Eurographics Symposium on Geometry Processing 2010
Olga Sorkine and Bruno Lévy
(Guest Editors)

Volume 29 (2010), Number 5

Persistent Heat Signature for Pose-oblivious Matching of Incomplete Models

T. K. Dey K. Li C. Luo P. Ranjan I. Safa Y. Wang

The Ohio State University, Columbus OH, USA.

Abstract

Although understanding of shape features in the context of shape matching and retrieval has made considerable progress in recent years, the case for partial and incomplete models in presence of pose variations still begs a robust and efficient solution. A signature that encodes features at multi-scales in a pose invariant manner is more appropriate for this case. The Heat Kernel Signature function from spectral theory exhibits this multi-scale property. We show how this concept can be merged with the persistent homology to design a novel efficient pose-oblivious matching algorithm for all models, be they partial, incomplete, or complete. We make the algorithm scalable so that it can handle large data sets. Several test results show the robustness of our approach.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations I.3.8 [Computer Graphics]: Applications—

1. Introduction

The need for effective and efficient shape retrieval algorithms is ubiquitous in a broad range of applications in science and engineering. With the increasing understanding of shape geometry and topology in the context of shape similarity, workable solutions for shape retrieval are being produced. Numerous work drawing upon insightful ideas such as [CTSO03, FS06, GCO06, JZ07, OFCD01, OBBG09] have made this possible. See also a comprehensive comparison study [SMKF04] and a survey [TV08].

The problem of shape matching and retrieval is not trivial even if only rigid body transformations are allowed to vary the shapes. Further difficulty ensues if shape variations include small deformations. Even more realistic assumption should allow the shapes to vary in pose meaning that the intrinsic metric is not distorted much though the three dimensional embedding varies widely. Spectral methods have shown remarkable resilience to shape variations caused by the extrinsic metric while remaining stable under small changes in the intrinsic metric [L0́6, Rus07, RWP06]. As a result, researchers have started to study this approach more in depth [JZ07, OBBG09, Reu10, SOG09]. In this paper we explore this approach to address one of the main difficulties still existing for shape retrieval systems– *pose-oblivious* matching of *partial* and *incomplete* 3D models.

The matching of a partial or an incomplete model against a complete one cannot rely on features that are too global. At the same time, any matching relying on only local measures becomes susceptible to noise caused by small perturbations. Therefore, we need something in between which can describe shape features at different scales. The Heat Kernel Signature (*HKS*) recently introduced in [SOG09] bears this multi-scale property. However, it has not yet been demonstrated how this signature can be used effectively for partial or incomplete shape retrievals from a database that may itself contain partial or incomplete models. We provide such a scheme which is both robust and scalable for large data sets.

Our method is based on a novel synergy between *HKS* and persistent homology. Given a surface M with an initial distribution of unit heat concentrated at any point $x \in M$, the *HKS* at x at time (i.e, duration of the flow) t provides the amount of heat retained at x after heat dissipates for time t according to the well known heat equation. This heat dissipation is determined by the intrinsic geometry of M and the

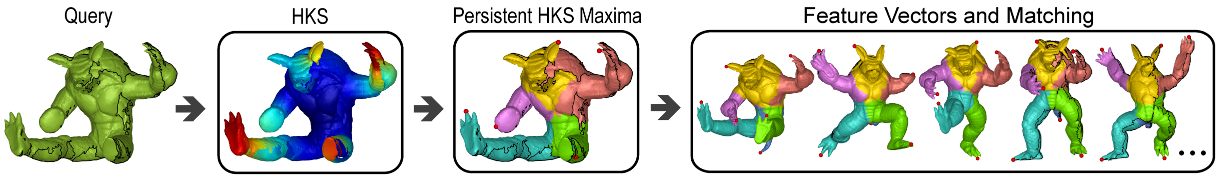

Figure 1: *Given a query Armadillo model that is pose-altered, incomplete, and partially scanned, our method first computes the heat kernel signature function at a certain scale, and then extract a set of HKS maxima (red dots) using persistent homology. Feature vectors computed at these maxima are then used to search for the most similar models, be it complete, partial, or incomplete, in a shape database. A few top matches are shown. The black curves are the boundary curves of either partial or incomplete models. Correspondence between segmentations of different models is shown with consistent coloring.*

influence of shape features on the heat flow can be controlled by regulating the time. Small time scales dissipate heat over a small region and thereby allow local features to regulate the heat whereas large time scales allow global features to exert more influence. This suggests that one can use *HKS* at different time values to describe features at multiple scales. Indeed, Ovsjanikov el al. use this as a shape descriptor for every vertex of an input surface, and quantize the space of shape descriptors by k-means clustering to obtain a succinct set of representative features [OBBG09]. They represent an input shape based on the distribution of its shape descriptors with respect to these representative features, and develop an efficient shape retrieval system.

The distribution-based features tend to be less discriminating for partial matching. The *HKS* function is more likely to remain similar on a surface and its partial counterpart when t is relatively small. However, a small diffusion interval tends to increase the local variation in the *HKS* function values, and makes it more sensitive to noise. To counter this, we bring in the tool of persistent homology [ELZ02] to identify important features.

We argue that the critical points, in particular, the maxima of *HKS*, serve as good candidates for feature points. We consider only *persistent features*, which are *HKS* maxima that persist beyond a given threshold. To compute such maxima, instead of using the standard persistent algorithm, we employ a region merging algorithm that bypasses detecting persistence values of all critical points and focuses only on eliminating those maxima that are not persistent. As a byproduct we also obtain a segmentation of the surface that appears to be robust, and consistent between a shape and its partial versions even if they are in different poses (see the last box in Figure 1). This may be of independent interest.

The persistent feature points, together with a multi-scale *HKS* description at each points, provide a concise yet discriminating shape descriptor for the input surface, which is also robust under near-isometric deformations and partial occlusions. Indeed, experimental results show that our algorithm is able to match partial, incomplete, and pose-altered query shapes in a database of moderate size efficiently and with a high success rate. Finally, for large input data, we also develop an efficient algorithm to approximate *HKS* values to make our algorithm scalable.

We remark that using point signatures for matching is not new, see e.g. [CJ96, GMGP05, SF06]. However, our method relies on a novel synergy between the multi-scale *HKS* and the topological persistence algorithm. Our algorithm is the first one designed specifically for matching partial, incomplete, as well as pose-modified shapes in a database. Experimental results show that our method is robust, efficient, and quite effective in partial shape retrieval.

2. Heat Kernel Signature

Our matching algorithm relies on the *Heat Kernel Signature* (*HKS*) proposed and analyzed in [SOG09]. This shape descriptor is derived from the heat operator which we briefly describe first.

