Eurographics Symposium on Point-Based Graphics (2007)
M. Botsch, R. Pajarola (Editors)

Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition

Gurjeet Singh[1] , Facundo Mémoli[2] and Gunnar Carlsson[†2]

[1]Institute for Computational and Mathematical Engineering, Stanford University, California, USA.
[2]Department of Mathematics, Stanford University, California, USA.

Abstract

We present a computational method for extracting simple descriptions of high dimensional data sets in the form of simplicial complexes. Our method, called `Mapper`*, is based on the idea of* partial clustering *of the data guided by a set of functions defined on the data. The proposed method is not dependent on any particular clustering algorithm, i.e. any clustering algorithm may be used with* `Mapper`*. We implement this method and present a few sample applications in which simple descriptions of the data present important information about its structure.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modelling.

1. Introduction

The purpose of this paper is to introduce a new method for the qualitative analysis, simplification and visualization of high dimensional data sets, as well as the qualitative analysis of functions on these data sets. In many cases, data coming from real applications is massive and it is not possible to visualize and discern structure even in low dimensional projections. As a motivating example consider the data being collected by the Oceanic Metagenomics collection [DAG*07], [SGD*07], which has many millions of protein sequences which are very difficult to analyze due to the volume of the data. Another example is the database of patches in natural images studied in [LPM03]. This data set also has millions of points and is known to have a simple structure which is obscured due to its immense size.

We propose a method which can be used to reduce high dimensional data sets into simplicial complexes with far fewer points which can capture topological and geometric information at a specified resolution. We refer to our method as `Mapper` in the rest of the paper. The idea is to provide another tool for a generalized notion of coordinatization for high dimensional data sets. Coordinatization can of course refer to a choice of real valued coordinate functions on a data set, but other notions of geometric representation (e.g., the Reeb graph [Ree46]) are often useful and reflect interesting information more directly. Our construction provides a coordinatization not by using real valued coordinate functions, but by providing a more discrete and combinatorial object, a simplicial complex, to which the data set maps and which can represent the data set in a useful way. This representation is demonstrated in Section 5.1, where this method is applied to a data set of diabetes patients. Our construction is more general than the Reeb graph and can also represent higher dimensional objects, such as spheres, tori, etc. In the simplest case one can imagine reducing high dimensional data sets to a graph which has nodes corresponding to clusters in the data. We begin by introducing a few general properties of `Mapper`.

Our method is based on *topological* ideas, by which we roughly mean that it preserves a notion of nearness, but can distort large scale distances. This is often a desirable property, because while distance functions often encode a notion of similarity or nearness, the large scale distances often carry little meaning.

The method begins with a data set X and a real valued function $f : X \to \mathbb{R}$, to produce a graph. This function can be a

† All authors supported by DARPA grant HR0011-05-1-0007. GC additionally supported by NSF DMS 0354543.

function which reflects geometric properties of the data set, such as the result of a density estimator, or can be a user defined function, which reflects properties of the data being studied. In the first case, one is attempting to obtain information about the qualitative properties of the data set itself, and in the second case one is trying to understand how these properties interact with interesting functions on the data set. The functions determine the space to which we produce a map. The method can easily be modified to deal with maps to parameter spaces other than $\mathbb{R}$, such as $\mathbb{R}^2$ or the unit circle S^1 in the plane. In the first of these cases, one produces a two dimensional simplicial complex, together with a natural map from the data set to it. In the second case, one constructs a graph with a map from the graph to a circle. In the case where the target parameter space is $\mathbb{R}$, our construction amounts to a stochastic version of the *Reeb graph* (see [Ree46]) associated with the filter function. If the covering of $\mathbb{R}$ is too coarse, we will be constructing an image of the Reeb graph of the function, while if it is fine enough we will recover the Reeb graph precisely.

The basic idea can be referred to as *partial clustering*, in that a key step is to apply standard clustering algorithms to subsets of the original data set, and then to understand the interaction of the partial clusters formed in this way with each other. That is, if U and V are subsets of the data set, and $U \cap V$ is non-empty, then the clusters obtained from U and V respectively may have non-empty intersections, and these intersections are used in building a simplicial complex. This construction produces a "multiresolution" or "multiscale" image of the data set. One can actually construct a family of simplicial complexes (graphs in the case of a one-dimensional parameter space), which are viewed as images at varying levels of coarseness, and maps between them moving from a complex at one resolution to one of coarser resolution. This fact allows one to assess the extent to which features are "real" as opposed to "artifacts", since features which persist over a range of values of the coarseness would be viewed as being less likely to be artifacts.

We do not attempt to obtain a fully accurate representation of a data set, but rather a low-dimensional image which is easy to understand, and which can point to areas of interest. Note that it is implicit in the method that one fixes a parameter space, and its dimension will be an upper bound on the dimension of the simplicial complex one studies. As such, it is in a certain way analogous to the idea of a Postnikov tower or the coskeletal filtration in algebraic topology [Hat02].

1.1. Previous work

We now summarize the relationships between our method and existing methods for the analysis and visualization of high-dimensional data sets. The *projection pursuit* method (see [Hub85]) determines the linear projection on two or three dimensional space which optimizes a certain heuristic criterion. It is frequently very successful, and when it succeeds it produces a set in $\mathbb{R}^2$ or $\mathbb{R}^3$ which readily visualizable. Other methods (Isomap [TSL00], locally linear embedding [RS00], multidimensional scaling [Abd07]) attempt to find non-linear maps to Euclidean space which preserve the distance functions on the data set to as high a degree as possible. They also produce useful two and three dimensional versions of data sets when they succeed. All three of these constructions are quite sensitive to distance metric chosen, and their output is a subset of Euclidean space. Also these methods cannot produce simplicial complexes directly. One could use a further stage which uses the output of the MDS algorithm for producing a simplicial complex. However, in contrast with mapper, the size of the resulting simplicial complexes is at least as large as the original dataset, thus not achieving any simplification. In contrast, `Mapper` is able to achieve substantial simplifications and at the same time that the resulting simplicial complex preservs certain topological structures from the original datset. In the domain of Shape Comparison and Matching, ideas with some similarity to our were presented in [BFS00].

