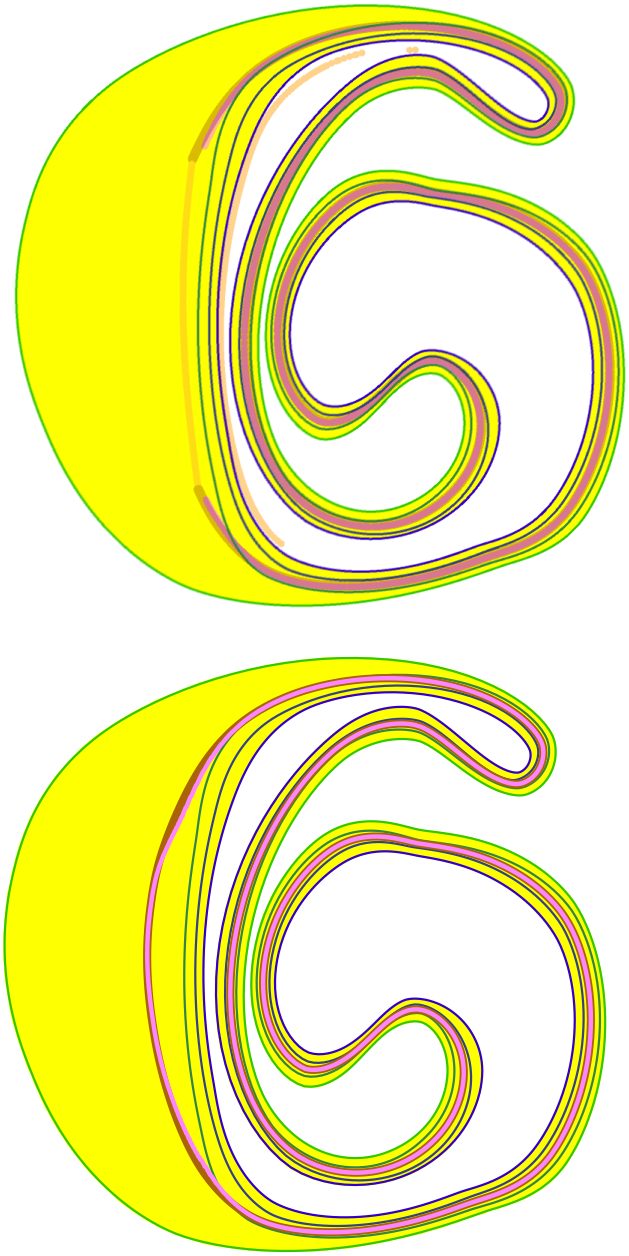

Figure 5: Configuration of four input curves, three of which are similar. Top: the valley of Q contains components inside (pink) and outside (orange) of the gap and is discontinuous (misses its left-most section) inside the gap. Overlaid over this is the zero set of D (thick brown) that lies within the gap, but contains an erroneous region (orange), which results from using projections onto segments of the left-most green input curve that are not visible form within the gap. Bottom: Correct result (pink vAC over thick, brown zAC) obtained by using the modified (GRP) definition of P_i.

Strong compatibility implies gap compatibility: If the input curves are strongly compatible, the spikes of C_0 are pairwise disjoint and their union is the gap. Indeed, each endpoint of a spike lies on another input curve, wlog C_1. Since C_0 and C_1 are normal compatible, the spikes

never cross (see Corollary 1 of [11]).

Given that the above holds for any choice of curve C_0, any point P of the gap lies on exactly one spike of each curve.

Hence, strong compatibility implies gap compatibility.

Conjectures: We advance the following conjectures for strongly compatible configurations: (1) the restriction of $Q(P)$ along each spike (i.e., the value of $Q(P)$ as a function of arc-length parameter along the spike) has a local minimum and (2) there is only one local minimum per spike.

We have proven the conjecture for sections of the gap where the input curves are locally straight (see Section 4).

Our experiments strongly suggest that the conjectures are true for more general configurations, but we do not have a formal proof.

Comparing vAC and zAC: For a typical valid input, the vAC and zAC are nearly identical. In fact, they are provably identical in sections of the gap where the input curves are parallel (constant radius offsets [39] of each other). A slight difference can be observed (see Fig. 6) in sections of the gap where corresponding GRPs have tangents with significantly different orientations.

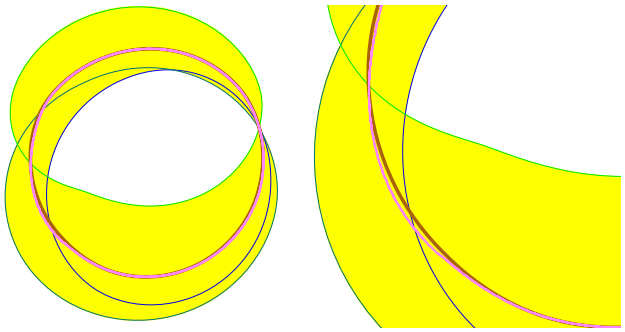

Figure 6: vAC (pink) over zAC (brown), and zoom (right).

Important properties: Both variants of the AC satisfy the following properties: (1) The AC is a non-contractible Jordan curve in the gap. (2) The AC is uniquely defined and independent of the order of the input curves. (3) The AC is independent of the representation and parametrization chosen for each curve. (4) In places where all input curves are close to each other and locally parallel, the GRPs, P_i, are nearly collinear and the corresponding point on the AC is close to their centroid. (5) The AC passes through all points that are contained in all input curves. (6) The AC commutes with similarity transformations.

Definitions of inflation: With each point, P, of the AC, we associate a radius, $r(P)$, that captures the local variability (i.e., deviation of the input curves from their AC). We define the *inflation* as the infinite union of the disks of center P on the AC and of radius $r(P)$[3]. In Fig. 1, we show the inflation for an L_1 norm, where $r(P) = \sum_i d(P, P_i)/n$. For some applications, one may prefer an L_2 norm, where $r(P) = \sum_i P_i P^2/n$.