2.1. Heat operator

Given a Riemannian manifold M, the *heat operator* $\mathcal{H}_t$ w.r.t. a parameter $t \in \mathbb{R}$ is an operator on $L^2(M)$, the space of square integrable functions on M. Specifically, imagine that there is an initial heat distribution on M at time 0. Now the heat starts to diffuse and this diffusion process is governed by the following *heat equation*, where $u(x,t)$ denotes the amount of heat at a point $x \in M$ at time t, and $\triangle$ is the Laplace-Beltrami operator of M: $\triangle u(x,t) = -\frac{\partial u(x,t)}{\partial t}$. Given a function $f : M \to \mathbb{R}$, the heat operator applied to f gives the heat distribution at time t with f being the initial heat distribution. That is, $\mathcal{H}_t f = u(\cdot,t)$ if $u(\cdot,0) = f$. For a square integrable function f, a unique solution to the heat equation exists, and $\mathcal{H}_t f$ has the form: $\mathcal{H}_t f(x) = \int_M \mathbf{h_t}(x,y) f(y) d\mu_y$, where $d\mu_y$ is the volume form at y, and $\mathbf{h_t} : M \times M \to \mathbb{R}$ is the so-called *heat kernel* function.

Heat kernel. Intuitively, for two points $x,y \in M$, $\mathbf{h_t}(x,y)$ measures the amount of heat that passes from y to x within time t out of unit heat at y. If M is the Euclidean space $\mathbb{R}^d$, then the corresponding heat kernel is: $\mathbf{h_t}(x,y) =$

$\frac{1}{(4\pi t)^{d/2}} e^{-||x-y||^2/4t}$. For a general manifold M, however, there is no known explicit expression for the heat kernel. There is fortunately an alternative way to represent the heat kernel. More specifically, the heat operator is compact, self-adjoint, and positive semi-definite. Thus it has discrete spectrum $1 = \rho_0 \geq \rho_1 \geq \ldots \geq 0$ with $\mathcal{H}_t \phi_i = \rho_i \phi_i$. By the Spectral Theorem, the heat kernel can be written as:

$$\mathbf{h_t}(x,y) = \sum_{i \geq 0} \rho_i \phi_i(x) \phi_i(y). \qquad (1)$$

Laplace-Beltrami operator. Eqn (1) implies that, if we know the spectrum and eigenfunctions of the heat operator, we can then compute the heat kernel function. Our algorithm computes the spectrum of the heat operator via the Laplace-Beltrami operator $\triangle$ of M, which is related to the heat operator as $\mathcal{H}_t = e^{-t\triangle}$. This means that $\mathcal{H}_t$ and $\triangle$ share the same eigenfunctions ϕ_i, and their eigenvalues satisfy $\rho_i = e^{-t\lambda_i}$, where $\triangle \phi_i = \lambda_i \phi_i$. In other words, we can compute the heat kernel by $\mathbf{h_t}(x,y) = \sum_{i \geq 0} e^{-t\lambda_i} \phi_i(x) \phi_i(y)$.

2.2. *HKS*

The heat kernel function has many nice properties; see [SOG09] for a detailed discussion. The family of heat kernel functions uniquely defines the underlying manifold up to an isometry; thus it is very informative and a potentially good tool to construct a shape descriptor.

Specifically, Sun et al. propose the following Heat Kernel Signature (*HKS*) as a shape descriptor for a manifold M [SOG09].

$$HKS_t(x) = \mathbf{h_t}(x,x) = \sum_{i \geq 0} e^{-t\lambda_i} \phi_i(x)^2. \qquad (2)$$

Intuitively, it measures how much heat is left at time t for the point $x \in M$ if unit amount of heat is placed at point x when $t = 0$. *HKS* inherits many nice properties of the heat kernel. It is invariant to isometric deformations, and not sensitive to noise or even slight topological changes. It is multi-scale: as t gets larger, the eigenfunctions corresponding to large eigenvalues play a smaller role, hence only the main features of the shape detected by small eigenvalues matter. Most importantly, *HKS* is almost as informative as the family of functions $\mathbf{h_t}(.,.)_{t>0}$ (see Theorem 1 in [SOG09]). At the same time, it reduces the family of two-variables kernel functions to a family of single-variable functions, and is hence more succinct and much easier computationally.

***HKS* Maxima.** It is well-known that as $t \to 0$, there is an asymptotic expansion of the *HKS* function at every point $x \in M$ of the form:

$$HKS_t(x) = \mathbf{h_t}(x,x) = (4\pi t)^{-d/2} \sum_{i \geq 0} a_i t^i,$$

where $a_0 = 1$ and $a_1 = \frac{1}{6} s(x)$ with $s(x)$ being the scalar curvature at point x. For a 2-manifold M, $s(x)$ is simply the Gaussian curvature at x. Thus intuitively, the heat diffusion for small t is governed by intrinsic curvature. Heat tends to diffuse slower at points with positive curvature and faster with negative curvature. This suggests that critical points of *HKS* correspond to features of the shape, where maxima of *HKS* capture the tips of protrusions or the bottoms of concave areas. Hence, we propose to use *HKS* maxima as feature points for shape matching. It turns out that we can select only a handful of *persistent* feature points for matching.

2.3. Discrete setting

In the discrete setting, the input is a triangular mesh K whose underlying space is homeomorphic to a smooth surface M. To compute the *HKS*, one needs to compute the eigenvectors and eigenvalues of the Laplace-Beltrami (thus heat) operator. Several discrete Laplace operators for meshes have been developed in the literature (see surveys in [LÓ6, ZvKD10]). The most popular ones are the cotangent-scheme [Dod78, DMSB99, PP93, War06] and the finite element method based approach [RWP06]. We use the mesh-Laplacian proposed in [BSW08] for two reasons. First, the construction of this mesh-Laplacian is based on the heat diffusion idea, making it consistent and perhaps natural for computing Heat Kernel Signatures. Second, the mesh-Laplacian is the only current discrete Laplace operator so that both the operator itself and its spectrum converge to those of the manifold Laplacian with increasing mesh density [BSW08, DRW10] while requiring no constraints on triangle qualities.

3. Persistent Heat Maxima

Our goal is to compute a signature of a shape from its *HKS* functions. To make the size of the signature concise, we want to select a subset of vertices of the mesh that may find correspondences in *HKS* values in a shape which we want it to match. We may take the critical points, in particular, the maxima of the *HKS* as this subset. This subset itself could possibly be large. Furthermore, discretization errors and small variations in the shapes may cause many maxima to loose their correspondences in a similar shape. Therefore, we need a mechanism to select a small subset of these maxima that hopefully remain stable under small perturbations but allow pose variations.