1.2. Our work

We present a method which is less sensitive to the metric, and produces a combinatorial object (a simplicial complex), whose interconnections reflect some aspects of the metric structure. It is not required to be embedded in Euclidean space, although in the case of a one-dimensional complex, it can always be embedded in $\mathbb{R}^3$. Also, the `Mapper` construction produces a multiresolution representation, which produces images of the data set at various levels of resolution. There are other constructions which also produce combinatorial rather than Euclidean output, notably *disconnectivity graphs* [BK97] and *cluster trees*. These constructions could in principle also be used to provide multiresolution output, but they are limited to dimension one output, and always produce trees. As we have indicated, our output can be based not only on maps to $\mathbb{R}$, but to higher dimensional spaces or to the circle, producing either higher dimensional complexes or graphs which can potentially have cycles. The graphs may display cycles even in the case when the parameter space is $\mathbb{R}$, as we will demonstrate in our examples.

1.3. Outline

The rest of this paper is organized as follows. Section 2 describes the underlying theoretical framework which supports `Mapper`. We will outline the topological construction which provides the motivation for the construction and give the construction in detail. Section 3 is a description of the algorithm and implementation details. Section 4 describes a few natural functions which can be used to explore data sets with `Mapper`. We illustrate the use of `Mapper` in a few sample applications in Section 5 including an example

of application of `Mapper` to shape comparison. In Section 6, we conclude with a discussion.

2. Construction

Although the interest in this construction comes from applying it to point cloud data and functions on point cloud data, it is motivated by well known constructions in topology. In the interest of clarity, we will introduce this theoretical construction first, and then proceed to develop the analogous construction for point cloud data. We will refer to the theoretical construction as the *topological version* and to the point cloud analogue as the *statistical version*.

2.1. Topological background and motivation

The construction in this paper is motivated by the following construction. See [Mun99] for background on topological spaces, and [Hat02] for information about simplicial complexes. Given a finite covering $\mathcal{U} = \{U_\alpha\}_{\alpha\in A}$ of a space X, we define the *nerve* of the covering $\mathcal{U}$ to be the simplicial complex $N(\mathcal{U})$ whose vertex set is the indexing set A, and where a family $\{\alpha_0,\alpha_1,\ldots,\alpha_k\}$ spans a k-simplex in $N(\mathcal{U})$ if and only if $U_{\alpha_0}\cap U_{\alpha_1}\cap\ldots\cap U_{\alpha_k}\neq\emptyset$. Given an additional piece of information, a *partition of unity*, one can obtain a map from X to $N(\mathcal{U})$. A partition of unity *subordinate to the finite open covering* $\mathcal{U}$ is a family of real valued functions $\{\varphi_{\alpha\in A}\}_{\alpha\in A}$ with the following properties.

- $0\leq\varphi_\alpha(x)\leq 1$ for all $\alpha\in A$ and $x\in X$.
- $\sum_{\alpha\in A}\varphi_\alpha(x)=1$ for all $x\in X$.
- The closure of the set $\{x\in X|\varphi_\alpha(x)>0\}$ is contained in the open set U_α.

We recall that if $\{v_0,v_1,\ldots,v_k\}$ are the vertices of a simplex, then the points v in the simplex correspond in a one-to-one and onto way to the set of ordered k-tuples of real numbers $(r_0,r_1,\ldots,r_k)$ which satisfy $0\leq r_i\leq 1$ and $\sum_{i=0}^k r_i=1$. This correspondence is called the *barycentric coordinatization*, and the numbers r_i are referred to as the *barycentric coordinates* of the point v. Next, for any point $x\in X$, we let $\mathcal{T}(x)\subseteq A$ be the set of all α so that $x\in U_\alpha$. We now define $\rho(x)\in N(\mathcal{U})$ to be the point in the simplex spanned by the vertices $\alpha\in\mathcal{T}(x)$, whose barycentric coordinates are $(\varphi_{\alpha_0}(x),\varphi_{\alpha_1}(x),\ldots,\varphi_{\alpha_l}(x))$, where $\{\alpha_0,\alpha_1,\ldots,\alpha_l\}$ is an enumeration of the set $\mathcal{T}(x)$. The map ρ can easily be checked to be continuous, and provides a kind of partial coordinatization of X, with values in the simplicial complex $N(\mathcal{U})$.

Now suppose that we are given a space equipped with a continuous map $f:X\to Z$ to a parameter space Z, and that the space Z is equipped with a covering $\mathcal{U}=\{U_\alpha\}_{\alpha\in A}$, again for some finite indexing set A. Since f is continuous, the sets $f^{-1}(U_\alpha)$ also form an open covering of X. For each α, we can now consider the decomposition of $f^{-1}(U_\alpha)$ into its path connected components, so we write $f^{-1}(U_\alpha)=\bigcup_{i=1}^{j_\alpha}V(\alpha,i)$, where j_α is the number of connected components in $f^{-1}(U_\alpha)$. We write $\overline{\mathcal{U}}$ for the covering of X obtained this way from the covering $\mathcal{U}$ of Z.

2.2. Multiresolution structure

If we have two coverings $\mathcal{U}=\{U_\alpha\}_{\alpha\in A}$ and $\mathcal{V}=\{V_\beta\}_{\beta\in B}$ of a space X, a *map of coverings* from $\mathcal{U}$ to $\mathcal{V}$ is a function $f:A\to B$ so that for all $\alpha\in A$, we have $U_\alpha\subseteq V_{f(\alpha)}$ for all $\alpha\in A$.

Example 2.1 *Let $X=[0,N]\subseteq\mathbb{R}$, and let $\varepsilon>0$. The sets $I_l^\varepsilon=(l-\varepsilon,l+1+\varepsilon)\cap X$, for $l=0,1,\ldots,N-1$ form an open covering $\mathcal{I}_\varepsilon$ of X. All the coverings $\mathcal{I}_\varepsilon$ for the different values of ε have the same indexing set, and for $\varepsilon\leq\varepsilon'$, the identity map on this indexing set is a map of coverings, since $I_l^\varepsilon\subseteq I_l^\varepsilon$.*

Example 2.2 *Let $X=[0,2N]$ again, and let I_l^ε be as above, for $l=0,1,\ldots,2N-1$, and let $J_m^\varepsilon=(2m-\varepsilon,2m+2+\varepsilon)\cap X$. Let $\mathcal{J}_\varepsilon$ denote the covering $\{J_0^\varepsilon,J_1^\varepsilon,\ldots,J_{N-1}^\varepsilon\}$. Let $f:\{0,1,\ldots,2N-1\}\to\{0,1,\ldots,N-1\}$ be the function $f(l)=\lfloor\frac{l}{2}\rfloor$. Then f gives a map of coverings $\mathcal{I}_\varepsilon\to\mathcal{J}_{\varepsilon'}$ whenever $\varepsilon\leq\varepsilon'$.*