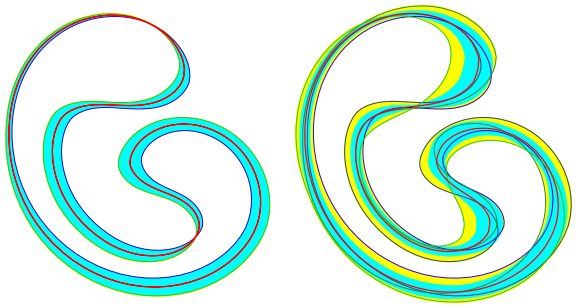

Figure 7: Left: AC (equal to the MA) and inflation (equal to the gap) for two compatible curves. Right: AC and inflation for four curves.

Relation to the MAT: Consider configurations of two curves. The *Medial Axis* (MA) of their gap is the set of the centers of all maximal disks [6] in the gap. The two curves are compatible if and only if the MA of their gap is a Jordan curve (has no bifurcation). The Medial Axis Transform (MAT) [6, 48] of the gap augments the MA with a radius function that is identical to r(P), regardless of whether we use the L_1 or L_2 norm (see Fig. 7).

Hence, we view the AC and its inflation as a natural extension of the MAT to an arbitrary number of curves.

4. AC of lines

Before presenting our algorithm for tracing the AC of compatible curves and for testing compatibility, we discuss the special case where the input curves are not Jordan curves, but infinite lines. We do so because our *trace* algorithm for tracing the PAC is based on the closed-form expressions for computing the closest projection of an arbitrary point P onto the AC of lines. But our formal vAC and zAC definitions of the average of a set of lines may have other applications.

We consider a set $\{L_i\}$ of oriented lines in the plane. We define line $L_i = L(S_i, T_i)$ by an arbitrary point S_i on the line and by a unit tangent vector T_i. N_i denotes the normal to L_i. We define the corresponding face F_i as the half-space $\{P : S_i P \cdot N_i < 0\}$.

We prove that the AC is a line and provide a closed-form expressions for computing its normal, N, and the closest projection, V, of an arbitrary point, P, onto the AC.

We do so for both the vAC and the zAC variants.

vAC of lines: Let $Q_L(P) = \sum_i (S_i P \cdot N_i)^2$. Using Cartesian coordinates (x, y) for P, (x_i, y_i) for S_i, and (u_i, v_i) for N_i, and writing $Q_L(x, y)$ for $Q_L(P)$, we obtain

[3] The envelop of the inflation can be computed as the radial offset of the AC [38].

$S_iP \cdot N_i = (x - x_i)u_i + (y - y_i)v_i$ and thus

$$Q_L(x,y) = \sum_i (u_i^2x^2 + v_i^2y^2 + 2u_iv_ixy - 2u_i(u_ix_i + v_iy_i)x - 2v_i(u_ix_i + v_iy_i)y + 2u_iv_ix_iy_i + u_i^2x_i^2 + v_i^2y_i^2)$$
$$= ax^2 + by^2 + cxy + dx + ey + f \quad (1)$$

Q_L is a quadratic polynomial. Hence, we can use the following matrix notation with $X = [x\ y]^T$:

$$Q_L(x,y) = X^T\mathbf{H}X + \mathbf{B}^TX + \mathbf{C}$$

with $\mathbf{H} = \begin{bmatrix} a & c/2 \\ c/2 & b \end{bmatrix}, \mathbf{B} = \begin{bmatrix} d \\ e \end{bmatrix}, \mathbf{C} = [f]$.

The vAC of a set of lines is the *valley* of their height field Q_L.

The valley of Q_L is defined by $\nabla Q_L \cdot E_2 = 0$, where E_2 is the eigenvector of the Hessian $\mathbf{H}$ of Q_L that corresponds to the larger eigenvalue of $\mathbf{H}$ [17, 37]. Here,

$$\nabla Q_L = \begin{bmatrix} 2ax + cy + d \\ 2by + cx + e \end{bmatrix}$$

The eigenvalues of $\mathbf{H}$ are the roots of its characteristic equation: $\lambda^2 - (a+b)\lambda + ab - c^2/4 = 0$. The eigenvectors (E_i) of $\mathbf{H}$ can be computed in closed form.

Because $\mathbf{H}$ is constant, E_2 is constant and $\nabla Q_L \cdot V = 0$ defines a line $L(S,T)$ that passes through point $S = (2bd - ce, 2ae - cd)/(c^2 - 4ab)$, which minimizes $Q_L(P)$, and that has for tangent a vector T orthogonal to E_2.

Thus, the valley of lines is a line and its representation can be computed it in closed form.

T is undefined for special configurations. For example, for two orthogonal lines, Q_L has radial symmetry. Conveniently, these configurations are excluded, since we assume that the input is strongly compatible.

One can easily show that the tangent $T = (\cos\alpha, \sin\alpha)$ defined above is the minimizer of $\sum_i \sin^2(w_i)$, where w_i is the angle between T and T_i, and that α may also be computed using:

$$\tan(2\alpha) = \frac{2\sum x_iy_i}{\sum x_i^2 - y_i^2} \quad (2)$$

Computing point S using the expression above for the global minimizer of Q_L fails in situations where the vectors T_i are (nearly) parallel, because the minimizer, S, may be (nearly) at infinity, which leads to (numeric instability or) a division by zero. To address this problem, we have devised a robust and fast computation for the closest projection, V, of P onto the vAC, L. We formulate it as $V = P + sN$, where N is orthogonal to T, and solve for the value of s that minimizes $Q_L(V)$.