One might think that the *HKS* at a large time scale t provides a few such stable maxima since such a *HKS* function describes the input shape at a large scale. However, the criticality of a point p is decided not only by the function value at p, but also by its relative value compared to its neighbors. Some unimportant local maxima at small t may survive for large t thereby invalidating t alone as a discriminating factor in filtering maxima. Furthermore, computing *HKS* at a large t value makes it more sensitive to partial and incomplete shapes. Hence we take the strategy that, first, we compute the *HKS* function for a moderately small t so that partial similarity tends to be well preserved. We next use the concept of persistent homology [ELZ02] to select a subset of the maxima that have large persistence. Experimental results suggest that this strategy produces robust feature points across partial and incomplete shapes.

3.1. Persistence

We now briefly review the concept of persistent homology [ELZ02]. Let K be the triangulation whose underlying space, denoted $|K|$, is a surface possibly with boundaries. Suppose we have a *HKS* function $h\colon V \to \mathbb{R}$ defined on vertices V of K. One can linearly interpolate h over all edges and triangles to obtain a piecewise linear function, still denoted by $h\colon |K| \to \mathbb{R}$.

The critical points of h occur at vertices of K where the homology of the *sub-level sets* $\{x \in |K| \text{ s.t. } h(x) < \alpha\}$ changes. Using the concept of persistent homology, one can define the persistence of these critical vertices. Informally speaking, while sweeping $|K|$ from $\min h$ to $\max h$, we inspect the topology change of the sub-level set at critical values which are values taken by critical vertices: either new topology is generated or some topology is destroyed, where topology is quantified by a class of 'cycles'. A critical vertex is a *creator* if new topology appears and a *destroyer* otherwise. It turns out that every destroyer v_2 is uniquely *paired* with a creator v_1 in the sense that v_2 destroys a topology created at v_1. The *persistence of* v_1 *and* v_2 is defined as $\mathrm{Pers}_h(v_1) = \mathrm{Pers}_h(v_2) = h(v_2) - h(v_1)$, which indicates how long a class of cycles created at v_1 lasts before it gets destroyed at v_2. A creator is either paired with a unique destroyer, or unpaired, in which case it has infinite persistence.

For a surface, there are three types of critical points, *maxima*, *minima*, and *saddles*. We are interested in computing the maxima with large persistence, which roughly correspond to feature points that are hard to remove by perturbing the *HKS* function values. The persistence pairings and values can be computed by the standard persistent algorithm on a filtration of K dictated by h. Specifically, modify h as:

$$\bar{h}\colon K \to \mathbb{R},\ \sigma \mapsto w \text{ where } w \text{ is maximum } h \text{ over vertices of } \sigma.$$

Sort the simplices in K with increasing order of $\bar{h}$ values . If two simplices have the same value but different dimensions, break the tie by putting the lower dimensional one first. Otherwise, break the tie arbitrarily. Note, this enforces any face of a simplex to appear before this simplex in the sorted order. Hence this sorted order $\sigma_1, \sigma_1, \ldots, \sigma_n$ defines a filtration of K: $\emptyset = K_0 \subset K_1 \subset \ldots \subset K_n = K$, where each K_i is a subcomplex of K and $K_{i+1} \setminus K_i = \sigma_i$.

The persistence algorithm when run on this filtration pairs up triangles with edges and edges with vertices. For paired simplices σ_1, σ_2 with $\bar{h}(\sigma_1) \le \bar{h}(\sigma_2)$, we define their persistence as $\mathrm{Pers}_{\bar{h}}(\sigma_1) = \mathrm{Pers}_{\bar{h}}(\sigma_2) = \bar{h}(\sigma_2) - \bar{h}(\sigma_1)$. It is known that Pers_h and $\mathrm{Pers}_{\bar{h}}$ have the following relation for a piecewise-linear function h. This justifies the discretized framework .

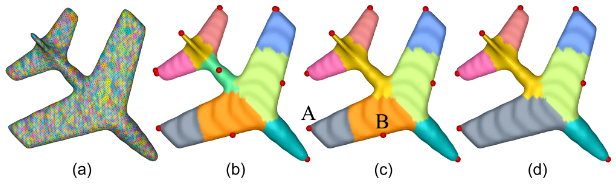

Figure 3: *Persistence based merging on an Airplane model. (a) Initially, every triangle represents a region; (b) shows the segmentation after merging all zero-persistence regions. The central triangle of every remaining region corresponds to a maximum of input function h. (c) A and B are two maxima, for the gray and orange regions, respectively, with A having a larger h value; (d) after the merging, A stays the maximum for the new region (gray colored).*

Proposition 1 ([CSEH09]) *Assume that σ_1 is paired with σ_2, and let v_1 and v_2 be the vertices of σ_1 and σ_2 respectively with the highest h value. Either $v_1 = v_2$ in which case it is a regular (non-critical) vertex. Otherwise, v_1 is paired with v_2 for h and $\mathrm{Pers}_h(v_1) = \mathrm{Pers}_h(v_2) = \mathrm{Pers}_{\bar{h}}(\sigma_1) = \mathrm{Pers}_{\bar{h}}(\sigma_2)$.*

By the above result, if a maximum v of h is paired with a saddle point u, then a triangle incident to v gets paired with an edge incident to u in the filtration induced by $\bar{h}$. The persistence value for v can also be computed by persistence value of that triangle. Hence it is safe to work with the filtration induced by $\bar{h}$ to compute persistence of critical points of h.

3.2. Region merging

It turns out that for a piecewise-linear function defined on a triangulation of a surface, one can compute the persistence pairing between maxima and saddles (and between minima and saddles) by sweeping the function value from large to small and maintaining the connected components throughout the course via a union-find data structure [ELZ02]. The process takes $O(n \log n)$ time (or $O(n\alpha(n))$ time if vertices are sorted), where n is the number of vertices and $\alpha(n)$ is the inverse Ackermann's function, a function that grows extremely slowly. We are interested only in important *HKS* maxima, that is, those corresponding triangles whose persistence values are more than a prescribed threshold. Instead of computing the complete list of persistence pairings, we employ a *region merging* algorithm that computes these pairings up to a level dictated by an input parameter λ.