Example 2.3 *Let $X=[0,N]\times[0,N]\subseteq\mathbb{R}^2$. Given $\varepsilon>0$, we let $B_\varepsilon(i,j)$ be the set $(i-\varepsilon,i+1+\varepsilon)\times(j-\varepsilon,j+1+\varepsilon)$. The collection $\{B_\varepsilon(i,j)\}$ for $0\leq i,j\leq N-1$ provides a covering $\mathcal{B}_\varepsilon$ of X, and the identity map on the indexing set $\{(i,j)|0\leq i,j\leq N-1\}$ is a map of coverings $\mathcal{B}_\varepsilon\to\mathcal{B}_{\varepsilon'}$ whenever $\varepsilon\leq\varepsilon'$. A doubling strategy such as the one described in Example 2.2 above also works here.*

We next observe that if we are given a map of coverings from $\mathcal{U}=\{U_\alpha\}_{\alpha\in A}$ to $\mathcal{V}=\{V_\beta\}_{\beta\in B}$, i.e. a map of sets $f:A\to B$ satisfying the conditions above, there is an induced map of simplicial complexes $N(f):N(\mathcal{U})\to N(\mathcal{V})$, given on vertices by the map f. Consequently, if we have a family of coverings $\mathcal{U}_i$, $i=0,1,\ldots,n$, and maps of coverings $f_i:\mathcal{U}_i\to\mathcal{U}_{i+1}$ for each i, we obtain a diagram of simplicial complexes and simplicial maps

$$N(\mathcal{U}_0)\overset{N(f_0)}{\to}N(\mathcal{U}_1)\overset{N(f_1)}{\to}\cdots\overset{N(f_{n-1})}{\to}N(\mathcal{U}_N)$$

When we consider a space X equipped with a $f:X\to Z$ to a parameter space Z, and we are given a map of coverings $\mathcal{U}\to\mathcal{V}$, there is a corresponding map of coverings $\overline{\mathcal{U}}\to\overline{\mathcal{V}}$ of the space X. To see this, we only need to note that if $U\subseteq V$, then of course $f^{-1}U\hookrightarrow f^{-1}(V)$, and consequently it is clear that each connected component of $f^{-1}(U)$ is included in exactly one connected component of $f^{-1}(V)$. So, the map of coverings from $\overline{\mathcal{U}}$ to $\overline{\mathcal{V}}$ is given by requiring that the set $U_\alpha(i)$ is sent to the unique set of the form $V_{f(\beta)}(j)$ so that $U_\alpha(i)\subseteq V_{f(\beta)}(j)$.

2.3. Examples

We illustrate how the methods work for the topological version.

Example 2.4 *Consider the situation where X is $[-M,M] \subseteq \mathbb{R}$, the parameter space is $[0,+\infty)$, and the function $f : X \to \mathbb{R}$ is the probability density function for a Gaussian distribution, given by $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$. The covering $\mathcal{U}$ of Z consists of the 4 subsets $\{[0,5),(4,10),(9,15),(14,+\infty)\}$, and we assume that N is so large that $f(N) > 14$. One notes that $f^{-1}([0,5))$ consists of a single component, but that $f^{-1}((4,10))$, $f^{-1}((9,15))$, and $f^{-1}((14,+\infty))$ all consist of two distinct components, one on the positive half line and the other on the negative half line. The associated simplicial complex now looks as follows.*

It is useful to label the nodes of the simplicial complex by color and size. The color of a node indicates the value of the function f (red being high and blue being low) at a representative point in the corresponding set of the cover $\overline{\mathcal{U}}$, or perhaps by a suitable average taken over the set. The size of a node indicates the number of points in the set represented by the node. In this way, the complex provides information about the nature of the function.

Example 2.5 *Let $X = \mathbb{R}^2$, and let the map be given by applying the Gaussian density function from the previous example to $r = \sqrt{x^2+y^2}$. We use the same covering $\mathcal{U}$ as in the previous example. We now find that all the sets $f^{-1}U$, for all $U \in \mathcal{U}$, are connected, so the simplicial complex will have only four vertices, and will look like this.*

When we color label the nodes, we see that this situation is essentially different from that in the previous example.

Example 2.6 *Consider the situation where we are given a rooted tree X, where Z is again the non-negative real line, and where the function $f(x)$ is defined to be the distance from the root to the point x in a suitably defined tree distance. In this case, when suitable choices of the parameter values are made, the method will recover a homeomorphic version of the tree.*

Example 2.7 *Let X denote the unit circle $\{(x,y)|x^2+y^2=1\}$ in the Euclidean plane, let Z denote $[-1,1]$, and let $f(x,y) = y$. Let $\mathcal{U}$ be the covering $\{[-1,-\frac{2}{3}),(-\frac{1}{2},\frac{1}{2}),(\frac{2}{3},1]\}$. Then the associated covering $\overline{\mathcal{U}}$ is now pictured as follows. We note that $f^{-1}([-1,-\frac{2}{3}))$ and $f^{-1}((\frac{2}{3},1])$ both consist of one connected component, while $f^{-1}((-\frac{1}{2},\frac{1}{2}))$ consists of two connected components. It is now easy to see that the simplicial complex will have four vertices, and will look as follows:*

3. Implementation

In this section, we describe the implementation of a statistical version of `Mapper` which we have developed for point cloud data. The main idea in passing from the topological version to the statistical version is that clustering should be regarded as the statistical version of the geometric notion of partitioning a space into its connected components. We assume that the point cloud contains N points $x \in X$, and that we have a function $f : X \to \mathbb{R}$ whose value is known for the N data points. We call this function a *filter*. Also, we assume that it is possible to compute inter-point distances between the points in the data. Specifically, it should be possible to construct a distance matrix of inter-point distances between sets of points.

We begin by finding the range of the function (I) restricted to the given points. To find a covering of the given data, we divide this range into a set of smaller intervals ($\mathcal{S}$) which overlap. This gives us two parameters which can be used to control resolution namely the length of the smaller intervals (l) and the percentage overlap between successive intervals (p).

Example 3.1 Let $I = [0-2]$, $l = 1$ and $p = \frac{2}{3}$. The set $\mathcal{S}$ would then be $\mathcal{S} = \{[0,1],[0.33,1.33],[0.66,1.66],[1,2]\}$

Now, for each interval $I_j \in \mathcal{S}$, we find the set $X_j = \{x|f(x) \in I_j\}$ of points which form its domain. Clearly the set $\{X_j\}$ forms a cover of X, and $X \subseteq \bigcup_j X_j$. For each smaller set X_j we find clusters $\{X_{jk}\}$. We treat each cluster as a vertex in our complex and draw an edge between vertices whenever $X_{jk} \cap X_{lm} \neq \emptyset$ i.e. the clusters corresponding to the vertices have non-empty intersection.