The details of computing T and V are presented in Algorithm 1 for the *ProjectV* function, which, given a point P returns the closest projection V of P onto the vAC. In our notation, vector $N_i = (N_i.x, N_i.y)$.

zAC of lines: Let $D_L(P) = \sum_i (S_iP \cdot N_i)$. Its zero-set is the line defined by $\sum_i (S_iP \cdot N_i) = 0$. Replacing S_iP by $S_iO + OP$ yields $OP \cdot (\sum_i N_i) = -\sum_i (S_iO \cdot N_i)$, which is the equation of a line with normal directed along $N = \sum_i N_i$.

```
a = 0; b = 0; c = 0; n = 0; d = 0;
for each line L_i do
    a += N_i.x^2;
    b += N_i.y^2;
    c += N_i.x N_i.y;
end
α = atan2(2c, a − b)/2;
N = (cos α, sin α);
for each line L_i do
    n += (PP_i · N_i)(N · N_i);
    d += (N · N_i)^2;
end
V = P + (n/d)N;
```

Algorithm 1: *ProjectV* on the vAC of lines

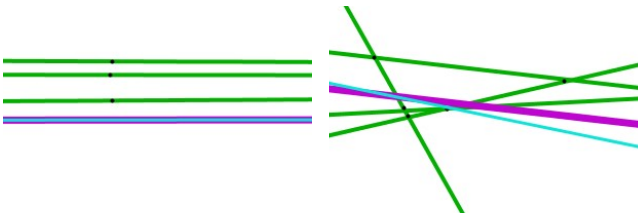

Figure 8: vAC (cyan) over zAC (thick magenta) for four parallel lines (left) and for non-parallel lines (right) oriented towards the right.

The details of *ProjectZ* for projecting onto a zAC of lines is shown in Algorithm 2. Note that we do not require normalization of the computed N.

```
N = (0, 0);
for each line L_i do
    N+ = N_i;
end
for each line L_i do
    n += (PP_i · N_i);
end
d = N · N;
V = P + (n/d)N;
```

Algorithm 2: *ProjectZ* on the zAC of lines

Comparing the vAC and zAC of lines: Fig. 8 shows the vAC and the zAC of a set of four lines in two configurations. Note that the vAC and zAC are identical when all lines are parallel.

5. Trace

Using prior art for tracing the AC: One may consider using previously proposed tracing algorithms to compute the vAC as the valley of $Q(P)$ in the gap.

The concepts of ridges and valleys have applications in non-photo-realistic rendering [30], image analysis [17],

shape recognition [36], and other areas. The literature on extracting valley and ridge curves of height fields is vast. Existing techniques can be classified into sampling methods [32], tracing methods [18] and differential equation solving methods [33]. Many prior techniques rely on a regular sampling of the plane and approximate the height field by a piecewise linear field. To obtain smoother approximations of the valley, one could approximate the field by a B-Spline function and use a variation of the tracing approach of [35].

Similarly, we could produce an approximation of the zAC by sampling $D(P)$ on a regular grid and using zero-set (iso-curve) tracing algorithms (see for example [12]).

Instead of these approaches, we propose below an efficient algorithm (*Trace*) that works from the input curves directly and traces the AC by producing points that lie exactly on the AC. The algorithm is almost identical for the vAC and zAC.

Project: The basic step in our solution is *Project*. It takes as input a candidate point P in the gap. For each C_i, it computes the GRP[4], P_i, of P on C_i. Then, it calls *ProjectV* (Algorithm 1) or *ProjectZ* (Algorithm 2) depending on whether we want the vAC or the zAC. It returns a point V on the vAC or on the zAC of the tangent lines. We view the AC as the set of fixed points of the *Project* operator, applied to points in the gap.

Snap: Given a candidate point P in the gap, *Snap(P)* iterates *P=Project(P)* until the distance between consecutive locations of P falls below a small threshold (we use 10^{-6} of the diagonal of the minimum axis-aligned bounding box containing the input set). The process is illustrated in Fig. 9. We found that 3 iterations suffice to converge in all cases that we tested. Therefore, *Snap(P)* can be implemented in closed form as:

```
Snap(P) { return Project(Project(Project(P)));}
```

Algorithm 3: *Snap*

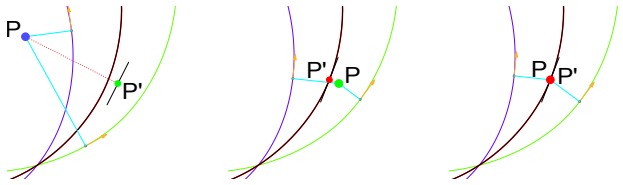

Figure 9: Point V on the AC (red dot) is computed by snapping candidate point P (blue dot) to the AC. Snap converges after only two Project moves. The first move (left) computes the GRPs of P on the input curves (joined by cyan lines) and the associated tangents (orange). We show (short black segment) a portion of the AC of the tangent lines around the projection P' of P onto it. The displacement (P, P') is shown by a thin red line. To move again (center), we set $P = P'$ and repeat the process. A third iteration (right) has no effect on P, since P already lies on the VA (red) of the input curves.

[4]The segment $(P,P_i.)$ is in the gap and is orthogonal to C_i at P_i.

Trace algorithm for computing the PAC: Our *Trace* (Algorithm 4) builds a polygonal approximation (PAC) of the AC incrementally, one vertex at a time. We compute the first (*seed*) vertex of the PAC as **Snap(S)**, where S is the left-most point of the input set. This step requires traversing each curve once, before the tracing of the PAC starts. Other, more robust, solutions may be used. For example, one may consider a small set of seed candidates randomly chosen on the input curves and select the one for which the minimum distances to the other curves have the smallest sum.

