Interacting with Self-Similarity

Josef Graus[a], Alec Jacobson[b], Yotam Gingold[a]

[a]*Department of Computer Science, George Mason University*
[b]*Department of Computer Science, University of Toronto*

Abstract

Shape similarity is a fundamental problem in geometry processing, enabling applications such as surface correspondence, segmentation, and edit propagation. For example, a user may paint a stroke on one finger of a model and desire the edit to propagate to all fingers. Automatic approaches have difficulty matching user expectations, either due to an algorithm's inability to guess the scale at which the user is intending to edit or due to underlying deficiencies in the similarity metric (e.g., semantic information not present in the geometry).

We propose an approach to interactively design self-similarity maps. We investigate two primitive operations, useful in a variety of scenarios: region and curve similarity. Users select example similar and dissimilar regions. Starting with an automatically generated multi-scale shape signature, our approach solves for a scale parameter and thresholds that group the example regions as specified. We propose a new Smooth Shape Diameter Signature (SSDS) as a more efficient alternative to the Heat or Wave Kernel Signature. If no such parameters can be found, our approach modifies the shape signature itself. Given a curve drawn on the surface, we perform hybrid discrete/continuous optimization to find similar curves elsewhere.

We apply our approach for interactive editing scenarios: propagating mesh geometry, patterns duplication, and segmentation.

Keywords: shape analysis, signature, interaction, similarity

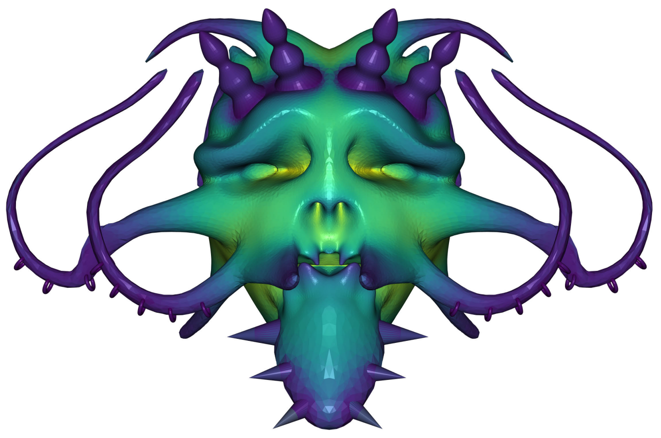
Figure 1: Our Smooth Shape Diameter Signature (SSDS) visualized on the sea monster model [1].

1. Introduction

Shape similarity is a well-studied, fundamental problem in geometry processing. Its importance stems from the necessity of establishing comparative surface metrics to drive a variety of mesh processing algorithms, such as edit propagation, attribute transfer, search, correspondence, and segmentation [2]. Many similarity approaches are based on computing and comparing point-wise shape "signatures" or feature descriptors. These signatures may be based on intrinsic or extrinsic properties on the mesh. Over the last decade, a popular vein of signatures emerged which offer a multi-scale view of the meshes they're defined upon, commonly through some scale-parameterizable approximation of a physical phenomenon. Many of these signatures involve the spectrum of the Laplace-Beltrami operator [3, 4]. Once proposed, it is often left to future work to create a framework to best leverage the specific properties of each class of these signatures.

Often overlooked is how these shape signatures can be interactively controlled for applications such as mesh segmentation and edit transfer at application-appropriate scales. Often, the scale best suited to a user's desired correspondence cannot be automatically determined. This is due to either an algorithm's inability to guess the scale at which the user is intending to edit or to underlying deficiencies in the automatic similarity metric (e.g., semantic information is not present in the geometry).

In this work, we focus on the interactive design of self-similarity maps, or finding self-similar regions on a single manifold mesh. We base our self-similarity maps on multi-scale shape signatures that define a scale-parameterized signature value at each vertex, thereby capturing local information when desirable or neglecting their contribution when not. We investigate two operations, propagating selected surface regions and hand drawn surface curves.

Our **contributions** are:

- An approach to allow users to interactively obtain desired self-similarity maps by automatically finding scale parameters and signature thresholds—and modifying the underlying shape signature when necessary. Our interactive step is intuitive since the user merely labels areas they consider similar and, optionally, dissimilar. We provide immediate visual feedback, so the user can provide minimal input,

*Corresponding author
Email address: jgraus@gmu.edu (Josef Graus)

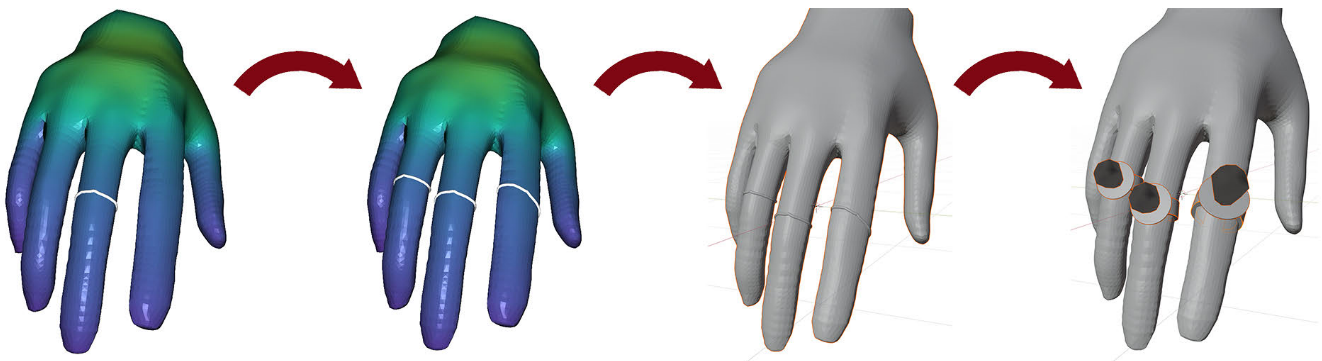

Figure 2: An illustration of geometry propagation using our curve propagation: (Left to right) (1) A single ring is drawn on a finger of the hand; (2) Additional ring placements are suggested; (3) The ring placements are exported to Blender; (4) Complex geometry (ring pops) are placed and aligned according to the exported curves (manually, as an illustrative application).

stopping when satisfied with the displayed result. Exception perturbations are employed to successfully establish thresholds even when there is no scale which brings user labeling into agreement. These established relationships between mesh regions are made part of any subsequent shape signature query, including generating other similarity maps on the same mesh.

- The Smooth Shape Diameter Signature (SSDS). While our approach for choosing shape signature thresholds is agnostic to a large class of shape signatures, the spectral signatures available in the literature have prohibitive computational requirements. We propose a signature based on smoothing the Shape Diameter Function [5] as an alternative informative similarity metric with significantly better time complexity.

- A method for self-similar curve propagation based on tangent-space curve unrolling for distortion-free surface parameterization.

We show the effectiveness of our method in a set of edit propagation and detail synthesis routines.

2. Related Work

Edit Propagation.. The most closely related works to ours focus on edit propagation. Zelinka et al. [6] use geodesic fans to compute surface similarity. They show edit propagation with a simple user-defined α threshold as a means to determine which vertices on the model are impacted by the edit of a "similar enough" mesh vertex. In contrast, we provide a robust automatic method for determining parameters matching user intent and the ability to propagate curve features as well as similar regions. SimSelect [1] presented approaches for propagating surface selections. Their approach is based on isolines of a harmonic fields. We focus on largely orthogonal tasks: selecting parameters and thresholds for multi-scale signature functions and propagating open curves. Peng et al. [7] presented an approach for "autocompleting" 3D curves drawn on the surface of an object. However, their approach is based on analyzing a set of strokes with respect to each other, not on surface properties. Mitra et al. [8] registers similarities in a transformation space, propagating alignments to spatial geometry to further refine a varying scale of symmetries.