Example 3.2 Consider point cloud data which is sampled from a noisy circle in $\mathbb{R}^2$, and the filter $f(x) = ||x-p||_2$, where p is the left most point in the data (refer to Figure 1). We cover this data set by a set of 5 intervals, and for each interval we find its clustering. As we move from the low end of the filter to the high end, we see that the number of clusters changes from 1 to 2 and then back to 1, which are connected as shown in Figure 1.

3.1. Clustering

Finding a good clustering of the points is a fundamental issue in computing a representative simplicial complex. `Mapper` does not place any conditions on the clustering algorithm. Thus any domain-specific clustering algorithm can be used.

We implemented a clustering algorithm for testing the ideas

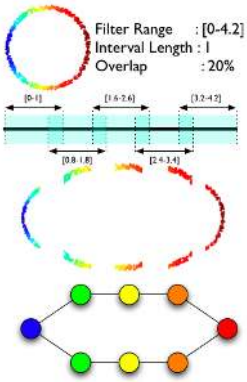

Figure 1: *Refer to Example 3.2. The data is sampled from a noisy circle, and the filter used is $f(x) = ||x-p||_2$, where p is the left most point in the data. The data set is shown on the top left, colored by the value of the filter. We divide the range of the filter into 5 intervals which have length 1 and a 20% overlap. For each interval we compute the clustering of the points lying within the domain of the filter restricted to the interval, and connect the clusters whenever they have non empty intersection. At the bottom is the simplicial complex which we recover whose vertices are colored by the average filter value.*

presented here. The desired characteristics of the clustering were:

1. Take the inter-point distance matrix ($D \in \mathbb{R}^{N\times N}$) as an input. We did not want to be restricted to data in Euclidean Space.
2. Do not require specifying the number of clusters beforehand.

We have implemented an algorithm based on *single-linkage clustering* [Joh67], [JD88]. This algorithm returns a vector $C \in \mathbb{R}^{N-1}$ which holds the length of the edge which was added to reduce the number of clusters by one at each step in the algorithm.

Now, to find the number of clusters we use the edge length at which each cluster was merged. The heuristic is that the inter-point distance within each cluster would be smaller than the distance between clusters, so shorter edges are required to connect points within each cluster, but relatively longer edges are required to merge the clusters. If we look at the histogram of edge lengths in C, it is observed experimentally, that shorter edges which connect points within each cluster have a relatively smooth distribution and the edges which are required to merge the clusters are disjoint from this in the histogram. If we determine the histogram of C using k intervals, then we expect to find a set of empty interval(s) after which the edges which are required to merge the clusters appear. If we allow all edges of length shorter than the length at which we observe the empty interval in the histogram, then we can recover a clustering of the data.

Increasing k will increase the number of clusters we observe and decreasing k will reduce it. Although this heuristic has worked well for many datasets that we have tried, it suffers from the following limitations: (1) If the clusters have very different densities, it will tend to pick out clusters of high density only. (2) It is possible to construct examples where the clusters are distributed in such a way such that we recover the incorrect clustering. Due to such limitations, this part of the procedure is open to exploration and change in the future.

3.2. Higher Dimensional Parameter Spaces

Using a single function as a filter we get as output a complex in which the highest dimension of simplices is 1 (edges in a graph). Qualitatively, the only information we get out of this is the number of components, the number of loops and knowledge about structure of the component flares etc.). To get information about higher dimensional voids in the data one would need to build a higher dimensional complex using more functions on the data. In general, the `Mapper` construction requires as input: (a) A Parameter space defined by the functions and (b) a covering of this space. Note that *any* covering of the parameter space may be used. As an example of the parameter space S^1, consider a parameter space defined by two functions f and g which are related such that $f^2+g^2 = 1$. A very simple covering for such a space is generated by considering overlapping angular intervals.

One natural way of building higher dimensional complexes is to associate many functions with each data point instead of just one. If we used M functions and let $\mathbb{R}^M$ to be our parameter space, then we would have to find a covering of an M dimensional hypercube which is defined by the ranges of the M functions.

Example 3.3 Consider using two functions f and g which are defined for each data point (refer to Figure 2). We need to define a covering of the rectangle $\mathcal{R} = [\min f, \max f] \times [\min g, \max g]$. This covering defines constraints on values of f and g within each region, which enables us to select subsets of the data. As in the case of covering an interval, the regions which cover $\mathcal{R}$ must overlap. Now, if we cover $\mathcal{R}$ using hexagons then we can adjust the size and overlap of hexagons such that a maximum of three hexagons intersect. Thus, the dimension of simplices which we use to construct the complex will always be 3 or less. On the other hand if we cover $\mathcal{R}$ using rectangles, there will be regions where four rectangles intersect. Thus, the dimension of simplices which we use to construct the complex will be 4 or less.

We now describe the `Mapper` algorithm using two functions and the parameter space $\mathbb{R}^2$. Consider two functions on each data point, and the range of these being covered by rectangles. Define a region $\mathcal{R} = [\min f_1, \max f_1] \times [\min f_2, \max f_2]$. Now say we have a covering $\cup_{i,j} A_{ij}$ such

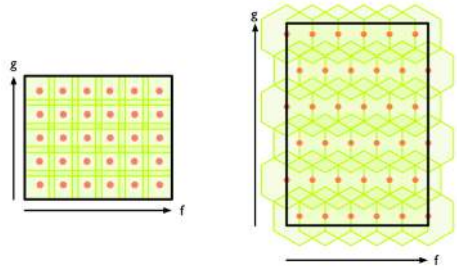

Figure 2: *Covering the range of two functions f and g. The area which needs to be covered is $[\min f, \max f] \times [\min g, \max g]$. On the left is a covering using rectangles and on the right is a covering using hexagons. The constraints on the smaller regions (rectangles or hexagons) define the indices of data which we pick. The red dots in each represent the center of the region. Refer to Example 3.3 for details. Please refer to the electronic version for color image.*

that each $A_{i,j}, A_{i+1,j}$ intersect and each $A_{i,j}, A_{i,j+1}$ intersect. An algorithm for building a reduced simplicial complex is:

1. For each i, j, select all data points for which the function values of f_1 and f_2 lie within $A_{i,j}$. Find a clustering of points for this set and consider each cluster to represent a 0 dimensional simplex (referred to as a *vertex* in this algorithm). Also, maintain a list of vertices for each $A_{i,j}$ and a set of indices of the data points (the cluster members) associated with each vertex.
2. For all vertices in the sets $\{A_{i,j}, A_{i+1,j}, A_{i,j+1}, A_{i+1,j+1}\}$, if the intersection of the cluster associated with the vertices is non-empty then add a 1-simplex (referred to as an *edge* in this algorithm).
3. Whenever clusters corresponding to any three vertices have non empty intersection, add a corresponding 2 simplex (referred to as a *triangle* in this algorithm) with the three vertices forming its vertex set.
4. Whenever clusters corresponding to any four vertices have non-empty intersection, add a 3 simplex (referred to as *tetrahedron* in this algorithm) with the four vertices forming its vertex set.