Then we repeat the following process. We consider the last vertex U of the PAC (the last one appended). We use the GRPs U_i of U to compute the tangent lines to the input curves and the direction T of their AC using the formulation of the normal N of the AC of lines, as provided in Algorithm 1 or 2[5]. T is the tangent to the AC at U. We compute a candidate point $P = U + eT$ as an extrapolation of U along the tangent to the AC. Then, we compute a new vertex $V = snap(P)$ and appends it to the PAC. It will serve as U for the next iteration.

```
S = left-most point of {C_i}:
PAVA.AppendVertex(Snap(S));
repeat
    U=PAVA.last.vertex();
    {U_i} = GRPs(U);
    T = tangentToACofLines(U,{U_i});
    PAVA.AppendVertex(Snap(U + eT));
until Closed;
```

Algorithm 4: Trace

The sampling density, and hence the number of samples appended to the PAC are controlled by the step size e. For example, we set it to 1/300 of the average length of the input curves if we wish to create an AC with about 300 vertices. To increase performance, we provide the option of increasing e automatically in segments of the gap where the distance from U to its GRP is large, but a large step size may reduce the smoothness of PAC. One may of course subdivide the PAC to produce a smoother curve.

Trace stops (i.e., *Closed* becomes *True*), when the distance from the new vertex V to the first vertex of the PAC is less than e. We test whether V projects onto the interior of the first edge of the PAC. If so, we do not append it.

Incremental closest projection: Each call to *Project* computes the GRPs $\{P_i\}$ of P onto the $\{C_i\}$. Assuming strong compatibility, we do not need to perform a search for P_i over the entire curve C_i. Instead, we start the search at the closest projection U_i of U and stop the search at the first point P_i along C_i where the distance to P reaches a local minimum.

Incremental compatibility check: Assume that U is the last vertex appended so far to the PAC and that

[5]Note that we may not need to recompute U_i and T if these are saved as the results of the previous snap.

V is the new vertex produced by *Snap*. To perform a local compatibility check, for each curve C_i, we verify that the GRP from the line segment (U, V) to the section of C_i between the closest projection U_i of U and the closest projection V_i of V is a homeomorphism.

Justification of correctness: The correctness of *Trace* hinges on the following assumptions: (1) *Snap(S)* of the left-most point S always lies on the AC. (2) Each candidate P lies in the gap. (3) Each candidate P has a GRP on each input curve. (4) The extrapolation $P = U + eT$ followed by *Snap(P)* yields a point further than U along the AC (so that *Trace* always moves forward).

Assumption (1) is true, since S is on the boundary of the gap.

Assumption (2) may fail (for example, when e is large and the gap is turning). We have considered testing whether P lies in the gap and reducing e if it does not, but our experiments indicate that this precaution is not necessary.

Assumption (3) is guaranteed if the input curves are strongly compatible.

We do not have a mathematical proof of (4). Our experiments (Fig. 10) indicate that the basin of attraction of each short segment of the AC (i.e., the set of points of the gap that snap to it) is a thin cross-section of the gap that is nearly straight and orthogonal to the AC.

Parametrization of the gap: This observation suggests a natural parametrization of the gap: a point P of the gap is represented by (s, h) where s is the arc-length parameter along the AC of point V returned by *Snap*(P) and where h is the distance from P to V.

Centroid variant: An interesting variant of the AC may be obtained by replacing each point P of the AC by the centroid, G, of its GRPs. Although the shape of that variant is nearly identical to the AC, it provides a more natural relation between G and the corresponding points points P_i. It may be viewed as an extension of the *Midpoint Locus* variation [5] of the medial axis.

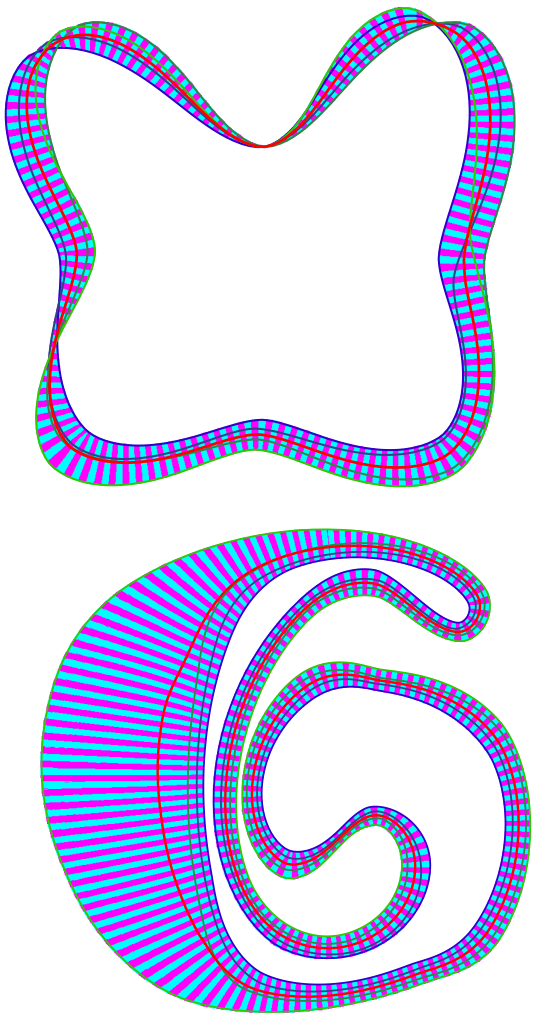

Figure 10: We assign alternating colors to segments of the AC and paint each pixel of the gap with the color of the run that it snaps to.

6. Extensions

In this section, we present simple extensions of the AC to weighted averages and to open curves.

Weighted average of curves: To compute a weighted vAC (resp. zAC) of $\{C_i\}$ with weights $\{w_i\}$ that sum up to 1, we replace, in *ProjectV* (resp. *ProjectZ*), the formulations of the sum of the squared distances (resp. of signed distances) by a weighted sum.