Specifically, this algorithm cancels maxima-saddle or equivalently triangle-edge pairs till only those maxima whose persistence values are more than a threshold are left. As a by product, the merging algorithm also produces an explicit segmentation of the surface which seems to be stable w.r.t. partial and incomplete input and which could be of independent interest (see Figure 2). The induced segmentation is

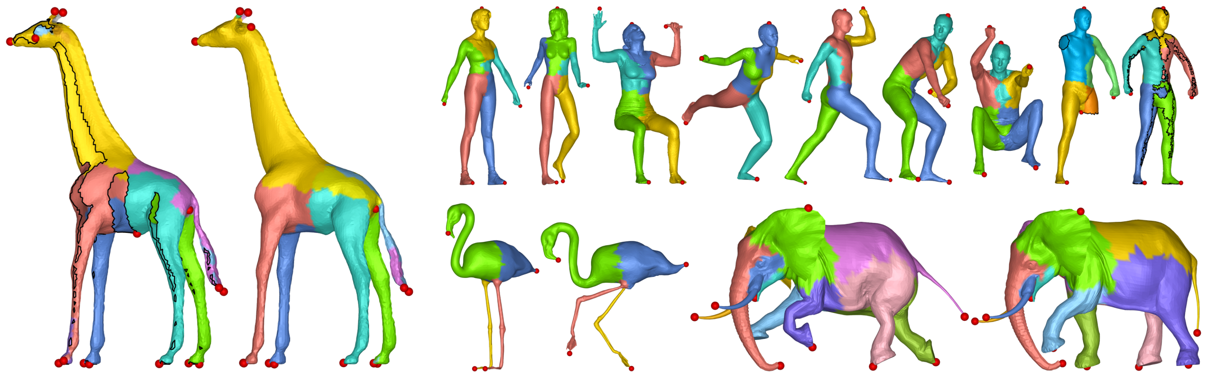

Figure 2: *Consistent identification of the persistent HKS maxima for different human / animal models in different poses. The two human models on the right are incomplete and partially scanned models with black curves being boundary curves.*

also later used to help us to approximate the *HKS* maxima for large data sets in Section 4.1.

For simplicity, we first describe the merging algorithm for a surface without boundary. The algorithm partitions the entire surface $M = |K|$ into regions which are grown iteratively. Each region maintains a *central triangle* and its *pair* edge which together provide a persistence value. At any stage, the central triangle t of R is the triangle with the highest function value in R, and its pair edge $\mathrm{p}r(t)$ is the edge with the highest function value among all edges in the boundary ∂R of R. Equivalently, the pair edge $\mathrm{p}r(t)$ is also the edge among all edges in ∂R such that the difference between $\bar{h}(t)$ and $\bar{h}(pr(t))$ is minimized. The current *persistence* of region R is defined as this difference: $\mathrm{p}(R) = \bar{h}(t) - \bar{h}(pr(t))$.

Initially, each region R consists of a single triangle $t \in K$, The central triangle of R is t itself whose pair-edge $\mathrm{p}r(t)$ is: $\mathrm{p}r(t) = argmax_{e \in \partial t}(\bar{h}(e))$. At any generic step, let R be the region whose persistence value $\mathrm{p}(R)$ is minimum among all regions — if regions have same persistence values, we break the tie by considering the one whose central triangle appears earlier in the filtration by $\bar{h}$. Let t be the central triangle in R paired with the edge $e \in \partial R$. Let R' be the *unique* region adjacent to e other than R. Let t' be the central triangle of R' and e' its pair edge. We merge R and R' to region $R \cup R'$. The central triangle t' of R' becomes the central triangle of this merged region whose pair edge is updated with the edge e'' in $\partial(R \cup R')$ that has the largest $\bar{h}$ value. Note that it is necessary that $\bar{h}(t') \geq \bar{h}(t)$ and $\bar{h}(e'') \leq \max\{\bar{h}(e), \bar{h}(e')\}$ as R is the region with the smallest persistence chosen for merging. Hence the persistence $\mathrm{p}(R \cup R')$ for the merged region becomes larger than or equal to both $\mathrm{p}(R)$ and $\mathrm{p}(R')$. If $\partial(R \cup R')$ is empty, then the central triangle is unpaired. We continue this process till the minimal persistence value of any current region exceeds an input threshold λ. The following result explains the utility of the merging algorithm. The proof is in Appendix A.

Proposition 2 *At any stage of the merging algorithm, if R is the region with the minimum persistence value $\mathrm{p}(R)$, then $\mathrm{Pers}_{\bar{h}}(t) = \mathrm{p}(R)$ where t is the central triangle in R, and the edge $\mathrm{p}r(t)$ pairs with t w.r.t the function $\bar{h}$.*

In particular, the set of central triangles t of regions that survive after merging with threshold $\lambda > 0$ are exactly the set of triangles with persistence $\mathrm{Pers}_{\bar{h}}(t) \geq \lambda$.

In other words, the above result implies that our algorithm merges regions with respect to the persistence order induced by the function $\bar{h}$, which by Proposition 1, is then connected to the persistence order of the function h. Note that at the beginning, all the regions we merge have zero-persistence value. These do not correspond to pairings between critical points of h (the case where $v_1 = v_2$ in Proposition 1). An example of the merging process is shown in Figure 3.

This merging process takes $O(n \log n)$ time in the worst case if we use the union-find data structure to maintain intermediate regions, and a priority-queue to maintain the order of regions to be processed. However, it terminates faster for lower threshold λ, and it also produces a segmentation of the input domain, which is in some sense a combinatorial version of the stable manifold decomposition induced by the input function. We remark that the merging idea has been used before to produce segmentations [CGOS09]. However, we show that our merging process and the resulting segmentation respect the topological persistence pairing.

Although the above procedure assumes no boundary in the input surface, it can be easily adapted to handle boundaries. Let E' be the set of boundary edges in a surface M. The algorithm simply ignores edges in E' when computing the boundary of each region. One can show that this is equivalent to first sealing all the boundaries in M to convert M to a manifold M' without boundary, and then performing the persistence algorithm on M'. A boundary loop C in ∂M is sealed (virtually) by adding a dummy vertex v with function value lower than all vertices in M, and then connecting all edges in C to v.

4. Matching

Feature vector. We now identify a small set of persistent heat maxima as *feature points* to concisely represent an input shape. Specifically, we use a fixed number κ, and perform the region-merging process till the remaining number of regions left equals κ. Experimentally we observe that $\kappa = 15$ maxima are usually sufficient to capture most prominent features, which we fix from now on. See the two Giraffe models in Figure 2 for an example, where similar feature points are captured for both the partial scan (left picture) and the complete version (right picture).

Next, we construct a feature vector for each one of the $\kappa = 15$ selected feature point. Since *HKS* for different time values describe local geometry at different scales, we compute *HKS* values for different t at each of the feature points — such a multi-scale description of local shape helps to enhance the discriminability of our feature vectors. We choose 15 different time scales to compute a 15D feature vector for each feature point. The time scales are chosen relative to a constant τ which is the parameter used for computing the discrete Laplace operator. The algorithm in [BSW08] approximates the discrete Laplacian based on a heat-dissipation process whose duration is determined by τ. Hence we take τ as the *time unit*, as it is not meaningful to use the eigenvectors of a *HKS* constructed at a resolution smaller than τ. In our experiments we fix τ to be 0.0002 and we consider $t = \alpha * \tau$ where α varies over 5, 20, 40, 60, 100, 150, 200, 300, 400, 500, 600, 700, 800, 900, 1000.