Shape Signatures.. Our work centers largely around interpreting, comparing, and manipulating shape signature values at vertices in a mesh. Rustamov [9] uses *d2* distributions to characterize the Global Point Signature values across a mesh for model-to-model comparison, showing a discernible difference between a variety of different models, while the same model with major and minor deformations would cluster together under the metric. Sun et al. [10] (and, simultaneously, Gębal et al. [11]) address multi-scale matching with the Heat Kernel Signature, side stepping the issue of rapid convergence at large time values t, by uniformly sampling values over the logarithmic scaled temporal domain of the signature, taking the L_2-norm of the resultant vectors as a dissimilarity metric. Ovsjanikov et al. [12] follow-up on this work by describing a technique for point correspondence propagation and symmetry detection by identifying and registering a small number local maxima points of the Heat Kernel Signature within and across whole and partial models. Dey et al. [13] also follow this maxima registration approach, but with a focus on incomplete model matching, and with an added step of merging maxima regions down to a total feature count κ. Aubry et al. [14] explores energy rather than temporal intervals with a Wave Kernel Signature; this, too, creates a feature vector of logarithmic scaled energies and computing the $L2$-norm of these feature vectors between points to determine matching. These works tend to focus on the single smallest dissimilarity when ascertaining a match, with little being said about the range of dissimilarity or how to discern "similar enough" clusterings. We provide this determination through our proposed interactive method, which is agnostic to the choice of scale-varying shape signature. We prefer a signature based on smoothing the Shape Diameter Function [5], which can be computed much more efficiently.

Gal et al.[15] also establish a framework for determining similarity between mesh regions for a variety of applications, including alignment and symmetry detection. However, they explore self-similarity using a purely geometric definition of what makes two parts of a mesh similar. Our approach consider similarity within the context of a creative process, where purely geometric definitions may be incompatible with user intention, perhaps resulting from semantic information known only to the user. Our framework allows users to edit self-similarity through interaction; we support various single-parameter signatures (SSDS, HKS, WKS, etc.).

Mesh Segmentation.. The goal of a signature thresholding scheme is the ability to discern one region from another across a mesh. There exist a selection of state-of-the-art methods that will deliver segmentations based on metrics ranging from surface curvature [16] to volume uniformity [17]. However, user studies have shown that none of them accurately emulate the segmentation choices of a human user across various classes of shapes and models [18, 19], or unable to adapt to unusual user segmentation desires [20]. We address this issue by opting for semi-interactive segmentation methods that rely on some sort of face/vertex seeding or initial guess, supplying the necessary selections through our signature thresholds applied to the surface. To that end, we seek to extend interactive segmentation algorithms by replacing a substantial portion of the user-driven element with feedback from our signature thresholds established from a small set of user guidance. Recent work in the same vein has turned to seeding the mesh with a smattering of user or random selections and growing component regions from those initial guesses. Lai et al. [21] do random seeding to initialize random walks in a probabilistic search for segment edges. In a survey of mesh segmentation methods, Shamir [22] surveys a number of interactive segmentation techniques largely centered around finding optimal cuts from user suggestion strokes [23, 24]. Zöckler et al. [25] outline a method where interactive user segmentation and selection of feature points are used to drive a patch-wise parameterization and subsequent geometric synthesis routine which morphs models. However, their method relies heavily on the user to establish correspondences and split the mesh into reasonable components. We find the most natural interactive segmentation method to modify for our purposes is that proposed in Surfacing by Numbers [26], wherein a user indicates by surface selection two distinct areas of the mesh, and the segmentation is solved as a min-cut problem between sources and sinks defined by the labeled user strokes. We define each connected cluster within our threshold as attached to source nodes, and define segments by the components disconnected by the min-cut, which is further detailed in Section 6.1.

Curves on Surfaces.. Central to our list of applications is the manipulation of direct-on-mesh drawn curves which can be freely and inexpensively transformed in terms of a lower-dimensional surface parameterization to yield intuitive higher-dimensional results. Luckily, mesh surface 2D parameterization is a well-trodden field with a variety of novel methods readily available either for direct use or inspiration. As it is important to preserve the geodesic fidelity of a surface-drawn curve, we chose to focus on parameterizations that either severely reduced or entirely eliminated triangle distortion, even if that meant sacrificing global injectivity. We were also only interested in high fidelity of parameterizations *local* to the drawn curve. Subsequently, prior works such as discrete exponential maps [27, 28] and bounded-distortion piece-wise mesh parameterization [29] stood as the most promising foundations from which to work. Within this context, our lower-dimensional curve operations most closely resemble snakes on surfaces (or active contours) [30], however our transformations *always* result in a curve whose discretized representation is directly on the 3D surface, and does not require any sort of smoothness energy minimization to maintain this guarantee. We also generalize a curve's minimizable energy across a class of permissive shape signatures. Our edit propagation method is distinct from ones similar to Pauly et al. [31] in that we are not constrained to subgroups of symmetries by construction, allowing for more versatility in application. Furthermore, because we do not rely on a formal learning process, we overcome limitations like those present in Digne et al. [32], whereby an infrequent feature would not be represented in the shape dictionary. Finally, we side-step assumptions of regularity in the underlying geometry [33], allowing similarities to be established beyond purely geometric considerations.

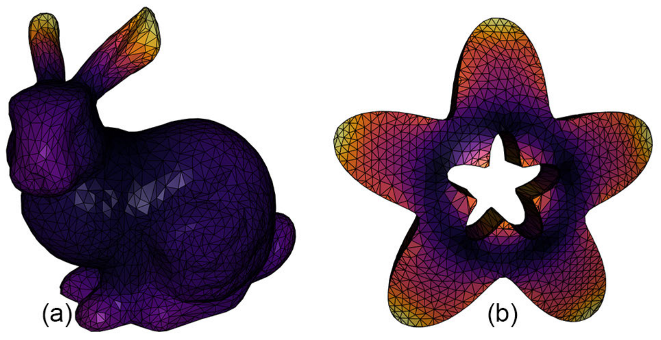

Figure 3: The Heat Kernel Signature visualized on the (a) Stanford Bunny and (b) Trim-Star at arbitrary temporal scales.

3. Overview

Problem Statement. Given a mesh and either a labeled region or a user-drawn curve on the mesh surface, we seek to propagate the user's creative intent to the remainder of the mesh. We also seek to learn from user labels of these indicated regions of interest to improve the quality of future edit suggestions.

Example scenario A. The user wishes to edit all fingers but not toes. When the user selects one finger, all fingers and toes become selected due to their similarity. The user then labels a toe as dissimilar. All toes become unselected.

Example scenario B. The user wishes to add rings to all the fingers of a character's hands except for the index finger. When the user draws a ring about one finger with a surface-bound curve, other fingers will show recommended edits based upon their similarity. The user labels one index finger suggestion as dissimilar; both index fingers become deselected. The user edits the fingers (e.g., ring placement). In the future, the dissimilarity of the index finger from the others will be remembered. If the user selects a finger, all fingers except for the index finger will be automatically selected.

Approach. We base our algorithm around finding an appropriate scale and signature thresholds for scale-varying per-vertex signatures. Our technique is agnostic to the choice of signature and relies only on there being a scale-varying per-vertex signature value. If appropriate thresholds cannot be found that match user input, we update the signature values to ensure that the distinction is remembered for future selections.

With a specific task in mind (e.g., sharpening the points of a star), the user selects faces of a manifold triangle mesh and labels them as either similar or dissimilar. For each set of labels, we classify all vertices of the mesh as similar or dissimilar, even when the user's labels conflict with the underlying shape signature. We display the similarity map as the user is labeling, allowing the user to iteratively refine the labeling until they are satisfied with the result.

While a user is interactively labeling and until they are satisfied with the result, our algorithm for classifying vertices should be independent of the user's labeling order. We believe this predictability is more intuitive for users.