The choice of the weights or their variation with time or some other parameter depends on the application.

For example, this solution may be used to support the design and animation of the morph between two curves. Fig. 11 shows a superposition of a series of frames of such an interpolation. These may also be used as a pattern of curves or as a tool to parametrize the gap.

Our solution extends to morphs defined by more than two curves, for example by using the paradigm proposed in [9] for a Bézier path in the space of polyhedra. But here, instead of using a weighted Minkowski sum, we use a weighted sum of (signed or squared) distance fields and trace the corresponding AC.

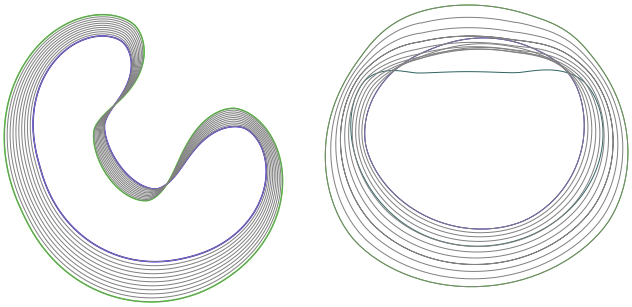

Figure 11: Left: Frames (black) of an interpolating morph between two input curves produced by using a weighted AC. Right: Frames of an animation defined by three control curves using quadratic Bézier weights of $(1-t)^2, 2t(1-t)$ and t^2. As expected, the central curve is not interpolated.

Because scaling and addition of (squared) distance fields behave as their equivalent operators on scalars or points, various formulations developed for motions or (parametric or subdivision) curves defined by control points may be trivially adapted to produce motions or patterns of curves that are defined by several control curves. For example, we can use Bézier weights or time parametrized weights derived from the Neville algorithm [22], so as to ensure that the animation interpolates all control curves with proper timing.

Open curve segments: We have extended the AC to sets of oriented open curve segments (which we call *strokes*). To test this extension, we have developed an interactive system where the user draws the input strokes. We use a light smoothing filter to remove trembling, pixel quantization, and time-dependent sampling artifacts.

We extend each stroke at each end with a half-line that abuts the stroke with tangent continuity. We start with the *Snap* of the centroid of the start points of the strokes. We then perform *Trace* until we pass the *Snap* of the centroid of the end points.

Optionally, one can trim the AC to the part that has a homeomorphic normal map with each input stroke. Examples are shown in Fig. 12.

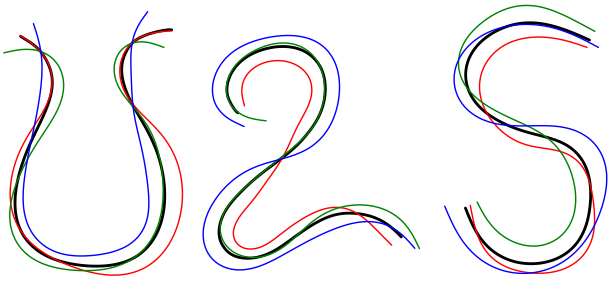

Figure 12: The average curves (black) generated for sets of 3 open curves (red, green, blue).

7. Example application

Statistical analysis of families of shapes is used in medicine to facilitate diagnosis and support research. For example, in [25], the authors study the variability of femur heads (using 3D models reconstructed from CT images) and quantify variations between femurs with and without Cam Femoroacetabular Impingement. As another example, PCA is used in [23] to compute a statistical shape atlas that complements anatomical features (acquired from X-ray scans) to compute patient specific femoral bone shapes.

To illustrate a possible application of the VA in this domain, we used a set of input curves that each approximate the projected silhouettes of a portion of the femoral head. We have traced these curves from images in Fig. 15 of [28]. We compute the average and variability of the first three curves, and also overlay the fourth curve, that corresponds to a complex cam-type deformity, on top of this information (Fig. 13) to show how the anatomy of a patient may be compared to a statistical model of a family of shapes.

Figure 13: Left: The input curves trace the cross-sectional outline of the shape of three instances of a femoral head. We also show their vAC and inflation. Right: Also overlaid is the outline of the shape of a femoral head that corresponds to a complex cam-type deformity. Notice that this femoral head outline is significantly varying from the computed average curve.

8. Implementation

So far, we have presented our definitions, properties, and algorithms assuming that the input curves are smooth. Our algorithms assumed that we can compute the closest projection of point P on any input curve and (to verify compatibility) that we can compute the tight cone of the normal directions spanned by points along any segment of any input curve.

Instead, the implementation used to demonstrate the proposed approach and to produce the illustrations for this paper involves Piecewise Linear Curves (PLC) that closely approximate a set of smooth input curves.

Our implementation follows closely the *Trace* algorithm presented above. We describe below the specific details that have been customized for PLCs.

Spikes of a PLC: Consider three consecutive vertices of C_i: X_i, Y_i, and Z_i. Each vertex Y_i has three spikes: one on the concave side of C_i and two on the convex side. For example, when Y_i is a right turn of C_i, it has one spike on the right and two on the left. All three start at Y_i.

The spike on the right is the bisector of the angle (X_i,Y_i,Z_i). One spikes on the left is orthogonal to edge (X_i,Y_i). The other one to edge (Y_i,Z_i).

Incremental search for the GRP: The GRP P_i of P fall onto a vertex (say Y_i) or onto an edge (say (X_i, Y_i)) of C_i.

To find the first GRP along C_i starting from the projection U_i of the last vertex appended to PAC, we consider, in order, the spikes of the vertices of C_i that follow U_i. We skip the vertices of C_i that have a spike that intersects the segment (U, P) until we find a vertex Y_i for which a spike does not.

Once we have identified that vertex Y_i, we compute P_i. Depending on the local configuration, either $P_i = Y_i$ or P_i lies on edge (X_i, Y_i).