Scoring. Let F_1 and F_2 be the set of feature vectors computed for two surfaces M_1 and M_2, respectively. Each F_i is a collection of 15 feature vectors, where each vector is of dimension 15. For two vectors $f_1 \in F_1$ and $f_2 \in F_2$, we use the L_1-norm $||f_1 - f_2||_1$ to measure their distance. The matching score between M_1 and M_2 is computed as

$$\sum_{f_1 \in F_1} min_{f_2 \in F_2} ||f_1 - f_2||_1 + \sum_{f_2 \in F_2} min_{f_1 \in F_1} ||f_1 - f_2||_1 .$$

Note that this scoring function does not satisfy the triangle inequality, and thus is not a metric. One could use more sophisticated distance functions for scoring. We resort to this simple scoring function since it already provides good results. Once F_1 and F_2 are given with a preprocessing, computation of the scores becomes very efficient for such a small set of vectors.

4.1. Handling large meshes

One issue common to spectral methods is that computing eigenvectors for large matrices is expensive, even though the discrete Laplace matrix is sparse and we typically need only the first 300 eigenvectors. To make our algorithm scalable, one may use one of the following three approaches.

First, one can use the Shift-Invert spectral transform approach as proposed in [VL08], which can compute the first few eigenvectors accurately much more efficiently than the standard eigen-decomposition approaches. However, the time required is still large (about 10 hours for computing the first 1300 eigenvectors for a mesh with one million vertices).

We can trade-off the accuracy of eigenvectors for a more efficient algorithm. A natural strategy is to first sub-sample the input mesh and then compute *HKS* for the simplified structure. The sub-sampling can be achieved by any mesh simplification method, such as the QSlim mesh decimation software [Gar] which is used in [JZ07] to reduce the size of the input mesh. We call this approach *QSlim-simp*.

We propose an alternative simplification method specifically geared toward computing Laplacian eigen-structures. It first simulates heat propagation at small time scales by matrix multiplication (which is much faster than eigen-decomposition). It then chooses the local heat accumulation points at a very small time scale as a set of sub-sampled vertices. The goal is to try to preserve main heat information in sub-sampled vertices. Some details can be found in Appendix B. This method is denoted as *HKS-simp*. This is the method we adopt in our current shape retrieval software.

Both *Qslim-simp* and *HKS-simp* are much more efficient than the Shift-Invert spectral transform method. For an elephant data set with 1.5 million vertices, *QSlim-simp* takes 37 seconds and *HKS-simp* takes 1032 seconds to simplify it to 10K vertices, and then to compute its persistent *HKS* maxima. However, we observe that *HKS-simp* tends to preserve the *HKS* function information better. See Table 4.1 where we compare the reconstructed *HKS* functions by our *HKS-simp* simplification and QSlim simplification methods with the "ground truth" (computed using the original mesh). The accuracy is measured by both the averaged L_1-norm distance and by the statistical correlations between the reconstructed *HKS* function and the ground truth. The error under the root-mean-square distance also shows a similar trend. We show the comparisons for several different t values for *HKS*. To be consistent with all our experiments in this paper, the unit time τ is fixed to be 0.0002 across all models for computation of the Laplacian eigenvectors. All the models here originally have $30K$ vertices, and are simplified to $3K$ vertices.

Models		time t used for computing *HKS*		
		5τ	40τ	100τ
Dragon	ours	4.90 / .882	0.22 / .998	0.09 / .999
	QSlim	26.5 / .363	0.77 / .981	0.33 / .999
Armadillo	ours	3.51 / .868	0.93 / .984	0.35 / .993
	QSlim	4.07 / .830	0.99 / .972	0.37 / .991
Neptune	ours	9.83 / .927	0.50 / .994	0.23 / .999
	QSlim	26.0 / .610	1.12 / .976	0.53 / .996

Table 1: *Each entry: the averaged L_1-error / Statistical correlation between the approximated HKS and ground truth.*

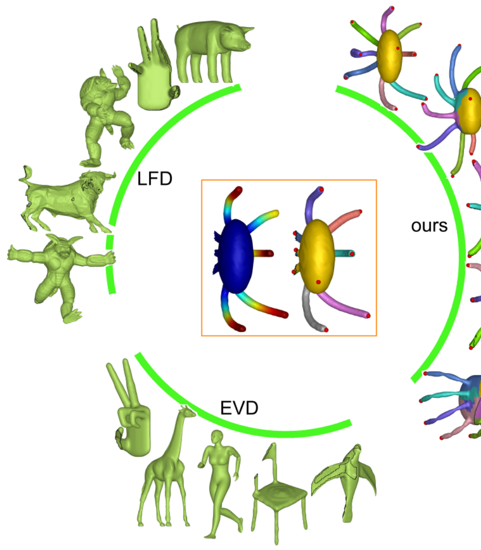

Figure 4: *Top five matches for an incomplete Octopus query model by our algorithm, EVD, and LFD.*

5. Results

Database. To test our matching method, we build a database of 300 shapes divided into 21 classes (dogs, horses, airplanes, chairs, glasses, humans, etc.), where each class has more than one shape including partial and incomplete versions. We create our own database mainly because we want to emphasize the robustness of our method for pose-oblivious partial and incomplete matching and existing shape databases do not necessarily cater to this need. Models in our database are collected from available sources [CGF09, SZM*08], to which we add partial and incomplete versions of them. The partial scan data is generated by taking a random viewing direction and keeping all vertices of a complete model that are visible from this direction. The incomplete data is generated by removing one or more portions off a complete model using cutting planes. For every pose of a model, only one version of it (i.e, either the complete model, its partial scan, or the incomplete version of it) is in the database or in the set of query models. There are 50 query models. Among them, 18 are complete models, and 32 are partial or incomplete models. Finally, we scale each shape to a unit bounding box to factor out the global scaling.

Parameters. For all models, we compute mesh-Laplacian and its eigenvectors using a universal time-unit $\tau = 0.0002$. For each model, we then compute 15 persistent maxima for the *HKS* function at time 5τ, which we observe provides robust feature points across partial and incomplete models. We then construct, for every persistent maximum, a feature vector of 15D as described in Section 4. A shape is now represented by 15 feature vectors, each of dimension 15.