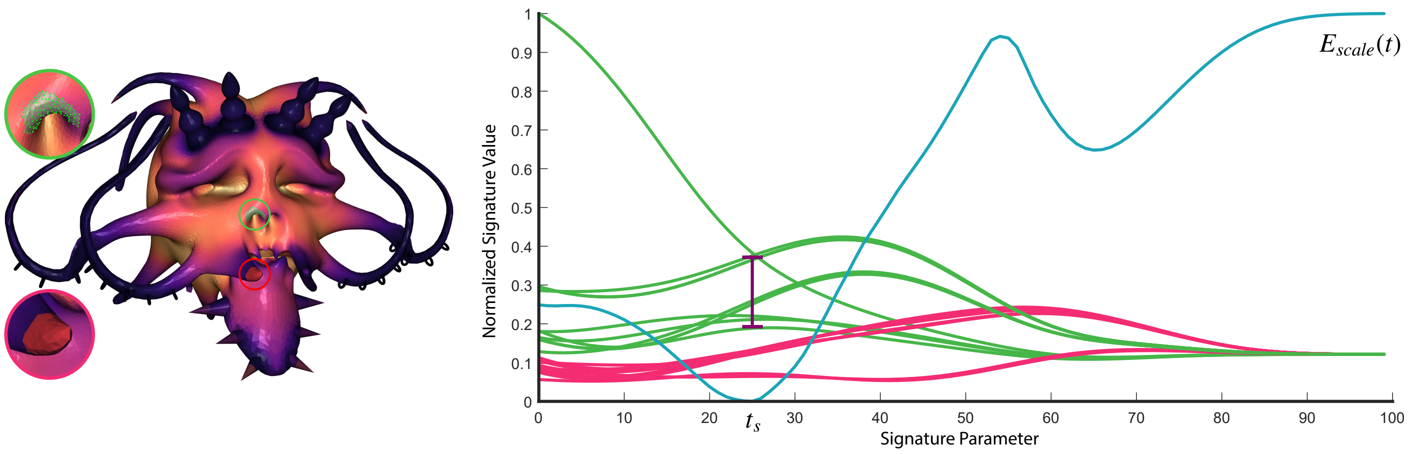

Figure 4: Left: A model with a region labeled similar (shaded green) and dissimilar (shaded red). The selected regions are shown zoomed in the circles at top- and bottom-left. Right: The signature values sampled at uniform logarithmic steps of the scale parameter. The green curves represent similar (green) vertices as their signature values vary with the scale parameter. The red curves represent dissimilar (red) vertices. The teal curve is the normalized objective function $E_{scale}(t)$. The minimum of the objective function t_s corresponds to the scale where green curves and red curves are clustered separately by label. The blue bar spanning the green curves at t_s displays the signature interval computed for this example.

Once the user is satisfied with the similar/dissimilar classification, the signature should be updated if necessary for future selection operations. Our understanding of similarity across the mesh may have indelibly shifted. This would translate, from a user's perspective, into a reduction of guidance necessary to establish a desired similarity map.

For the purposes of this paper, we restrict our attention to signature thresholding on a single model represented by a single manifold triangle mesh. Extending this work to make threshold comparisons across multiple models would require that the underlying signature be commensurable across models.

Shape Signatures. The method we propose can be applied to any per-vertex signature with a single scalar parameter $S(x,t)$, where x is a point on the shape and t is the scalar parameter. We experimented with signatures whose scalar parameter corresponds to scale: the Heat Kernel Signature [10, 11], the Wave Kernel Signature [14], and a new signature we call the Smoothed Shape Diameter Function (3.1), based on smoothing the Shape Diameter Function [5].

The Heat Kernel Signature is a restriction on the heat kernel as it models the diffusion of heat through vertices of the mesh over some temporal scale. Given x, y points on the underlying manifold, the heat kernel $k_t(x,y)$ can be thought of as the amount of heat transferred from x to y over time t. The Heat Kernel Signature fixes the heat kernel to only consider changes over the temporal domain, defining the signature as,

$$\mathrm{HKS}(x,t) = k_t(x,x) \tag{1}$$

where the fixed heat kernel can be estimated by the eigendecomposition of the Laplace-Beltrami operator taken over a mesh of n vertices,

$$k_t(x,x) = \sum_{i=0}^{n} e^{-\lambda_i t} \phi_i(x)^2 \tag{2}$$

where λ_i and ϕ_i are the eigenvalues and eigenfunctions respectively. Figure 3 shows the Heat Kernel Signature at an arbitrary temporal scale on two familiar meshes.

Similarly, the Wave Kernel Signature describes the average probability of measuring a particle at a given location, with probability values varying by the energy assigned to the particle. Assuming a normal distribution, the signature is defined as,

$$WKS(x,e) = C_e \sum_k \phi_k^2(x)^2 e^{\frac{-(e-logE_k)^2}{2\sigma^2}} \tag{3}$$

where C_e is a scale factor and E_k is the given energy. HKS and WKS are nearly equivalent in computational cost and surface evaluation, so we primarily compare to HKS, unless otherwise noted.

Unfortunately, spectral signatures are impractical for large meshes. They require the computation of eigenvalues and eigenvectors, whose time complexity is $O(n^3)$ for n vertices. Computing only a fraction of the eigenvalues and eigenvectors is still costly in time. Experimentally, computing $\frac{1}{3}$ of them is as costly as computing all of them. This is compounded by the fact that the needed fraction is difficult to establish. In our experiments, sometimes 20% and sometimes 80% were needed to accurately reproduce the HKS. Given these limitations, we devised a more efficient signature.

3.1. Smoothed Shape Diameter Signature

The Shape Diameter Function $SDF(x)$ [5] computes the average length of a ray cast from x into the interior of the shape. We define the *Smoothed Shape Diameter Signature* $SSDS(x,t)$ to be the function that minimizes the weighted combination of an energy measuring functional smoothness and an energy measuring fidelty to the raw *SDF* value:

$$SSDS(x,t) = \underset{f}{\operatorname{argmin}} \int_{\Omega} \left((1-t)(f(x) - SDF(x)) + t\|H_f(x)\|_F^2 \right) dx$$

where $\|H_f(x)\|_F^2$ is the squared Hessian energy with natural boundary conditions [34]. An example of the Smoothed Shape Diameter Signature is shown in Figure 1.

Computing the SSDS has time complexity $O(kn\log n + n^{\sim 1.5})$, where k is the number of rays shot per vertex and the ~1.5 comes from the linear solve for the near-biharmonic system. This is much faster than the $O(n^3)$ required for a spectral signature.

We provide pseudocode for SSDS in Algorithm 1, with implementations of subroutines available in libigl [35] and Eigen.

Algorithm 1 Computing SSDS for a mesh with vertices V, faces F, and per-vertex normals N.

```
 1: $a \leftarrow 100$ {SSDS granularity}
 2: $M \leftarrow \texttt{MassMatrix}(V, F)$
 3: $H \leftarrow \texttt{HessianEnergy}(V, F)$
 4: $S \leftarrow \texttt{ShapeDiameterFunction}(V, F, N)$
 5: $SSDS \leftarrow \mathbf{0}_{\#V\times a}$
 6: for $i \leftarrow 0$ to a do
 7:     $t \leftarrow i \div a$
 8:     $Q \leftarrow (1-t)M + tH$
 9:     $SSDS_{*,i} \leftarrow \texttt{LDLTSolve}(Q, (1-t)MS)$
10: end for
11: return $SSDS$
```

4. Self-Similarity Map Design

Given a manifold triangle mesh, we allow the user to design a self-similarity map. The user selects regions they consider to be *similar*. We propagate that selection using shape signatures. If the user is dissatisfied with the propagated selection, they can select undesired regions and explicitly label them as *dissimilar* to their desired selection. For each successive region the user labels, the following sequence of steps are performed:

1. The label is propagated to all vertices in the region.
2. An optimal scale parameter t_s is found for our shape signature at which the vertices labeled as similar and dissimilar are maximally separated (Section 4.1).
3. The shape signature of all mesh vertices is sampled at scale parameter t_s. A signature value interval is created to encompass vertices labeled similar.
4. If the interval also contains vertices labeled dissimilar, we apply a small perturbation to such vertices' signatures to *shift* them out of the interval. We apply the same perturbation to other unlabeled mesh vertices with approximately the same signature values.

The resulting interval contains all vertices labeled as similar; the scale parameter and interval bounds are the similarity thresholds we use to characterize other vertices on the mesh. In the proceeding subsections we focus on a single set of relations, composed of n similar and m dissimilar vertices, where $n > 0$ and $m \geq 0$, unless otherwise noted.