It is very easy to extend `Mapper` to the parameter space $\mathbb{R}^M$ in a similar fashion.

Example 3.4 Consider the unit sphere in $\mathbb{R}^3$. Refer to Figure 3. The functions are $f_1(x) = x_3$ and $f_2(x) = x_1$, where $x = (x_1, x_2, x_3)$. As intervals in the range of f_1 and f_2 are scanned, we select points from the dataset whose function values lie in both the intervals and then perform clustering. In case of a sphere, clearly only three possibilities exist:

1. The intersection is empty, and we get no clusters.
2. The intersection contains only one cluster.
3. The intersection contains two clusters.

After finding clusters for the covering, we form higher dimensional simplices as described above. We then used the homology detection software PLEX ([PdS]) to analyze the resulting complex and to verify that this procedure recovers the correct Betti numbers: $\beta_0 = 1, \beta_1 = 0, \beta_2 = 1$.

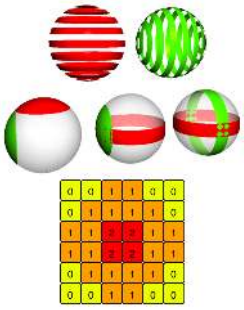

Figure 3: *Refer to Example 3.4 for details. Let the filtering functions be $f_1(x) = x_3$, $f_2(x) = x_1$, where x_i is the i^{th} coordinate. The top two images just show the contours of the function f_1 and f_2 respectively. The three images in the middle row illustrate the possible clusterings as the ranges of f_1 and f_2 are scanned. The image in the bottom row shows the number of clusters as each region in the range$(f_1) \times$ range(f_2) is considered. Please refer to the electronic version for color image.*

4. Functions

The outcome of `Mapper` is highly dependent on the function(s) chosen to partition (filter) the data set. In this section we identify a few functions which carry interesting geometric information about data sets in general. The functions which are introduced below rely on the ability to compute distances between points. We assume that we are given a collection of N points as a point cloud data X together with a distance function $d(x,y)$ which denotes the distance between $x, y \in X$.

4.1. Density

Density estimation is a highly developed area within statistics. See [Sil86], for a thorough treatment. In particular for $\varepsilon > 0$ consider estimating density using a Gaussian kernel as:

$$f_\varepsilon(x) = C_\varepsilon \sum_y \exp\left(\frac{-d(x,y)^2}{\varepsilon}\right)$$

where $x, y \in X$ and C_ε is a constant such that $\int f_\varepsilon(x)\,dx = 1$. In this formulation ε controls the smoothness of the estimate of the density function on the data set, estimators using large values of ε correspond to smoothed out versions of the estimator using smaller values of this parameter. A number of other interesting methods are presented in [Sil86] and many of them depend only the ability to compute distance between members of the point cloud. As such, they yield functions which carry information about the geometry of the data set.

4.2. Eccentricity

This is a family of functions which also carry information about the geometry of the data set. The basic idea is to identify points which are, in an intuitive sense, far from the center, without actually identifying an actual center point. Given p with $1 \leq p < +\infty$, we set

$$E_p(x) = \left(\frac{\sum_{y \in X} d(x,y)^p}{N} \right)^{\frac{1}{p}} \tag{4–1}$$

where $x, y \in X$. We may extend the definition to $p = +\infty$ by setting $E_\infty(x) = \max_{x' \in X} d(x,x')$. In the case of a Gaussian distribution, this function is clearly negatively correlated with density. In general, it tends to take larger values on points which are far removed from a "center".

4.3. Graph Laplacians

This family of functions originates from considering a Laplacian operator on a graph defined as follows (See [LL06] for a thorough treatment). The vertex set of this graph is the set of all points in the point cloud data X, and the weight of the edge between points $x, y \in X$ is:

$$w(x,y) = k(d(x,y))$$

where d denotes the distance function in the point cloud data and k is, roughly, a "smoothing kernel" such as a Gaussian kernel. A (normalized) graph Laplacian matrix is computed as:

$$L(x,y) = \frac{w(x,y)}{\sqrt{\sum_z w(x,z)} \sqrt{\sum_z w(y,z)}}$$

Now, the eigenvectors of the normalized graph Laplacian matrix gives us a set of orthogonal vectors which encode interesting geometric information, [LL06] and can be used as filter functions on the data.

5. Sample Applications

In this section, we discuss a few applications of the `Mapper` algorithm using our implementation. Our aim is to demonstrate the usefulness of reducing a point cloud to a much smaller simplicial complex in synthetic examples and some real data sets.

We have implemented the `Mapper` algorithm for computing and visualizing a representative graph (derived using one function on the data) and the algorithm for computing a higher order complex using multiple functions on the data. Our implementation is in MATLAB and utilizes GraphViz for visualization of the reduced graphs.

We use the following definitions in this section. `Mapper` reduces an input point cloud Y to a simplicial complex $\mathcal{C}$. Let the vertex set of this complex be X, and the 1-skeleton of $\mathcal{C}$ be G. All vertices in the set X represent clusters of data. Let

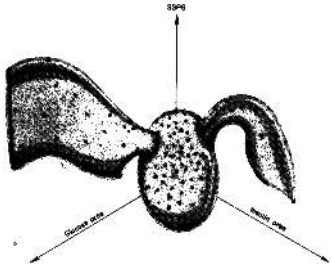

Figure 4: *Refer to Section 5.1 for details. Three dimensional projection of the diabetes data obtained using projection and pursuit.*

C_i be the cluster of points in Y associated with X_i. Let N_i be the cardinality of the cluster associated with X_i. Recall from Section 3.1, that we have a distance matrix D for points in the set Y. By using it, we can also associate a metric between the members of X. We define two notions of distance as:

- $D_H(X_i,X_j) = \max\left(\frac{\sum_y \min_x d(x,y)}{N_j}, \frac{\sum_x \min_y d(x,y)}{N_i} \right)$, where $x \in C_i, y \in C_j$. Informally, this is a smooth approximation to the Hausdorff distance between two sets.
- We construct an adjacency matrix A for G, where $A(i,j) = D_H(X_i,X_j)$ if there is an edge between X_i and X_j in G. Now, computing graph distance using Dijkstra's algorithm gives us an "intrinsic" distance on the output. Let this be D_I.