Hit rate. Given a query shape, we compute the matching score between it and every shape in the database. The top 10 matches for some queries are shown in Table 3. We also compute a *hit rate* as follows. For a query shape in a class, there is a *Top-k hit* if a model is retrieved from the same class within the top *k* matches. For *N* query shapes, the Top-k hit rate is the percentage of the Top-k hits with respect to *N*.

We compare our method with two competitive shape retrieval methods: Eigenvalue descriptor (EVD) method of [JZ07] and Light Field Distribution (LFD) method of [CTSO03]. We chose these two methods because EVD represents a state-of-the-art spectral technique for articulated shape retrieval, and LFD represents a technique that has been reported to be one of the best in literature for rigid models [SMKF04]. Table 2 shows the Top-3 and Top-5 hit rates of all three methods for 50 query models.

We find that, if pose variations are disallowed and models are complete or even near-complete, all three methods work very well achieving a hit rate close to 100% with LFD performing the best (details of this result omitted here). If only pose variations are allowed, EVD performs quite well for retrieving different pose variations of input queries as the second row in Table 2 shows. However, for pose-altered models with significant incompleteness our method beats LFD and EVD by large margin as the first row in Table 2 exhibits. An example of the top 5 matches returned for an incomplete Octopus query by these methods is shown in Figure 4. An additional advantage of our method over EVD and LFD is that it can also provide correspondences between matching features in models.

#queries	**ours**	**EVD**	**LFD**
32 incompl.	88% / 91%	62% / 62%	56% / 59%
18 compl.	78% / 83%	100% / 100%	39% / 39%
50 total	84% / 88%	76% / 76%	50% / 52%

Table 2: *Each entry shows Top-3 / Top-5 hit rates for our method, EVD, and LFD. "32 incompl." includes **both** partial and incomplete queries and "18 compl." includes only pose-altered queries. The database contains 300 models.*

Timing data. We present the timing data of all three algorithms in Table 4. To make our algorithm scalable, we use the exact algorithm for models with 25K vertices or less, and run the approximation algorithm *HKS-simp* to compute *HKS* for larger data. For EVD algorithm, the timing reported is obtained by an optimized version of the original implementation obtained from the authors. To handle large meshes using EVD, we also follow their original strategy to first decimate the models with QSlim to 3K vertices, and then process them for matching. For LFD algorithm, we use the original implementation from the authors. LFD is slowest in terms of the retrieval time though its preprocessing time for models is the best. EVD has the fastest retrieval time and its preprocessing

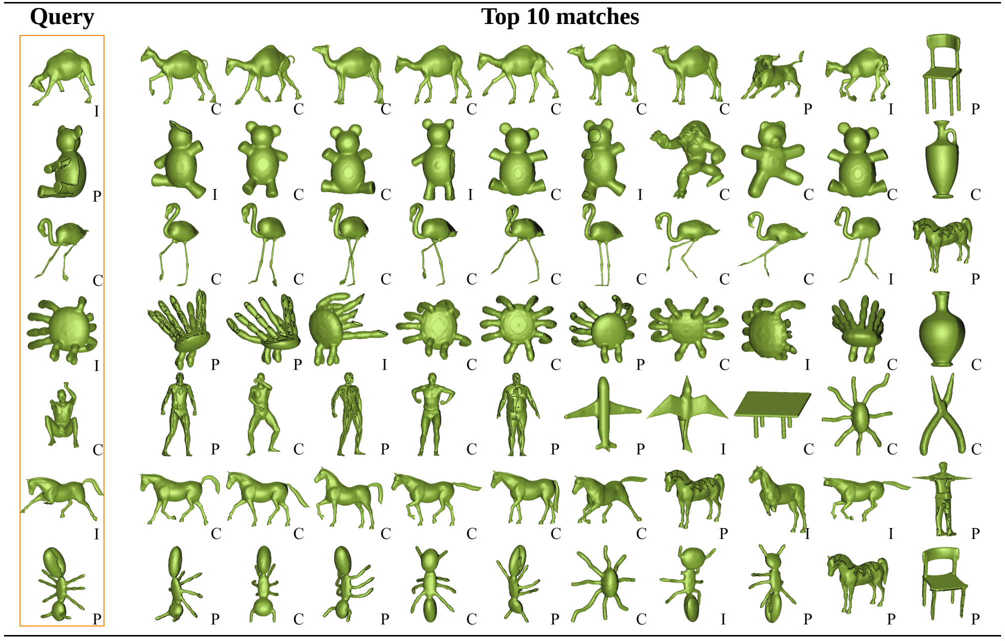

Table 3: *Query models and top ten matches returned for each query. Letters* C, P, I *indicate complete, partial, and incomplete models respectively.*

		Pre-processing (sec)		
Model	**#v**	**ours**	**EVD**	**LFD**
Plier	4.8k	9.4	7.9	0.7
Hand	8.7k	19.8	8.1	1.0
Octopus	11.0k	25.2	8.5	1.3
Teddy	12.6k	30.7	8.6	1.4
Human	15.2k	42.8	8.9	1.5
Dragon	1000k	712.0	28.0	40.0
Elephant	1500k	1032.0	37.0	60.0
Retrieval	**time (sec)**	0.02	0.006	36.0

Table 4: *All experiments are carried out on a Dell computer with Intel 2.4GHz CPU and 6GB RAM.*

time is comparable to ours, although its accuracy for partial and incomplete query shapes is worse than our algorithm.

Validity of Persistent Maxima. We choose persistent maxima as feature points for matching. We provide some experimental results to validate this choice. First, Figure 2 shows that the persistent maxima and their induced segmentations are rather robust in practice for different models in various poses, and in their partial and incomplete versions.

Next, we conduct the following experiment: we replace the *HKS* maxima by extrema of the average geodesic distance (AGD) as in [ZSCO*08] and by extrema of discrete Gaussian curvature (GC) computed using the method from [MDSB02]. We compute the top 15 persistent AGD maxima and persistent GC maxima by the region merging algorithm and then construct feature vectors for these feature points to match shapes. The Top-3 and Top-5 hit rates of the three algorithms using different feature points (i.e, persistent *HKS*, AGD, or GD maxima) are reported in Table 5. AGD takes global information into account, thus does not match well for partial / incomplete models. GC takes only a very local neighborhood into account, thus the function value at each point does not reflect global features very well, and performs the worst. Interestingly, the hit-rate for the GC method improves if one first *sparsify* the set of potential feature points by keeping only one extremum within every neighborhood of an appropriate size. However, there is no clear method to choose this appropriate "neighborhood" automatically.