4.1. Parameter search

Different regions of a mesh will look similar or dissimilar at different scales. Given a set of vertices labeled as similar or dissimilar, we wish to find a single scale parameter value that groups the vertices according to their labels. To do so, we pose the choice of scale parameter t as a global optimization problem. We minimize an objective function on the signature scale parameter given our set of relations:

$$t_s = \arg\min_t E_{scale}(t) = \frac{E_{similar}}{E_{dissimilar}} \qquad (4)$$

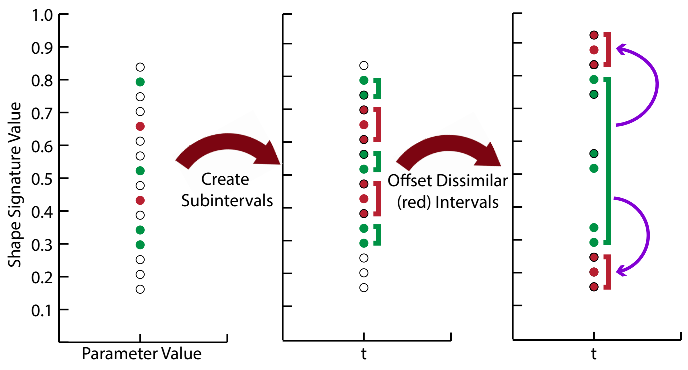

Figure 5: An illustration of signature value perturbation. Left: The signature values of a set of vertices at scale t. Vertices labeled similar are shown in green and dissimilar in red. Unlabeled vertices are shown in white. Middle: Unlabelled vertices are assigned the label of their closest labelled neighbor. Right: The dissimilar sub-intervals are offset to remove them from the larger interval around all similar (green) vertices, producing a consistent classification.

where,

$$\begin{aligned} E_{similar} &= 1 + \sum_{x,y\in \text{similar}} |S(x,t) - S(y,t)| \\ &\quad + \sum_{x,y\in \text{dissimilar}} |S(x,t) - S(y,t)| \\ E_{dissimilar} &= 1 + \sum_{\substack{x\in\text{similar}\\ y\in\text{dissimilar}}} |S(x,t) - S(y,t)| \end{aligned}$$

E_{scale} is defined to be the ratio of relations we wish to cluster together against relations between which we wish to maximize separation. There are two ways to reduce the ratio: driving the numerator $E_{similar}$ to one, which requires tightly clustering groupings of similarly labeled relations, and driving the denominator $E_{dissimilar}$ towards infinity, which necessitates increasing the distances between dissimilar relations. This is exactly the desired behavior for our signature interval. Therefore, by minimizing our objective function, we determine the optimal scale parameter for thresholding unlabeled patches. Figure 4 illustrates the objective function applied to a set of relations.

Since our objective function is a 1-dimensional optimization problem, we obtain a coarse solution for t_s by sampling the scale space 100 times. For the Smoothed Shape Diameter Signature, we uniformly sample the range $[0, 1]$. For the Heat Kernel Signature, we logarithmically sample the interval suggested by Sun et al.: $[4\ln 10/\lambda_k, 4\ln 10/\lambda_2]$, where k is the number of mesh vertices and λ_i are eigenvalues of the Laplace-Beltrami operator. We refine this coarse solution by using it as the initial guess to an L-BFGS-B [36] solver, with the same lower and upper bounds, to obtain our final scale parameter value t_s.

4.2. Thresholding

Having determined the scale parameter t that is most consistent with the user's labeling of various mesh patches, we proceed to construct our similarity thresholds. We set the interval's minimum and maximum signature values to the minimum and maximum signature value from the set of all vertices labeled similar: $[I_{\min}, I_{\max}]$, where $I_{\min} = \min_{x\in\text{similar}} S(x,t)$ and $I_{\max} = \max_{x\in\text{similar}} S(x,t)$.[1]

[1]If there is only a single vertex labeled similar, the above interval collapses to a single value. No other vertices' signatures values will be inside, which is

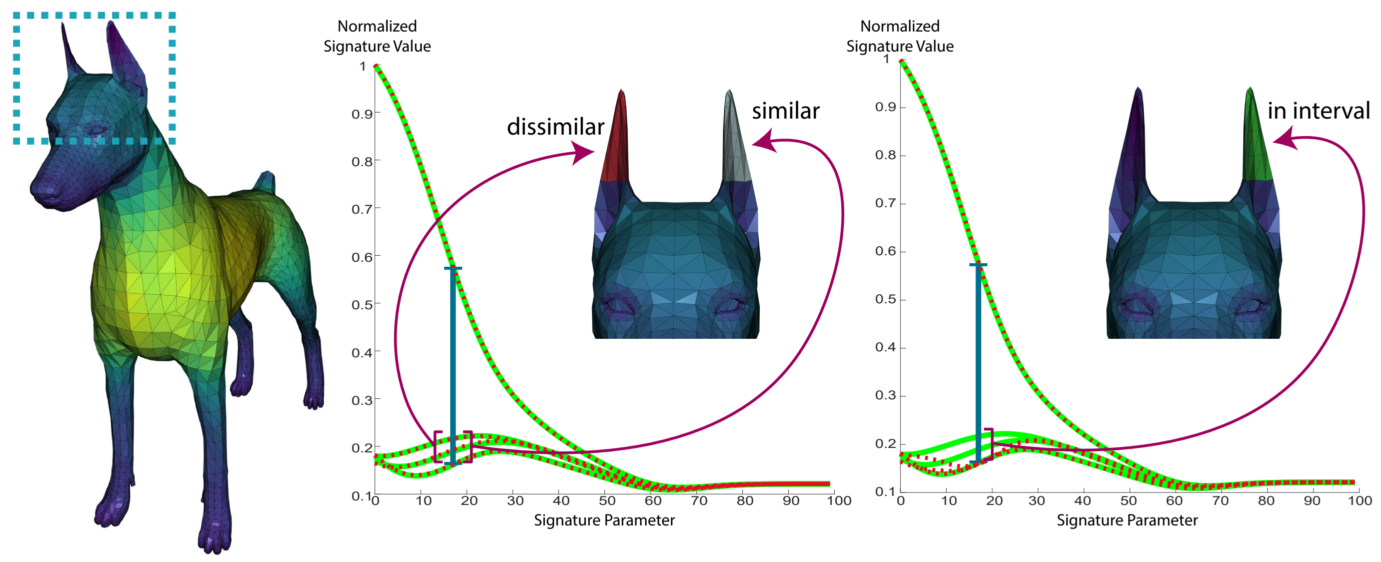

Figure 6: Signature threshold perturbations applied to two regions that share a high degree of symmetry (the dog's ears). Left plot: The right ear is labeled similar and the left ear as dissimilar. Our objective function selects a scale parameter and initial interval for the similar regions. Right plot: Dissimilar vertices within the interval are shifted out with perturbations that fall off smoothly to nearby scale parameters. The ear labeled similar remains within the interval.

At this point, there may still be vertices labeled dissimilar in $[I_{\min}, I_{\max}]$. To remain consistent with user guidance, we modify the signature function of such vertices—as well as unlabeled vertices with signature values "close" to the value of the dissimilar vertex—to remove them from the interval. We add the smallest perturbation to all such vertices' signatures that pushes them out of the interval (Figure 5). This perturbation is a scalar impulse at scale t. In Section 4.3, we smooth the falloff to nearby scale values.

Let Q be the set of vertices labeled dissimilar that lie within the signature interval $[I_{\min}, I_{\max}]$. We wish to adjust the signatures of all vertices in Q so that they no longer lie within the interval. We also wish to adjust the signatures of any unlabeled vertices with signatures closer to a vertex in Q than to a vertex labeled similar. For each vertex $v \in Q$, we find the smallest absolute difference d_v from its signature to the two interval endpoints $I_{\min}$ and $I_{\max}$. We also find the smallest absolute difference w_v between v's signature and any vertex labeled similar. Any *unlabeled* vertex with a signature value within $\frac{w_v}{2}$ of v is closer to v than a similar vertex and so should also be adjusted. We push the dissimilar vertex v out of the similar signature interval by adjusting its signature. We add (or subtract) the offset $o_v = d_v + \frac{w_v}{2} + \epsilon$, where $\epsilon = 1e^{-7}$ keeps interval boundaries from overlapping We also add (or subtract) the same offset o_v to unlabeled vertices whose signature values are within $[I_{\min}, I_{\max}]$ and closer to v's than to any other labeled vertex. In effect, this removes a sub-interval around each $v \in Q$ from the larger similarity interval.

The perturbations o_v are stored in a separate data structure which we call the *exception map*. Once all perturbations are added, we are left with similarity thresholds (our interval $[I_{\min}, I_{\max}]$) on our shape signature that demarcate similar vertices; we use the interval to classify unlabeled vertices. We prevent contradictory situations by not allowing the user to label the same face or vertex as both similar and dissimilar.

undesirable. In this case, we set the default interval bounds to $[\frac{1}{2}(s_0 + d_0), \frac{1}{2}(s_0 + d_1)]$, where s_0 is the signature value of the vertex labeled similar at time t, and d_0, d_1 are the next smaller and larger signature values among vertices labeled dissimilar. (If no vertices are labelled dissimilar, the minimum and maximum vertex signature values of the entire mesh at time t are used.)