We scale both D_H and D_I such that the maximum distance is 1 so as to normalize them.

5.1. The Miller-Reaven diabetes study

In [Mil85], G. M. Reaven and R.G.Miller describe the results they obtain by applying the projection pursuit method [Hub85] to data [AH85] obtained from a study performed at Stanford University in the 1970's. 145 patients who had diabetes, a family history of diabetes, who wanted a physical examination, or to participate in a scientific study participated in the study. For each patient, six quantities were measured: age, relative weight, fasting plasma glucose, area under the plasma glucose curve for the three hour glucose tolerance test (OGTT), area under the plasma insulin curve for the (OGTT), and steady state plasma glucose response. This created a 6 dimensional data set, which was studied using projection pursuit methods, obtaining a projection into three dimensional Euclidean space, under which the data set appears as in Figure 4. Miller and Reaven noted that the data set consisted of a central core, and two "flares" emanating from it. The patients in each of the flares were regarded as suffering from essentially different diseases, which correspond to the division of diabetes into the adult onset and juvenile onset forms. One way in which we wish to use `Mapper` is as an automatic tool for detecting such flares in the data, even in situations where projections into two or three dimensional

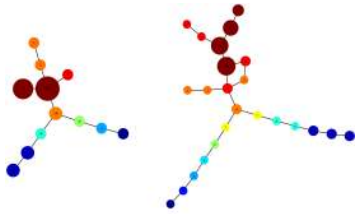

Figure 5: *Refer to Section 5.1 for details. On the left is a "low-resolution" Mapper output which was computed using 3 intervals in the range of the filter with a 50% overlap. On the right is a "high-resolution" Mapper output computed using 4 intervals in the range of the filter with a 50% overlap. The colors encode the density values, with red indicative of high density, and blue of low. The size of the node and the number in it indicate the size of the cluster. The low density ends reflect type I and type II diabetes. The flares occurring in Figure 4 occur here as flares with blue ends. Please refer to the electronic version for color image.*

space do not provide such a good image. Figure 5 shows the results obtained by applying Mapper to this same data set, using density estimated by a kernel estimator. We show two different resolutions.

5.2. Mapper on Torus

We generated 1500 points evenly sampled on the surface of a two dimensional torus (with inner radius 0.5 and exterior radius 1) in $\mathbb{R}^3$. We embedded this torus into $\mathbb{R}^{30}$ by first padding dimensions 4 to 30 with zeros and then applying a random rotation to the resulting point cloud. We computed the first two non-trivial eigenfunctions of the Laplacian, f_1 and f_2 (see Section 4.3) and used them as filter functions for Mapper. Other parameters for the procedure were as follows. The number of intervals in the range of f_1 and f_2 was 8 and any two adjacent intervals in the range of f_i had 50% overlap. The output was a set of 325 (clusters of) points together with a four dimensional simplicial complex. The 3-D visualization shown in Figure 6 was obtained by first endowing the output points with the metric D_H as defined above and using Matlab's MDS function mdscale and then attaching 1 and 2-simplices inferred from the four dimensional simplicial complex returned by Mapper. The three-dimensional renderings of the 2-skeleton are colored by the functions f_1 and f_2. These experiments were performed only to verify that the embedding produced by using the inferred distance metric actually looked like a torus and to demonstrate that the abstract simplicial complex returned by Mapper has the correct Betti numbers: $\beta_0 = 1, \beta_1 = 2, \beta_2 = 1$ (as computed using PLEX).

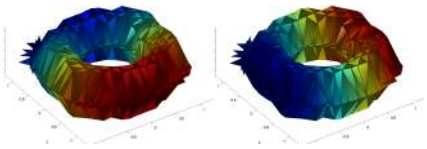

Figure 6: *Refer to Section 5.2 for details. Please refer to the electronic version for color image.*

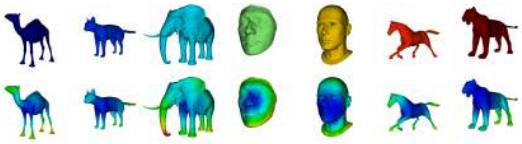

Figure 7: *Refer to Section 5.3 for details. The top row shows the rendering of one model from each of the 7 classes. The bottom row shows the same model colored by the E_1 function (setting $p = 1$ in equation 4–1) computed on the mesh. Please refer to the electronic version for color image.*

5.3. Mapper on 3D Shape Database

In this section we apply Mapper to a collection of 3D shapes from a publicly available database of objects [SP]. These shapes correspond to seven different classes of objects (camels, cats, elephants, faces, heads, horses and lions). For each class there are between 9 and 11 objects, which are different *poses* of the same shape, see Figure 7.

This repository contains many 3D shapes in the form of triangulated meshes. We preprocessed the shapes as follows. Let the set of points be P. From these points, we selected 4000 *landmark* points using a Euclidean maxmin procedure as described in [dSVG04]. Let this set be $Y = \{P_i, i \in L\}$, where L is the set of indices of the landmarks.

In order to use this as the point cloud input to Mapper, we computed distances between the points in Y as follows. First, we computed the adjacency matrix A for the set P by using the mesh information e.g. if P_i and P_j were connected on the given mesh, then $A(i,j) = d(P_i, P_j)$, where $d(x,y)$ is the Euclidean distance between $x, y \in P$. Finally, the matrix of distances D between points of Y was computed using Dijkstra's algorithm on the graph specified by A.

In order to apply Mapper to this set of shapes we chose to use $E_1(x)$ as our filter function (setting $p = 1$ in equation 4–1), see Figure 7. In order to minimize the effect of bias due to the distribution of local features we used a global threshold for the clustering within all intervals which was determined as follows. We found the threshold for each interval by the histogram heuristic described in Section 3.1, and used the median of these thresholds as the global threshold.

The output of Mapper in this case (single filter function) is a graph. We use GraphViz to produce a visualization of this graph. Mapper results on a few shapes from the database

are presented in Figure 9. A few things to note in these results are:

1. `Mapper` is able to recover the graph representing the skeleton of the shape with fair accuracy. As an example, consider the horse shapes. In both cases, the three branches at the bottom of the recovered graph represent the font two legs and the neck. The blue colored section of the graph represents the torso and the top three branches represent the hind legs and the tail. In these examples, endowing the nodes of the recovered graph with the mean position of the clusters they represent would recover a skeleton.
2. Different poses of the same shape have qualitatively similar `Mapper` results however different shapes produce significantly different results. This suggests that certain intrinsic information about the shapes, which is invariant to pose, is being retained by our procedure.