#queries	**ours (*HKS*)**	**AGD**	**GC**
32 incompl.	88% / 91%	78% / 81%	38% / 41%
18 compl.	78% / 83%	72% / 94%	83% / 89%
50 total	84% / 88%	76% / 86%	55% / 58%

Table 5: *Each entry shows Top-3 / Top-5 hit rates for our method using persistent HKS, persistent AGD and persistent GC feature points.*

6. Conclusions

In this paper we combine techniques from spectral theory and computational topology to design a method for matching partial and incomplete shapes. Heat Kernel Signature functions from spectral theory provide a way of capturing features of a shape at various scales. However, discretization and approximations at various stages inject noise into this function which requires a filtering. We achieve this filtering by identifying maxima of this function that are persistent. Heat values of these maxima at different time scales become the signature of the shape with which a simple scoring scheme can be adopted for matching.

Our results show that the method is quite effective in shape matching. It outperforms existing techniques for pose-oblivious matching of partial and incomplete models. Our current method requires manifold meshes, although it can tolerate mild discrepancies in this regard. It would be interesting to investigate whether this method can be adapted to handle triangle soups in general. We also remark that our algorithm fails to match shapes in the category "Snake". This can be attributed to the fact that snakes do not possess many "features" that *HKS* can describe. It will be interesting to investigate how to augment our algorithm to handle such shapes.

We compute eigenvectors of the mesh-Laplacian without enforcing any boundary condition. It is believed that the mesh-Laplacian assumes certain implicit boundary condition though it has not been shown what this boundary condition is. It will be interesting to investigate if our method can be improved by changing the boundary condition, by using say, the high-order finite element method based approach in [Reu10].

Acknowledgment. The authors thank anonymous reviewers for helpful comments. This work is supported by NSF grants CCF-0830467, CCF-0915996, and CCF-0747082.

References

[BSW08] BELKIN M., SUN J., WANG Y.: Discrete laplace operator on meshed surfaces. In *Proc. 24th Annu. ACM Sympos. Comput. Geom.* (2008), pp. 278–287. 3, 6

[CGF09] CHEN X., GOLOVINSKIY A., , FUNKHOUSER T.: A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH) 28*, 3 (2009). 7

[CGOS09] CHAZAL F., GUIBAS L. J., OUDOT S., SKRABA P.: Analysis of scalar fields over point cloud data. In *Proc. 20th ACM-SIAM Sympos. Discrete Algs.* (2009), pp. 1021–1030. 5

[CJ96] CHUA C., JARVIS R.: Point signatures: A new representation for 3D object recognition. *International Journal of Computer Vision 25*, 1 (1996), 63–85. 2

[CSEH09] COHEN-STEINER D., EDELSBRUNNER H., HARER J.: Extending persistence using poincaré and lefschetz duality. *Foundations of Computational Mathematics 9*, 1 (2009), 79–103. 4

[CSEM06] COHEN-STEINER D., EDELSBRUNNER H., MOROZOV D.: Vines and vineyards by updating persistence in linear time. In *Proc. 22nd Annu. ACM Sympos. Comput. Geom.* (2006), pp. 119–126. 10

[CTSO03] CHEN D.-Y., TIAN X. P., SHEN Y.-T., OUHYOUNG M.: On visual similarity based 3d model retrieval. *Computer Graphics Forum 22*, 3 (2003), 223–232. 1, 7

[DMSB99] DESBRUN M., MEYER M., SCHRÖDER P., BARR A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. *Computer Graphics 33*, Annual Conference Series (1999), 317–324. 3

[Dod78] DODZIUK J.: Finite-difference approach to the hodge theory of harmonic forms. *American Journal of Mathematics 98*, 1 (1978), 79–104. 3

[DRW10] DEY T. K., RAJAN P., WANG Y.: Convergence, stability, and discrete approximation of Laplace spectra. In *Proc. 21st ACM-SIAM Sympos. Discrete Algs.* (2010). 3

[ELZ02] EDELSBRUNNER H., LETSCHER D., ZOMORODIAN A.: Topological persistence and simplification. *Discrete Comput. Geom. 28* (2002), 511–533. 2, 4, 10

[FS06] FUNKHOUSER T., SHILANE P.: Partial matching of 3d shapes with priority-driven search. In *SGP* (2006), pp. 131–142. 1

[Gar] GARLAND M.: qslim 2.1: The QSlim mesh simplification software. URL: http://graphics.cs.uiuc.edu/~garland/software/qslim.html. 6

[GCO06] GAL R., COHEN-OR D.: Salient geometric features for partial shape matching and similarity. *ACM Transactions on Graphics 25(1)* (2006), 130–150. 1

[GMGP05] GELFAND N., MITRA N. J., GUIBAS L. J., POTTMANN H.: Robust global registration. In *Proc. Symp. Geom. Processing (SGP)* (2005), pp. 197–206. 2

[JZ07] JAIN V., ZHANG H.: A spectral approach to shape-based retrieval of articulated 3d models. *Comput. Aided Des. 39*, 5 (2007), 398–407. 1, 6, 7

[LÓ6] LÉVY B.: Laplace-Beltrami eigenfunctions towards an algorithm that "understands" geometry. In *Internat. Conf. Shape Model. Applications* (2006). 1, 3

[MDSB02] MEYER M., DESBRUN M., SCHRÖDER P., BARR A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. In *VisMath'02* (2002). 8

[OBBG09] OVSJANIKOV M., BRONSTEIN A. M., BRONSTEIN M. M., GUIBAS L. J.: Shape google: a computer vision approach to invariant shape retrieval. In *Proc. NORDIA* (2009). 1, 2

[OFCD01] OSADA R., FUNKHOUSER T., CHAZELLE B., DOBKIN D.: Matching 3d models with shape distributions. In *Shape Modeling International* (2001), p. 154. 1

[PP93] PINKALL U., POLTHIER K.: Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics 2*, 1 (1993), 15–36. 3

[Reu10] REUTER M.: Hierarchical shape segmentation and registration via topological features of laplace-beltrami eigenfunctions. *Internat. J. Computer Vision 89*, 2 (2010), 287–308. 1, 9

[Rus07] RUSTAMOV R. M.: Laplace-Beltrami eigenfunctions for deformation invariant shape representation. In *SGP* (2007), pp. 225–233. 1

[RWP06] REUTER M., WOLTER F.-E., PEINECKE N.: Laplace-beltrami spectra as "shape-dna" of surfaces and solids. *Computer-Aided Design 38*, 4 (2006), 342–366. 1, 3

[SF06] SHILANE P., FUNKHOUSER T.: Selecting distinctive 3d shape descriptors for similarity retrieval. In *Shape Modeling International* (2006). 2

[SMKF04] SHILANE P., MIN P., KAZHDAN M., FUNKHOUSER T.: The Princeton shape benchmark. In *Proc. Shape Model. Internat. (SMI)* (2004), pp. 167–178. 1, 7