4.3. Impulse falloff

We have now constructed a similarity threshold that respects user guidance for a set of labeled vertices *at a particular scale value t*. In an interactive session, it is desirable to smooth the exception map in order to propagate the corrections made through the perturbations to other nearby scales in the signature, $t_f \in [t - \delta, t + \delta]$. For each o_v, the perturbation applied at scale parameter t to v and its nearby unlabeled vertices, we construct a quadratic falloff "bump" and apply the perturbation o_v to a scale neighborhood about t. The perturbation we ultimately apply is therefore, $b_v(p) = o_v f(p)$, where

$$f(p) = \begin{cases} 1 - (3p^2) & p < 1/3 \\ 1 - (-1.5p^2 + 3p - 0.5) & 1/3 \le p < 1 \\ 0 & \text{otherwise} \end{cases}, \qquad (5)$$

$p = \left|\frac{t_f - t}{t_N - t}\right|$, t_f is the scale value we're applying the fall-off on, and t_N is the closest scale value of a previously established similarity threshold on the mesh (or the scale upper/lower bound, whichever is closer). t_N prevents the perturbation for one set of labeled vertices from affecting a previously established self-similarity.

The practical impact of this smooth perturbation is that any subsequent set of labeled vertices will create a similarity threshold on a modified version of the underlying shape signature, incorporating relationships previously inferred from other sets of user-labeled relations. Our exception map stores the collection of quadratic bumps separately from the initial signature values and applies them on-the-fly. Figure 6 shows perturbations removing dissimilar vertices from the similarity threshold interval and their falloff impacting nearby signature values. Note that falloff contributions are order dependent. That is, there is no guarantee the similarity thresholds generated by solving for sets of labeled vertices A then B will produce the same result as solving for B then

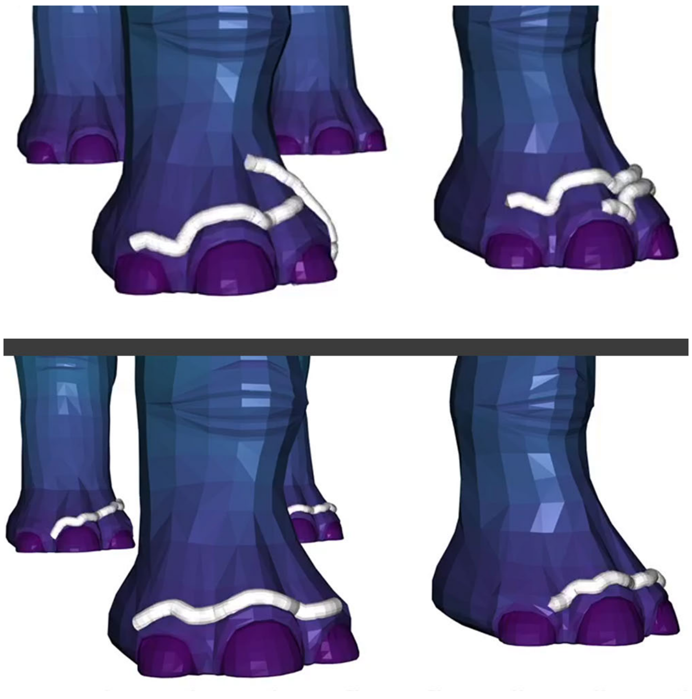

Figure 7: (top) The best curves from random seeding without gradient-based optimization; (bottom) The best curves with gradient-based optimization

A. This is because the impact of offset falloff is reduced with each set of labels, as the parameter search space is bisected by each solve. Functionally, this behaves as though the first labels having a higher weight then the second set of labels. If the user changes their mind, or the order of labelling, more interaction is required to come to the desired result. However, in our usage, this scenario generally only requires one more suggestion to correct.

5. Curve Propagation

Beyond point-based region similarity, many operations can be formulated in terms of curve feature propagation. For example, candidate locations for repeated geometry can be found by searching for a similar feature curve, or a stroke of 3D paint can be applied to many locations simultaneously. To support this, users can draw an open curve on the surface and see similar curves elsewhere. The user chooses how many similar curves to show.

5.1. Curve Definition

We represent our curves in tangent space on the surface via an "unrolling" routine which creates a distortion-free parameterization of only those faces populated by the curve. We use an arbitrary 2D coordinate frame in the plane of the triangle containing the first curve point as the tangent space. To initially obtain tangent-space coordinates for the entire curve, the points drawn in 2D are projected onto the mesh via ray casting to obtain barycentric coordinates. Unrolling proceeds from the first face containing the first curve point. If a subsequent curve point lies in the same triangle, its tangent-space coordinates are known. If a subsequent curve point lies in an adjacent triangle, the adjacent triangle is rotated about the shared edge into the tangent plane, extending it. The point's tangent-space coordinates can now be trivially computed. Due to a low input sampling rate (relative to triangle density) or concavities in the mesh, adjacent points in the curve may not lie inside the same or adjacent triangles. When such a gap is detected, we run a Dijkstra's shortest-path algorithm from the triangle on one side of the gap until the triangle on the other side is found. The shortest-path algorithm is run on the graph of medial triangles of the 3D mesh (the triangles formed by connecting the edge midpoints of each triangle). The sequence of faces selected by the shorted path algorithm are each rotated in turn about their shared edges to extend the tangent plane to the subsequent curve point. This process continues until the entire curve has been traversed. We refer to the resulting flattened sequence of mesh faces as the "curve cover" and store our curve's 2D tangent space coordinates and barycentric coordinates with respect to these faces. Note that the same mesh face may appear more than once in a curve cover.

5.1.1. Curve Similarity

Deciding if one curve is similar enough to another is a key part of our applications. We measure similarity between curves P and Q on the surface discretized with the same number of points as

$$E_{curve}(P, Q) = \sum_{i=1}^{n} (\varphi(P_i) - \varphi(Q_i))^2 \tag{6}$$

where φ is a function that takes a face and barycentric coordinate and returns a barycentrically interpolated shape signature value. If $E_{curve}(P, Q) < E_{curve}(P, O)$, then Q is more similar to P than O.

Consider the situation in which Q is a copy of P's tangent-space coordinates transplanted into another arbitrary tangent space on the surface. The core of our edit propagation routine relies on the ability to slide Q around on the surface in directions reduce $E(P, Q)$. We define a method for transforming our curves rigidly within tangent space while maintaining a valid curve cover. The transformation is simply a 2D rigid operation, applying a rotation of θ and an (x, y) translation to all the points in the given curve:

$$P' = R_\theta P + T_{(x,y)} \tag{7}$$

However, after transforming the curve points, some of the points will no longer be contained in the curve cover. In order to find a new cover, we first unroll faces along the tangent-space vector connecting the untransformed and transformed first curve point. Each time an edge is intersected, we unroll the adjacent triangle. The triangle containing the transformed first curve point defines our new tangent space. We obtain our new curve cover by similarly unrolling faces along the tangent-space vector connecting each curve point to the next.

5.2. Hybrid Global/Local Optimization

In order to suggest new curve locations and mesh edits, it's necessary to choose locations on the surface where we'll initially place copies, C_i, of the original user drawn curve, P. Following this initial (global) placement, we then perform gradient-based optimization. Ideally, this seed placement would place a seed in each energy basin. In practice, this is virtually impossible to guarantee.

5.2.1. Seed point search

For our global step, we sample a fixed number of randomly chosen faces as initial locations ($n = 500$ in all cases). Based on the user's desired number of curve suggestions k, we select the k lowest values of $E(P, C_i)$. Since the user is generally not interested in such a large number of curves, we select the k lowest values of $E(P, C_i)$ where k is the number of suggestions the user would like. While this step filters our suggestions within some neighborhood of a possible similarity match, it does not solve for orientation, nor can it make any sort of local minima guarantee about placement.

5.2.2. Gradient-based optimization

Our objective in finding best placements for suggested curves is reducing $E_{curve}(P, C_i)$. Since our only parameters are the (x, y, θ) 2D transformation we can apply to a curve, we can easily express the analytical derivative

$$\nabla_{x,y,\theta} E_{curve}(P, C_i') = \sum_i \nabla_{x,y,\theta}(\varphi(P_i) - \varphi(C_i'))^2$$

where $C_i' = R_\theta C_i + T_{(x,y)}$. This is because locally the signature values vary barycentrically inside the tangent-space triangle containing each curve point. We can express the barycentrically interpolated signature function $\varphi(C_i')$ in terms of tangent-space coordinates x, y as a linear equation $H(x, y) = Ax + By + D$ by solving

$$\begin{bmatrix} S(v_1, t) \\ S(v_2, t) \\ S(v_3, t) \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ D \end{bmatrix} \tag{8}$$

where x_i, y_i are the tangent-space vertices of the triangle containing the point under evaluation in C_i and $S(v_i, t)$ the corresponding signature value.