The behaviour exhibited by `Mapper` (using E_1 as filter) suggest it may be useful as a tool for simplifying shapes and subsequently performing database query and (pose invariant) shape comparison tasks. We briefly explore this possibility in the next section.

5.4. Shape Comparison Example

In this section we show how `Mapper`'s ability to meaningfully simplify data sets could be used for facilitating Shape Comparison/Matching tasks. We use the output of `Mapper` as simplified objects/shapes on which we will be performing a shape discrimination task.

Following the same procedure detailed in the Section 5.3, we reduce an input shape (Y) to a graph (G). Let the vertex set of this graph be X and the adjacency matrix by A. For each X_i, we define $\mu_i = \frac{N_i}{\sum_j N_j}$ to be used later as a weight associated with each point of the simplified shape.

Each simplified object is then specified as a triple $\{(X, D_X, \mu_X)\}$, where D_X is a distance matrix which is computed using one of the choices D_H or D_I as defined earlier and μ_X is the set of weights for X. We used the method described in [Mem07] to estimate a measure of dissimilarity between all the shapes. The point to make here is that the clustering procedure underlying the `Mapper` construction provides us with not only a distance between clusters but also with a natural notion of *weight* for each point (cluster) in the simplified models, where both encode interesting information about the shape. The comparison method takes as input both the distance matrix between all pairs of points in the simplified model and the weight for each point. It then proceeds to compute an L_p version of the Gromov-Hausdorff distance. The output of this stage is a dissimilarity matrix $\mathcal{D}$ where element $\mathcal{D}(X,Z)$ expresses the dissimilarity between (simplified) objects X and Z.

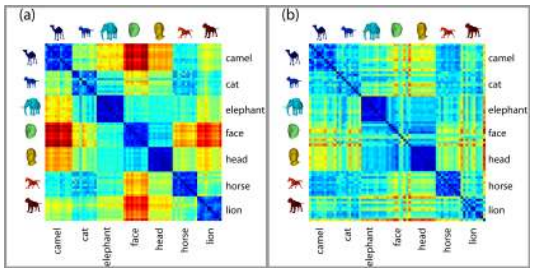

Figure 8: *Refer to Section 5.4 for details. Comparing dissimilarity matrices $\mathcal{D}$: (a) This dissimilarity matrix was computed using D_H. (b) This dissimilarity matrix was computed using D_I. Clearly D_H is much better at clustering various poses of the same shape together. Please refer to the electronic version for color image.*

We tried this method on two different constructions for the simplified sets. The first construction estimated the metric D_X of simplified model X as D_H defined above. The second construction estimates the metric as D_I defined above. Figure 8 depicts the resulting dissimilarity matrices for all simplified models, for both methods of construction. Note the clear discrimination in case (a).

We quantified the ability to discriminate different classes of shapes by computing a probability of error in classification. Let C_X denote the class to which the shape X belongs. From each class of shapes C_i we randomly pick a landmark pose L_i to represent that class. We have 7 classes in the data, so our landmark set is $L = \{L_i, i = 1 \dots 7\}$. Now, to each shape X, we assign the *implied* class $\widehat{C_X} = C_Y$ where $Y = \arg\min_{Z \in L} \mathcal{D}(X,Z)$. Note that $\widehat{C_X}$ depends on the choice of the landmark set L. We define the per class probability of error for a particular choice of L as:

$$P_{C_i}^{(L)} = \frac{\#\left(\{X|\widehat{C_X} \neq C_i\} \cap \{X|C_X = C_i\}\right)}{\#(\{X|C_X = C_i\})}$$

Now, the probability of error for a particular choice of L is $P^{(L)} = \sum_i P_{C_i}^{(L)} P_{C_i}$ where $P_{C_i} = \frac{\#(\{X|C_X=C_i\})}{\sum_i \#(\{X|C_X=C_i\})}$. Since the choice of L is random, we repeat the above procedure M times and find the probability of error as $P = \sum_{i=1}^{M} \frac{P^{(L_i)}}{M}$.

We calculated the probability of error for the two cases: (a) When D_H is used to find $\mathcal{D}$ and (b) when D_I is used to find $\mathcal{D}$. In the former case P was found to be 3.03% and in the latter case P was found to be 23.41%. In both cases we used $M = 100000$. Note that despite having reduced a shape with 4000 points to less than 100 for most classes, the procedure manages to classify shapes with a low error probability.

The idea of using topological methods associated with the filtering functions for simplifying shape comparison has been considered before, e.g. [BFS00]. Our approach naturally offers more information as it provides a measure of im-

portance of the the vertices of the resulting simplicial complex (simplified shape).

6. Conclusions

We have devised a method for constructing useful combinatorial representations of geometric information about high dimensional point cloud data. Instead of acting directly on the data set, it assumes a choice of a filter or combination of filters, which can be viewed as a map to a metric space, and builds an informative representation based on clustering the various subsets of the data set associated the choices of filter values. The target space may be Euclidean space, but it might also be a circle, a torus, a tree, or other metric space. The input requires only knowledge of the distances between points and a choice of combination of filters, and produces a multiresolution representation based on that filter. The method provides a common framework which includes the notions of density clustering trees, disconnectivity graphs, and Reeb graphs, but which substantially generalizes all three. We have also demonstrated how the method could be used to provide an effective shape discriminator when applied to interesting families of shapes. An important direction for future work on the method is improving the method for performing the partial clustering, specifically for choosing a reasonable scale on the clusters. Our current method, while ad hoc, performs reasonably well in the examples we study, but a more principled method, perhaps including some local adaptivity, would doubtlessly improve the results.

References

[Abd07] ABDI H.: Metric multidimensional scaling. In *Encyclopedia of Measurement and Statistics*. Sage, Thousand Oaks (Ca), 2007, pp. 598–605.

[AH85] ANDREWS D. F., HERZBERG A. M.: *Data : a collection of problems from many fields for the student and research worker*. Springer-Verlag, New York, 1985.

[BFS00] BIASOTTI S., FALCIDIENO B., SPAGNUOLO M.: Extended reeb graphs for surface understanding and description. In *DGCI '00: Proceedings of the 9th International Conference on Discrete Geometry for Computer Imagery* (London, UK, 2000), Springer-Verlag, pp. 185–197.