Incremental testing of compatibility: Each time we append a vertex V to the PAC, we perform a local

compatibility test. Let U be the vertex preceding V in the PAC.

For each i, we consider the portion K_i of C_i between the projections U_i of U and V_i of V. For each vertex M of K_i, if M lies on the left of the line L through U and V, we consider its right spike(s), otherwise, its left spike(s). We test that no two consecutive such spikes cross before intersecting L. If the segment passes the test, we have a discrete, normal map, homeomorphism between K and the line segment (U, V). It all segments pass the test, we declare that the segments that correspond to the portion (U, V) of the PAC are compatible.

9. Results

In this section, we report performance measurements and show a variety of results in an attempt to illustrate what types of inputs that produce a valid VA and what inputs are declared incompatible by our implementation of *Trace*.

Convergence of *Snap*: The average number of iterations of P=*ProjectV* that were needed to converge to the AC from an arbitrary point P in the plane was 2.03.

Overall performance: We used a machine with 8 GB of RAM and a 2.2 GHz quad-core processor. Our single threaded implementation performs an average of 15,590 *Snap* operation per second on an input set of 768 vertices. Performance drops to 8,104 operations when we double the vertex count.

Incremental solution: Switcing to the incremental, rather than global, computation for the GRP improved the performance of *Trace* by a factor of 7 when the input has 768 vertices, by a factor of 14 when the vertex count is doubled, and by a factor of 38 when it is quadrupled.

Diversity of input sets handled by *Trace*: We tested *Trace* on a large number of configurations of input curves. A typical example is shown in Fig.1.

Trace works (i.e., yields a Jordan curve that has homeomorphic normal-maps to each input curve) for a variety of configurations, including configurations where the curves are far from similar (see for example Fig. 14).

Since *Trace* performs a local analysis, it also works for configurations that are locally compatible, but where the curves may self-cross once or more (Fig. 15).

Examples of incompatible configurations where *Trace* may fail: *Trace* may declare an input set to be invalid for several reasons. We provide two representative examples.

The *wart* is an example of configurations where the Medial Axis of the gap has a bifurcation. Our construction may fail in such configurations because no AC exists that establishes a normal map homeomorphism to both input curves. The two images in Fig 16 show how the normal map may jump (be discontinuous) when moving along the gap. Although one could construct some form of an "average curve" ignoring such jumps, the result would not capture the local variability of the input curves.

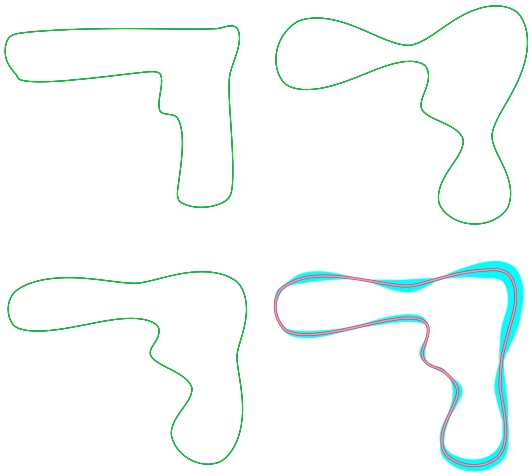

Figure 14: Three input curves (green) and the resulting CA with inflation (cyan).

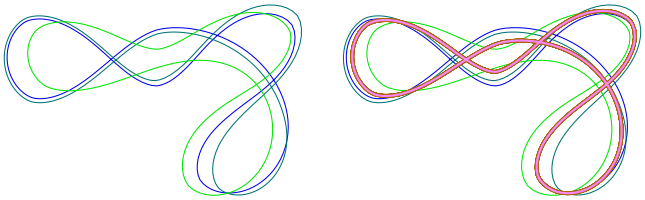

Figure 15: Self-crossing input set (left) and its AC.

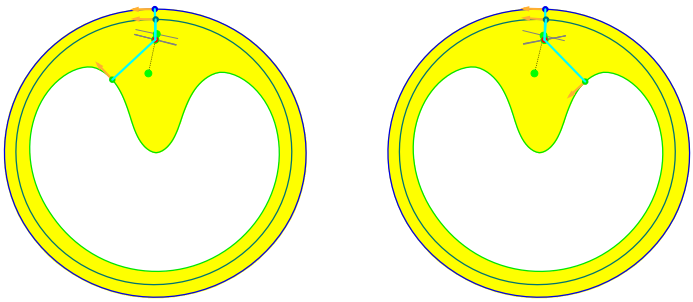

Figure 16: Incompatible configurations (wart): the GRP on the green curve jumps while moving along the gap.

The *GRP confusion* is an example of configurations where the gap is not an annulus. Because our *Trace* algorithm is works on a local section of the gap at a time, it may create a valid AC for some of such configurations, but may fail to start properly (as shown in Fig 17) or be confused when these overlaps are local.

10. Summary and conclusion

In this paper, we report the following contributions. We propose two variants of the Average Curve (AC) of a set of smooth and compatible input curves. The valley Average Curve (vAC) variant is defined in terms of the valley of a height field that is the sum of the squared distances to

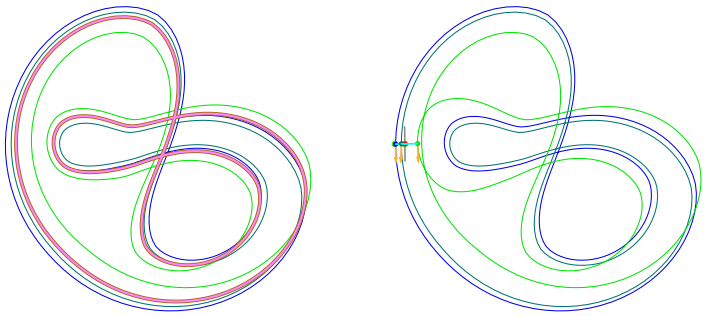

Figure 17: Incompatible configurations (GRP confusion): The AC exists for both configurations, but a particular choice of the seed for computing the first vertex of the PAC (here we picked the left-most vertex of the input curves) leads to a wrong GRP onto the green curve (right).

the input curves. The zero Average Curve (zAC) variant is defined in terms of the zero set of a height field that is the sum of the signed distances to the input curves.