The 2D rigid transformation function is:

$$f(x, y, \theta) = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_i cos\theta - y_i sin\theta + x \\ x_i sin\theta + y_i cos\theta + y \end{bmatrix} \tag{9}$$

Looking at the Jacobians,

$$J_f = \begin{bmatrix} 1 & 0 & -x_i sin\theta - y_i cos\theta \\ 0 & 1 & x_i cos\theta - y_i sin\theta \end{bmatrix} \tag{10}$$

$$J_H = \begin{bmatrix} A & B \end{bmatrix} \tag{11}$$

$$J_H J_f = \begin{bmatrix} A & B & A(-x_i sin\theta - y_i cos\theta) + B(x_i cos\theta - y_i sin\theta) \end{bmatrix} \tag{12}$$

We optimize this using L-BFGS-B from CppOptimizationLibrary [37]. An optimization with line search is necessary because E_{curve} (Equation 6) is piecewise smooth but only C^0 across triangle boundaries.

5.3. Comparable Methods

Our definition of surface curves tackles a similar problem as the snakes-on-a-surface approach of Bischoff et al. [38] in that we also transform curves on a surface. However, Bischoff et al.'s approach has several limitations. First, Bischoff et al.'s curves cannot self-intersect, whereas our curves can self-intersect an arbitrary number of times. Second, the resolution of snakes is dependent on the mesh resolution, since snake vertices always lie on mesh edges, whereas our curves can have any number of vertices within the interior of a triangle. Third, our tangent-space curve definition allows us to arbitrarily transplant our curves to any location and orientation on the surface while preserving curve fidelity. Figure 8 shows a curve that Bischoff et al.'s approach could not handle.

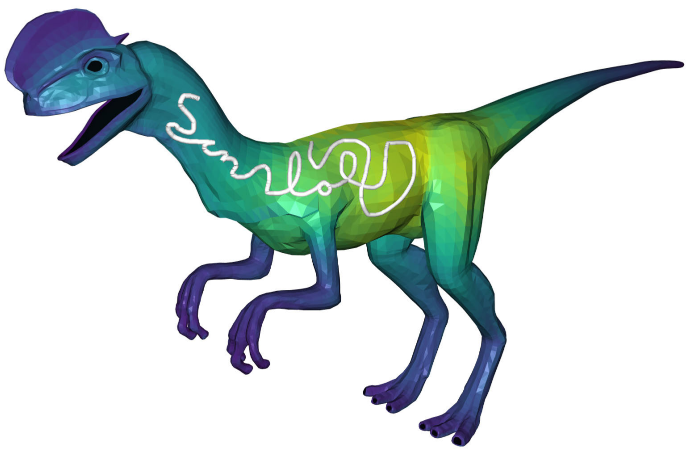

Figure 8: Our approach to curves allows us to effortlessly handle self-intersections, keeps curve resolution independent of mesh resolution, and transforms curves anywhere on the mesh without introducing intrinsic distortion.

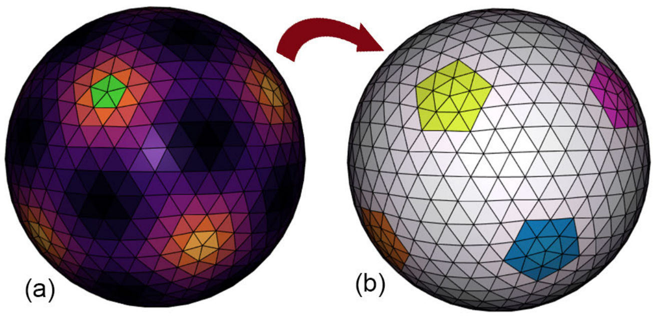

Figure 9: A subdivided icosahedron with (a) a single five face patch labeled as similar and (b) the subsequent segmentation given the sparse selection.

6. Applications

To demonstrate the utility of our interactive similarity thresholds, we modified a set of fundamental mesh operations to take as input our classification of patches on the mesh under thresholding. This reduces the workload on the user or extends the functionality of the originally proposed technique. The supplemental video demonstrates the behavior and performance of our curve propagation, as well as a comparison of HKS and SSDS signatures.

Implementation. We use libigl [35]'s implementation of core algorithms and routines, such as cotangent matrix calculations and the shape diameter function. Eigen's solver was used to find eigenvalues and eigenvectors for Heat Kernel Signature and Wave Kernel Signature construction.

6.1. Segmentation

We adopt the same mesh segmentation strategy as presented in Surfacing by Numbers [26], wherein the authors treat the mesh as a graph, establishing a collection of source and sink nodes based off of labeled user strokes across the surface, and solve for the min-cut. We use the same edge weights as Zelinka et al.:

$$e_{ij} = \eta \frac{2\pi - \theta_{ij}}{2\pi} + \gamma \frac{d_{ij}}{\bar{d}} \tag{13}$$

where e_{ij} is the edge weight between nodes i and j, θ_{ij} is the dihedral angle between the faces that share the edge (i, j), d_{ij} is the edge length between i and j, and $\bar{d}$ is the average edge length of all edges in the mesh. η and γ balance the contribution of dihedral angle versus edge length to the edge weight, but for our purposes we set $\eta = \gamma = 1$ unless otherwise noted. The min-cut of this graph is used as the boundary between two classes of segments. With regards to implementation, we solved for the min-cut using MAXFLOW [39]. Distinct segments were identified by a breadth-first search of the cut mesh, with each disconnected region identified as a segment.

Previously, the user would need to indicate, via drawing on the surface, a significant amount of the approximate extent of the shape segment they were interested in. Incorporating similarity thresholds into the interactive portion of the segmentation process, we replace the stroke drawing interface with sparse patch classification. Figure 9 shows that only a single patch of five faces is required to successfully segment all the mesh symmetries present in a subdivided icosahedron, whereas the original interactive step would require the user to mark every single symmetry manually. Figure 10 demonstrates the convenience of shape signature thresholds applied to a more interesting deer model. Note that this example uses HKS so an ambiguity exists when selecting the leg and deselecting the body due to similar curvature. This is also seen when selecting an antler and marking the head as dissimilar. While the legs and antlers are better matches under the HKS, the head and snout still display geometric properties similar to antlers and legs—notably curvature at the tips. One additional user edit would remove these matches in both cases. If we were to instead use SSDS, these matches would not occur, since the volume difference is significant.

6.2. *Geometry transfer*

We enable the propagation of 3D geometry edits made directly on the mesh. By employing the aforementioned curves as an editing tool, we can find appropriate locations for new geometry placement.

1. The application chooses an appropriate shape signature (or a vectorized amalgamation that can be queried for a scalar metric). For our purposes, we use the Smoothed Shape Diameter Signature (3.1).
2. The user draws the contour around the border, or along the central axis, of the geometry they wish to be copied on the surface.
3. The user requests some k suggestions for new placements.
4. New curve seeding and location refinement proceed according to Section 5.2, finding the k best curves to return.
5. The suggested placement of new curves will appear, and the user can accept or reject them. This labelling will classify the curve covers under the similar/dissimilar scheme described in Section 4, and new suggestions can be requested that will respect these new value assignments.
6. Accepted curves have the geometry assigned to them replicated relative to their placement.

An example of this process can be seen in Figure 2 in which a curve in the shape of a ring is drawn on finger, then the returned suggested curves are all accepted, and a ring pop is placed atop each curve's location, aligned with the curve direction.

In general, some matches may be quantifiably worse (by signature residuals) than others. In the future, we would like to explore automatically selecting the number of suggestions, such as by clustering based on result residuals and showing the best cluster.