[SOG09] Sun J., Ovsjanikov M., Guibas L. J.: A concise and provably informative multi-scale signature based on heat diffusion. *Comput. Graph. Forum 28*, 5 (2009), 1383–1392. 1, 2, 3

[SZM*08] Siddiqi K., Zhang J., Macrini D., Shokoufandeh A., Bouix S., Dickinson S.: Retrieving articulated 3D models using medial surfaces. *Machine Vision and Applications 19*, 4 (2008), 261–274. 7

[TV08] Tangelder J. W., Veltkamp R. C.: A survey of content based 3d shape retrieval methods. *Multimedia Tools Appl. 39*, 3 (2008), 441–471. 1

[VL08] Vallet B., Lévy B.: Spectral geometry processing with manifold harmonics. *Computer Graphics Forum (Proceedings Eurographics) 27* (2008), 251–260. 6

[War06] Wardetzky M.: *Discrete Differential Operators on Polyhedral Surfaces – Convergence and Approximation*. PhD thesis, Freie Universität Berlin, 2006. 3

[ZSCO*08] Zhang H., Sheffer A., Cohen-Or D., Zhou Q., van Kaick O., Tagliasacchi A.: Deformation-drive shape correspondence. *Computer Graphics Forum 27*, 5 (2008), 1431–1439. 8

[ZvKD10] Zhang H., van Kaick O., Dyer R.: Spectral mesh processing. *Computer Graphics Forum* (2010). to appear. 3

Appendix A: Proof for Proposition 2

Recall that at any stage of the merging process, we maintain a central triangle t and its pair edge e for every region R. We always choose the region with the current smallest persistence for merging. It is easy to verify that this greedy strategy maintains the following invariant: t has the highest $\bar{h}$ value among all triangles in R, and e has the highest $\bar{h}$ value among all edges in the boundary ∂R.

We prove the proposition by taking the matrix view of the persistence algorithm which reduces the initial incidence matrix iteratively. In the original persistence algorithm [ELZ02], when a new triangle t is considered, columns for certain triangles which are on the left (older triangles) are added to the column of t till the lowest 1 in the column of t does not have another such lowest 1 in its row. This lowest 1 corresponds to the edge to which t pairs. If no such 1 exists, t is declared unpaired. It turns out that [CSEM06], it is not necessary to execute this algorithm sequentially as proposed in the original persistence algorithm. Let D be the original edge-triangle incidence matrix that respects the filtration order. One can add columns of D arbitrarily with the constraint that a column is added only to a column on its right, i.e., a triangle's column is added only to another triangle's column which is younger. If these additions provide a unique lowest 1 for each triangle column, then Cohen-Steiner et al. showed that the triangle-edge pairing is same as the original persistence pairing [CSEM06].

In our merging algorithm we are simulating this column additions with the guarantee that we reach a unique pairing. Each column of a triangle t represents the boundary of the region R it is currently central to. The edge e currently paired with t corresponds to the lowest 1 in this column (as it has the highest function value among all edges in ∂R). We merge the region R with minimum persistence value to another region R' adjacent to e. That is, we add the column of t to the column of another triangle t' which is central to R', and t' has 1 at the entry corresponding to e.

Since e is also in $\partial R'$, it is necessary that the pairing edge e' for t' at this point has a function value that is at least as large as that of e; i.e, $\bar{h}(e') \geq \bar{h}(e)$. On the other hand, as R is the region currently with the smallest persistence, we have $\mathrm{p}(R) \leq \mathrm{p}(R')$. It follows that $\bar{h}(t) \leq \bar{h}(t')$. If $\bar{h}(t) = \bar{h}(t')$, then recall that we break the tie by considering first the central triangle that comes earlier in the filtration order. Hence whether $\bar{h}(t) < \bar{h}(t')$ or $\bar{h}(t) = \bar{h}(t')$, the central triangle t' of R' must be younger than the central triangle of R. This means our algorithm simulates the column addition to the right. Furthermore, when we merge R with R', the paired edge of its central triangle is absorbed in the merged region $R \cup R'$ as we consider modulo-2 addition in persistence homology. This edge will not be paired in the future. Using induction, one can show that the central triangle of R pairs uniquely with its pair edge at the time of merging. The first statement of proposition 2 follows.

Since we merge in non-decreasing order of persistence values of the regions, combining the first half of the proposition with Proposition 1, the second half of the claim follows.

Appendix B: *HKS-simp* Method to simplify large mesh

Given a mesh K with vertex set $V = \{v_1, \ldots, v_n\}$, we first construct a *discrete heat operator* w.r.t. a very small time τ by $H_\tau = I - \tau D^{-1} L$, where L is the discrete mesh-Laplacian, D is a diagonal matrix with $D[i][i] = \frac{1}{4\pi h} \sum_k A_k e^{-\frac{\|v_i - v_k\|^2}{4h}}$, and A_i is one-third of the total area of all triangles incident to vertex v_i. Intuitively, $D[i][i]$ is the sum of weights of all vertices w.r.t v_i that are used to compute L. This discrete heat operator is derived from the first order Taylor expansion of the relation $\mathcal{H}_t = e^{-t\triangle}$.

By definition, $HKS_t(x) = \mathbf{h_t}(x,x) = \mathcal{H}_t \delta_x$, where δ_x is the Dirac delta function. Translated into the discrete setting, this means that $HKS_\tau(v_i) = H_\tau \mathbf{u}_i[i] = H_\tau[i][i]$, where the unit vector $\mathbf{u}_i = [0, \ldots, 0, 1, 0, \ldots, 0]$ is the discrete analog of the Dirac delta function δ_{v_i}. Furthermore, the heat distribution after $\alpha\tau$ time is $H_\tau^\alpha \mathbf{u}_i$, implying that $HKS_{\alpha\tau}(v_i) = H_\tau^\alpha[i][i]$. Hence we can approximate HKS values at time $\alpha\tau$ by taking the diagonal of the matrix H_τ^α. Computing H_τ^α by matrix multiplication is much more efficient for small α than computing the eigen-decomposition of the Laplacian matrix. However, this method is only accurate when α is small. Therefore, we do not directly use this method to approximate the HKS for large t values. Instead, we take a specific small α and approximate the function $HKS_{\alpha h}$ as described above. $HKS_{\alpha h}$ encodes the short-time behavior of heat propagation and reflects the local geometry of the input surface K. We sub-sample K by keeping only the set of local maxima $\widehat{V}$ of

$HKS_{\alpha\tau}$ which tend to include all future feature points (HKS maxima for larger t). The parameter α controls the resolution of simplification.