We provide closed form expressions for computing the vAC and zAC exactly when the input curves are lines and for computing the closest projection onto them from an arbitrary point in the plane. We propose an algorithm that traces a polygonal approximation (PAC) of the AC (vAC or zAC). It does not use a regular grid sampling of these height fields. Instead it computes candidate points via tangent extrapolation and snaps them onto the AC using 3 steps that each compute the closest projections onto the input curves, the associated tangent lines, and project the candidate point onto the AC of these tangent lines.

To avoid the cost of a global search for the closest projection and to avoid using erroneous closest projection onto segments of input curves that are closed in the plane but distant along the curve, we define the gap between the curves and the gap restricted projection (GRP).

We define when an input configuration is compatible and also propose a sufficient compatibility condition that may be checked without constructing the AC.

We illustrate the generality of our solution by showing a broad range of valid input configurations, which include input sets where the curves each self-cross once or multiple times.

We provide the details of an implementation of our algorithm designed to work on densely sampled polygonal approximations of smooth input curves. It has linear space and time complexity and is fast enough to provide real-time feedback by re-computing the AC at each frame during the interactive editing of the input curves.

We anticipate that the theoretical contributions and practical implementations described here will impact various fields in metrology, CAD, GIS, and medical modeling by providing an important building block and practical tool for statistical shape analysis.

We also hope that this initial investigation will fuel various extensions. These include formal definitions of the exact AC for coarse polygonal curves, a graceful extension for computing the AC in portions where the curves are not compatible, and extensions of to curves and surfaces in 3D.

11. Acknowledgements

The problem of computing the average of planar curves was first investigated by some of the authors of this paper at the 9^{th} McGill-INRIA Workshop on Computational Geometry. We thank SPM reviewers for excellent suggestions that have helped us to improve the paper and thank Leo Rossignac-Milon for asking whether our initial solution could be extended to weighted averages of curves. Partial support for this project came from the Natural Sciences and Engineering Research Council of Canada.

References

[1] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, January 2003.

[2] Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5:75–91, 1995.

[3] Nina Amenta and Marshall Bern. Surface reconstruction by Voronoi filtering. *Discrete & Computational Geometry*, 22(4):481–504, 1999.

[4] Nina Amenta and Yong Joo Kil. Defining point-set surfaces. *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH*, page 270, 2004.

[5] Haruo Asada and Michael Brady. The curvature primal sketch. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (1):2–14, 1986.

[6] H Blum. A transformation for extracting descriptors of shape. 1967.

[7] Kevin Buchin, Maike Buchin, Marc Van Kreveld, Maarten Löffler, Rodrigo I Silveira, Carola Wenk, and Lionov Wiratma. Median trajectories. *Algorithmica*, 66(3):595–614, 2013.

[8] Frédéric Chazal, André Lieutier, Jarek Rossignac, and Brian Whited. Ball-map: Homeomorphism between compatible surfaces. *International Journal of Computational Geometry and Applications*, 20:285–306, 2010.

[9] Jarek Rossignac and Anil Kaul. AGRELs and BIPs: Metamorphosis as a Bézier Curve in the space of polyhedra. *Computer Graphics Forum*, 13(3):179–184, 1994.

[10] Frédéric Chazal, André Lieutier, and Jarek Rossignac. Projection-homeomorphic surfaces. In *Proceedings of the 2005 ACM symposium on Solid and Physical Modeling*, pages 9–14. ACM, 2005.

[11] Frédéric Chazal, André Lieutier, and Jarek Rossignac. Normal-map between normal-compatible manifolds. *Int. J. Comput. Geometry Appl.*, 17(5):403–421, 2007.

[12] Antoni Chica, Jason Williams, Carlos Andújar, Pere Brunet, Isabel Navazo, Jarek Rossignac, and Àlvar Vinacua. Pressing: Smooth isosurfaces with flats from binary grids. In *Computer Graphics Forum*, volume 27, pages 36–46. Wiley Online Library, 2008.

[13] Jonathan M. Cohen, Lee Markosian, Robert C. Zeleznik, John F. Hughes, and Ronen Barzel. An interface for sketching 3d curves. In *Proceedings of the 1999 Symposium on Interactive 3D Graphics, SI3D*, pages 17–21, 1999.

[14] Ge Cong and Bahram Parvin. A new regularized approach for contour morphing. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 458–463. IEEE, 2000.

[15] 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.

[16] Adrian Dumitrescu and Günter Rote. On the Fréchet distance of a set of curves. In *CCCG*, pages 162–165, 2004.

[17] David Eberly. *Ridges in image and data analysis*, volume 7. Springer, 1996.

[18] David Eberly, Robert Gardner, Bryan Morse, Stephen Pizer, and Christine Scharlach. Ridges for image analysis. *Journal of Mathematical Imaging and Vision*, 4(4):353–373, 1994.

[19] Alon Efrat, Quanfu Fan, and Suresh Venkatasubramanian. Curve matching, time warping, and light fields: New algorithms for computing similarity between curves. *Journal of Mathematical Imaging and Vision*, 27(3):203–216, 2007.

[20] Shaw C Feng and Yuhwei Yang. A dimension and tolerance data model for concurrent design and systems integration. *Journal of Manufacturing systems*, 14(6):406–426, 1995.