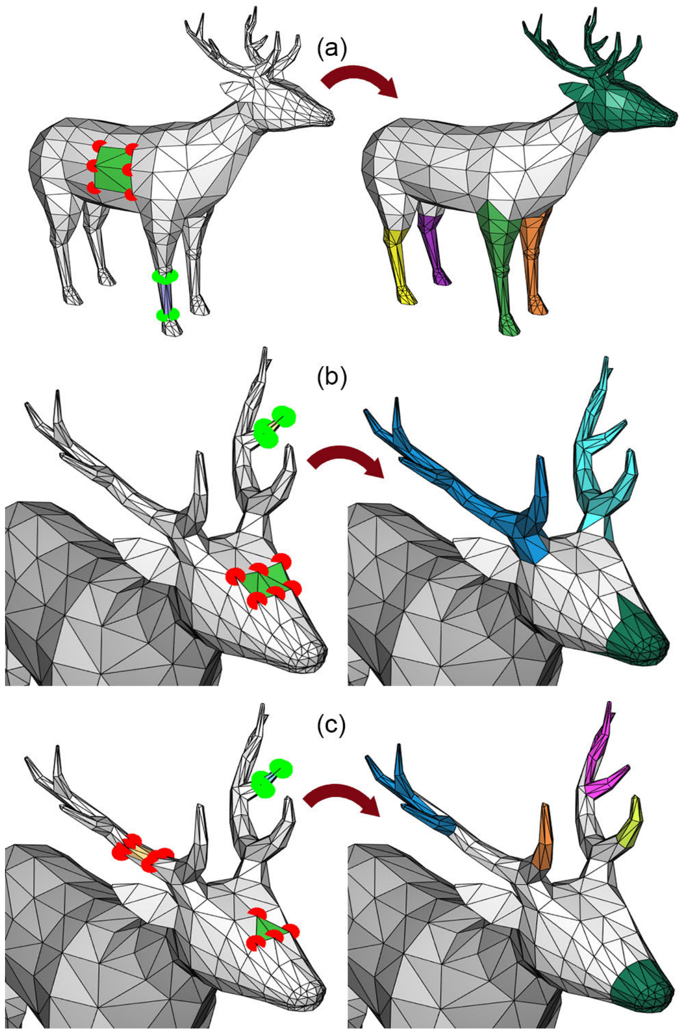

Figure 10: A deer segmented by our method using HKS given different sets of labels (green vertices are similar, red vertices are dissimilar). (a) Two small patches segment all the legs. (b) Extracting segments for the antlers. (c) A refinement of (b), excluding the main body of the antlers and just segmenting the tips.

6.3. *Pattern Duplication*

In addition to geometry propagation, we also apply our technique to pattern duplication. Using our implementation, the user draws a pattern on the mesh, arbitrarily or aesthetically placed, defining a curve whose surface-bound shape will remain undistorted in all of the suggestions that will later be presented. The workflow is essentially the same as 6.2, but the results are used for either merging the actual curve geometry with the surface, or using the surface-bound coordinates to access and modify mesh attributes.

Figure 11 shows this procedure illustrated on Homer with a single stripe running down the side of the stomach. Given this sparse information, the inferred suggestions show a series of stripes that equally space out on both sides of the stomach, with proper orientation and a visually pleasing tendency towards equal spacing. We commit these stripes to the mesh in a merge operation, creating new geometric features patterned across the surface from a single user effort.

It should be emphasized that the behavior of the suggestions is influenced by the shape signature being used. The same example

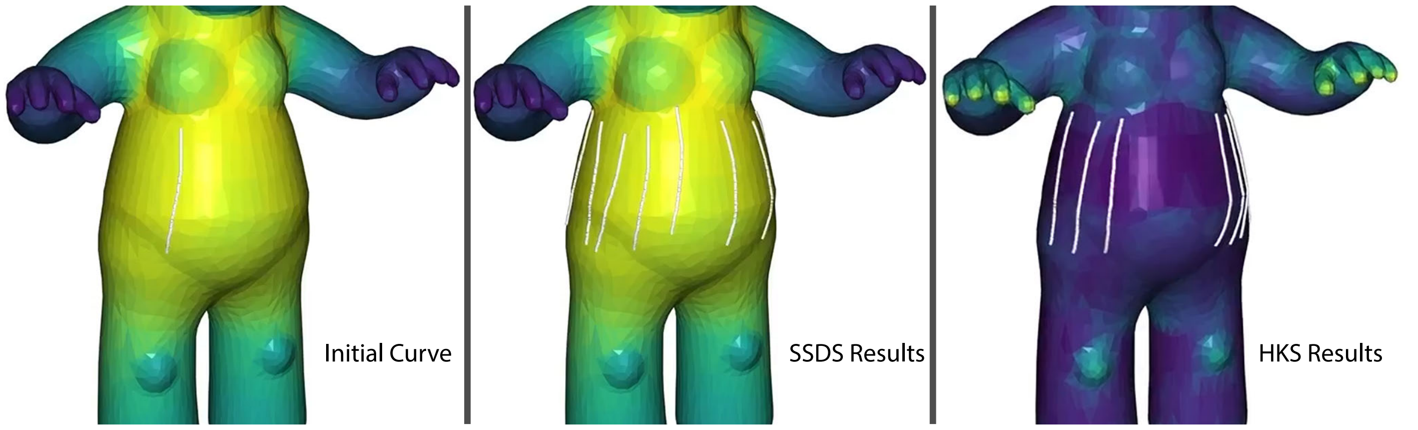

Figure 11: A line is drawn down Homer's belly (left). The curve propagation suggestions are shown for SDSS (middle) and HKS (right). The HKS suggestions more closely follow the curvature of the surface under the line (staying where there is an inflection around Homer's sides), whereas the SSDS suggestions correspond to consistent volume regardless of overall curvature.

of stripes on Homer with the Heat Kernel Signature, whose values follow curvature far more than the Smoothed Shape Diameter Signature, would return suggestions in regions that shared similar curvature instead of shape volume. Neither of these behaviors is objectively correct more correct than the other, but should be taken into account when choosing a shape signature as an initial expression of surface similarity.

6.4. Performance

All applications presented were computed in a short amount of time that exceeds what would be considered realtime, with most results produced within roughly 12 seconds on average, ranging from approximately 4 seconds to 27 seconds. These times reflect the codebase run on modest consumer laptop and desktop hardware. The timings do not include calculating the Smoothed Shape Diameter Signature, whose values are stored as part of a model pre-processing step for demonstration purposes.

7. Conclusion

We described a method for the interactive design of self-similarity maps through solving for shape signature scales, thresholds, and exceptions that best represent user guidance on the mesh in the form of sparse patch labeling. We also described an approach to propagate curve features throughout a mesh via global/local optimization. Our applications demonstrate the potential broad applicability of our method, though we anticipate other uses arising. Tracking a user interacting with a mesh via some creative process to determine implicit labels could be an easy way to integrate our method into existing tools. Likewise, the Smoothed Shape Diameter Function is generally useful and deserves broader adoption.

Limitations. One limitation of our method is that it does not handle contradictory labeling. A user cannot label the same vertex or patch as both similar and dissimilar. However, classification of a vertex or patch at scale parameter t can be different than that at a different scale parameter u for a different set of relations. We believe this is a reasonable restriction as the logical construction of the labeling makes both tags mutually exclusive from an interactive perspective anyway.

A second limitation is that while similarity thresholds handle approximate symmetries well, discerning a user pattern of selections on a mesh containing a large number of perfect symmetries can degenerate into the user manually excluding each symmetry they don't want. This is due to the fact that if two signature values for two different vertices are identical at every scale, and one is labeled similar, the other will also always be inferred to be similar unless manually labeled dissimilar.

Finally, our global/local curve propagation is seeded via uniform sampling of the mesh. While this did not present a problem in our examples, it could become problematic for more irregular shapes. It would be interesting to explore rough feature discovery driven by the signature to obtain a better sampling.

Future Work. In the future, we would like to explore the ramifications of updating signatures, and subsequently thresholds, across interactive geometry edits. This becomes particularly relevant if we're progressively building similarity mappings as the user work on and adds to a mesh. There are a number of challenges that would need to be addressed, not the least of which is quickly making small incremental approximate updates to the shape signature in real-time. This could be done based on visibility for the Smoothed Shape Diameter Signature. This would be more challenging for the Heat or Wave Kernel Signature without changing and decomposing the Laplace-Beltrami operator, which is restrictively expensive. It will also be necessary to ensure that similar changes propagated to regions that are all within a threshold do not shift any of the modified regions outside of the threshold, which would break similarity when it should intuitively be preserved. This might be approached by having a hybrid signature system that makes very coarse approximations in the short-term before mesh geometry changes are fully committed and a better signature update can be made without impact on the user experience. We would also like to investigate correspondence editing in the functional map setting [40], where our impulse falloffs may also be able to provide user control. Finally, we would like to integrate our approach into a popular mesh editor for further evaluation.

8. Acknowledgment

This work was funded in part by the US NSF (IIS-1453018), a Google research award, and a gift from Adobe Systems Inc.

References

[1] E. Guy, J.-M. Thiery, T. Boubekeur, SimSelect: Similarity-based selection for 3d surfaces, in: Computer Graphics Forum, Vol. 33, Wiley Online Library, 2014, pp. 165–173.

[2] O. Van Kaick, H. Zhang, G. Hamarneh, D. Cohen-Or, A survey on shape correspondence, in: Computer Graphics Forum, Vol. 30, Wiley Online Library, 2011, pp. 1681–1707.

[3] H. Zhang, O. Van Kaick, R. Dyer, Spectral mesh processing, in: Computer graphics forum, Vol. 29, Wiley Online Library, 2010, pp. 1865–1894.

[4] Yulan Guo, M. Bennamoun, F. Sohel, Min Lu, Jianwei Wan, 3D Object Recognition in Cluttered Scenes with Local Surface Features: A Survey, IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (11) (2014) 2270–2287. `doi:10.1109/TPAMI.2014.2316828`.
URL `http://ieeexplore.ieee.org/document/6787078/`

[5] L. Shapira, A. Shamir, D. Cohen-Or, Consistent mesh partitioning and skeletonisation using the shape diameter function, The Visual Computer 24 (4) (2008) 249.

[6] S. Zelinka, M. Garland, Similarity-based surface modelling using geodesic fans, in: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, ACM, 2004, pp. 204–213.

[7] M. Peng, J. Xing, L.-Y. Wei, Autocomplete 3d sculpting, ACM Transactions on Graphics (TOG) 37 (4) (2018) 132.

[8] N. J. Mitra, L. J. Guibas, M. Pauly, Symmetrization, ACM Transactions on Graphics (TOG) 26 (3) (2007) 63.

[9] R. M. Rustamov, Laplace-Beltrami eigenfunctions for deformation invariant shape representation, Computer Graphics Forum (2007) 9.

[10] J. Sun, M. Ovsjanikov, L. Guibas, A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion, Computer Graphics Forum 28 (5) (2009) 1383–1392. `doi:10.1111/j.1467-8659.2009.01515.x`.
URL `http://doi.wiley.com/10.1111/j.1467-8659.2009.01515.x`

[11] K. Gębal, J. A. Bærentzen, H. Aanæs, R. Larsen, Shape analysis using the auto diffusion function, Computer Graphics Forum 28 (5) (2009) 1405–1413.

[12] M. Ovsjanikov, Q. Mrigot, F. Mmoli, L. Guibas, One point isometric matching with the heat kernel, in: Computer Graphics Forum, Vol. 29, Wiley Online Library, 2010, pp. 1555–1564.

[13] T. K. Dey, K. Li, C. Luo, P. Ranjan, I. Safa, Y. Wang, Persistent heat signature for pose-oblivious matching of incomplete models, in: Computer Graphics Forum, Vol. 29, Wiley Online Library, 2010, pp. 1545–1554.

[14] M. Aubry, U. Schlickewei, D. Cremers, The wave kernel signature: A quantum mechanical approach to shape analysis, in: 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), IEEE, 2011, pp. 1626–1633. `doi:10.1109/ICCVW.2011.6130444`.
URL `http://ieeexplore.ieee.org/document/6130444/`

[15] R. Gal, D. Cohen-Or, Salient geometric features for partial shape matching and similarity, ACM Transactions on Graphics (TOG) 25 (1) (2006) 130–150.

[16] R. Liu, H. Zhang, Mesh Segmentation via Spectral Embedding and Contour Analysis, Computer Graphics Forum 26 (3) (2007) 385–394. `doi:10.1111/j.1467-8659.2007.01061.x`.
URL `http://doi.wiley.com/10.1111/j.1467-8659.2007.01061.x`

[17] L. Shapira, A. Shamir, D. Cohen-Or, Consistent mesh partitioning and skeletonisation using the shape diameter function, The Visual Computer 24 (4) (2008) 249–259. `doi:10.1007/s00371-007-0197-5`.
URL `http://link.springer.com/10.1007/s00371-007-0197-5`

[18] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, D. Dobkin, Modeling by example, in: ACM Transactions on Graphics (TOG), Vol. 23, ACM, 2004, pp. 652–663.

[19] X. Chen, A. Golovinskiy, T. Funkhouser, A benchmark for 3d mesh segmentation, ACM Transactions on Graphics 28 (3) (2009) 1. `doi:10.1145/1531326.1531379`.
URL `http://portal.acm.org/citation.cfm?doid=1531326.1531379`

[20] E. Kalogerakis, A. Hertzmann, K. Singh, Learning 3D Mesh Segmentation and Labeling, ACM Transactions on Graphics 29 (3).

[21] Y.-K. Lai, S.-M. Hu, R. R. Martin, P. L. Rosin, Fast mesh segmentation using random walks, in: SPM '08: Proceedings of the 2008 ACM symposium on Solid and physical modeling, 2008, p. 9.

[22] A. Shamir, A survey on Mesh Segmentation Techniques, Computer Graphics Forum 27 (6) (2008) 1539–1556. `doi:10.1111/j.1467-8659.2007.01103.x`.
URL `http://doi.wiley.com/10.1111/j.1467-8659.2007.01103.x`

[23] R. Liu, H. Zhang, Segmentation of 3d meshes through spectral clustering, in: Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on, IEEE, 2004, pp. 298–305.

[24] Y. Lee, S. Lee, A. Shamir, D. Cohen-Or, H.-P. Seidel, Intelligent mesh scissoring using 3d snakes, in: Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on, IEEE, 2004, pp. 279–287.

[25] M. Zöckler, D. Stalling, H.-C. Hege, Fast and Intuitive Generation of Geometric Shape Transitions.pdf (Sep. 1999).

[26] S. Zelinka, M. Garland, Surfacing by numbers, in: Proceedings of Graphics Interface 2006, Canadian Information Processing Society, 2006, pp. 107–113.

[27] R. Schmidt, C. Grimm, B. Wyvill, Interactive Decal Compositing with Discrete Exponential Maps.pdf (2006).

[28] R. Schmidt, Stroke parameterization, in: Computer Graphics Forum, Vol. 32, Wiley Online Library, 2013, pp. 255–263.

[29] O. Sorkine, D. Cohen-Or, R. Goldenthal, D. Lischinski, Bounded-distortion piecewise mesh parameterization, in: Proceedings of the Conference on Visualization '02, VIS '02, IEEE Computer Society, Washington, DC, USA, 2002, pp. 355–362.
URL `http://dl.acm.org/citation.cfm?id=602099.602154`

[30] Y. Lee, S. Lee, Geometric snakes for triangular meshes, Computer Graphics Forum 21 (3) (2002-09) 229,238.

[31] M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, L. J. Guibas, Discovering structural regularity in 3d geometry, ACM transactions on graphics (TOG) 27 (3) (2008) 43.

[32] J. Digne, S. Valette, R. Chaine, Sparse geometric representation through local shape probing, IEEE transactions on visualization and computer graphics 24 (7) (2017) 2238–2250.

[33] A. Hamdi-Cherif, J. Digne, R. Chaine, Super-resolution of point set surfaces using local similarities, Computer Graphics Forum 37 (1) (2018) 60–70.

[34] O. Stein, E. Grinspun, M. Wardetzky, A. Jacobson, Natural boundary conditions for smoothing in geometry processing, ACM Trans. Graph. 37 (2) (2018) 23:1–23:13. `doi:10.1145/3186564`.
URL `http://doi.acm.org/10.1145/3186564`

[35] A. Jacobson, D. Panozzo, et al., libigl: A simple C++ geometry processing library, http://libigl.github.io/libigl/ (2017).

[36] R. H. Byrd, P. Lu, J. Nocedal, A Limited-Memory Algorithm for Bound-Constrained Optimization.pdf (Feb. 1996).

[37] P. Wieschollek, Cppoptimizationlibrary, `https://github.com/PatWie/CppNumericalSolvers` (2016).

[38] L. K. Stephan Bischoff, Tobias Weyand, Snakes on triangle meshes, Bildverarbeitung für die Medizin.

[39] Y. Boykov, V. Kolmogorov, An Experimental Comparison of Min-Cut Max-Flow Algorithms for Energy Minimization in Vision (Sep. 2004).
URL `http://dx.doi.org/10.1109/TPAMI.2004.60`

[40] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, L. Guibas, Functional maps: a flexible representation of maps between shapes, ACM Transactions on Graphics (TOG) 31 (4) (2012) 30.