[21] Henry Fuchs, Zvi M. Kedem, and Samuel P. Uselton. Optimal surface reconstruction from planar contours. *Communications of the ACM*, 20(10):693–702, 1977.

[22] Ron Goldman. *Pyramid algorithms: A dynamic programming approach to curves and surfaces for geometric modeling*. Morgan Kaufmann, 2002.

[23] Jerome Grondin Lazazzera. Predicting femoral geometry from anatomical features. 2014.

[24] Sariel Har-Peled and Benjamin Raichel. The Fréchet distance revisited and extended. *ACM Trans. Algorithms*, 10(1):3:1–3:22, January 2014.

[25] Michael D Harris, Manasi Datar, Ross T Whitaker, Elizabeth R Jurrus, Christopher L Peters, and Andrew E Anderson. Statistical shape modeling of cam femoroacetabular impingement. *Journal of Orthopaedic Research*, 31(10):1620–1626, 2013.

[26] Trevor Hastie and Werner Stuetzle. Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989.

[27] Gabor T Herman, Jingsheng Zheng, and Carolyn A Bucholtz. Shape-based interpolation. *IEEE Computer Graphics and Applications*, 12(3):69–79, 1992.

[28] N. Yanguas J. Salceda Artola I. Sanmartin R. M. Cozcolluela Cabrejas J. M. Mellado, I. Quintana. Normal anatomy and variations of the proximal femur: A reappraisal with 64-slice mdct. 2014.

[29] Jr Chester W Jackson, Clark R Alexander, and David M Bush. Application of the AMBUR R package for spatio-temporal analysis of shoreline change: Jekyll island, Georgia, USA. *Computers & Geosciences*, 41:199–207, 2012.

[30] Tilke Judd, Frédo Durand, and Edward Adelson. Apparent ridges for line drawing. In *ACM Transactions on Graphics (TOG)*, volume 26, page 19. ACM, 2007.

[31] Anil Kaul and Jarek Rossignac. Solid-interpolating deformations: Construction and animation of pips. pages 107–115, 1992.

[32] Gordon L Kindlmann, Raúl San José Estépar, Stephen M Smith, and C-F Westin. Sampling and visualizing creases with scale-space particles. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):1415–1424, 2009.

[33] Antonio M López and Joan Serrat. Tracing crease curves by solving a system of differential equations. In *Computer VisionECCV'96*, pages 241–250. Springer, 1996.

[34] Greg Mori, Serge Belongie, and Jitendra Malik. Efficient shape matching using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1832–1837, 2005.

[35] Suraj Musuvathy, Elaine Cohen, James Damon, and Joon-Kyung Seong. Principal curvature ridges and geometrically salient regions of parametric b-spline surfaces. *Computer-Aided Design*, 43(7):756–770, 2011.

[36] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. Ridge-valley lines on meshes via implicit surface fitting. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 609–612. ACM, 2004.

[37] Ronald Peikert and Filip Sadlo. Height ridge computation and filtering for visualization. In *Visualization Symposium, 2008. PacificVIS'08. IEEE Pacific*, pages 119–126. IEEE, 2008.

[38] Jarek Rossignac. Ball-based shape processing. In *Discrete Geometry for Computer Imagery*, pages 13–34. Springer, 2011.

[39] Jaroslaw Rossignac and Aristides Requicha. Constant-radius blending in solid modelling. 1984.

[40] Günter Rote. Computing the Fréchet distance between piecewise smooth curves. *Computational Geometry*, 37(3):162 – 174, 2007. Special Issue on the 20th European Workshop on Computational Geometry {EWCG} 2004 20th European Workshop on Computational Geometry.

[41] U Roy, CR Liu, and TC Woo. Review of dimensioning and tolerancing: representation and processing. *Computer-aided design*, 23(7):466–483, 1991.

[42] William Rucklidge. *Efficient visual recognition using the Hausdorff distance*, volume 1173. Springer Heidelberg, 1996.

[43] Martin Rumpf and Benedikt Wirth. A nonlinear elastic shape averaging approach. *SIAM Journal on Imaging Sciences*, 2(3):800–833, 2009.

[44] Scott Schaefer, Travis McPhail, and Joe Warren. Image deformation using moving least squares. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 533–540. ACM, 2006.

[45] Thomas W Sederberg and Eugene Greenwood. A physically based approach to 2–d shape blending. In *ACM SIGGRAPH Computer Graphics*, volume 26, pages 25–34. ACM, 1992.

[46] Heiko Seim, Dagmar Kainmueller, Markus Heller, Hans Lamecker, Stefan Zachow, and Hans-Christian Hege. Automatic segmentation of the pelvic bones from CT data based on a statistical shape model. In *VCBM*, pages 93–100, 2008.

[47] Kaveh Shahbaz. Applied similarity problems using Fréchet distance. *arXiv preprint arXiv:1307.6628*, 2013.

[48] Kaleem Siddiqi and Stephen Pizer. *Medial representations: mathematics, algorithms and applications*, volume 37. Springer Science & Business Media, 2008.

[49] Robert W Sumner, Matthias Zwicker, Craig Gotsman, and Jovan Popović. Mesh-based inverse kinematics. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 488–495. ACM, 2005.

[50] Robert B. Tilove. Set membership classification: A unified approach to geometric intersection problems. *Computers, IEEE Transactions on*, 100(10):874–883, 1980.

[51] Greg Turk and James F O'brien. Shape transformation using variational implicit functions. In *ACM SIGGRAPH 2005 Courses*, page 13. ACM, 2005.

[52] Herbert Weisberg. *Central tendency and variability*. Number 83. Sage, 1992.

[53] Brian Whited and Jarek Rossignac. Relative blending. *Computer-Aided Design*, 41(6):456–462, June 2009.

[54] Brian Whited and Jarek Rossignac. Ball-morph: Definition, implementation, and comparative evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 17(6):757–769, June 2011.