[BK97] BECKER O. M., KARPLUS M.: The topology of multidimensional potential energy surfaces: Theory and application to peptide structure and kinetics. *The Journal of Chemical Physics 106*, 4 (1997), 1495–1517.

[DAG*07] DB R., AL H., G S., KB H., ET AL. W. S.: The sorcerer ii global ocean sampling expedition: Northwest atlantic through eastern tropical pacific. *PLoS Biology 5*, 3 (2007).

[dSVG04] DE SILVA V., G. C.: Topological estimation using witness complexes. In *Symposium on Point-Based Graphics* (2004), pp. 157–166.

[Hat02] HATCHER A.: *Algebraic topology*. Cambridge University Press, Cambridge, 2002.

[Hub85] HUBER P. J.: Projection pursuit. *Ann. Statist. 13*, 2 (1985), 435–525. With discussion.

[JD88] JAIN A. K., DUBES R. C.: *Algorithms for clustering data*. Prentice Hall Advanced Reference Series. Prentice Hall Inc., Englewood Cliffs, NJ, 1988.

[Joh67] JOHNSON S. C.: Hierarchical clustering schemes. *Psychometrika 2* (1967), 241–254.

[LL06] LAFON S., LEE A. B.: Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28*, 9 (2006), 1393–1403.

[LPM03] LEE A. B., PEDERSEN K. S., MUMFORD D.: The nonlinear statistics of high-contrast patches in natural images. *Int. J. Comput. Vision 54*, 1-3 (2003), 83–103.

[Mem07] MEMOLI F.: On the use of gromov-hausdorff distances for shape comparison. In *Symposium on Point-Based Graphics* (2007).

[Mil85] MILLER R. J.: Discussion - projection pursuit. *Ann. Statist. 13*, 2 (1985), 510–513. With discussion.

[Mun99] MUNKRES J. R.: *Topology*. Prentice-Hall Inc., Englewood Cliffs, N.J., 1999.

[PdS] PARRY P., DE SILVA V.: Plex: Simplicial complexes in matlab. `http://comptop.stanford.edu/programs/`.

[Ree46] REEB G.: Sur les points singuliers d'une forme de Pfaff complètement intégrable ou d'une fonction numérique. *C. R. Acad. Sci. Paris 222* (1946), 847–849.

[RS00] ROWEIS S. T., SAUL L. K.: Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science 290*, 5500 (2000), 2323–2326.

[SGD*07] S Y., G S., DB R., AL H., ET AL. W. S.: The sorcerer ii global ocean sampling expedition: Expanding the universe of protein families. *PLoS Biology 5*, 3 (2007).

[Sil86] SILVERMAN B. W.: *Density estimation for statistics and data analysis*. Monographs on Statistics and Applied Probability. Chapman & Hall, London, 1986.

[SP] SUMNER R. W., POPOVIC J.: Mesh data from deformation transfer for triangle meshes. `http://people.csail.mit.edu/sumner/research/deftransfer/data.html`.

[TSL00] TENENBAUM J. B., SILVA V. D., LANGFORD J. C.: A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science 290*, 5500 (2000), 2319–2323.

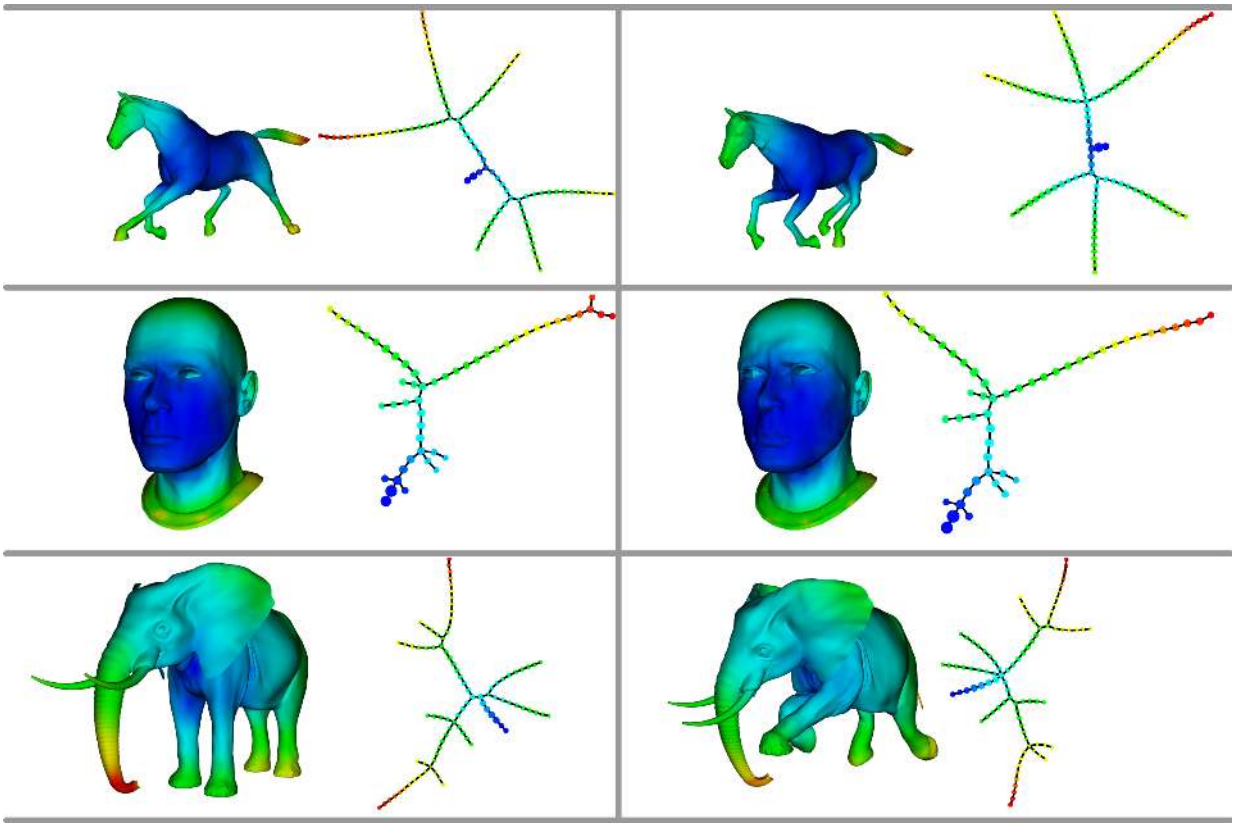

Figure 9: *Refer to Section 5.3 for details. Each row of this image shows two poses of the same shape along with the* `Mapper` *result which is computed as described in Section 5.3. For each* `Mapper` *computation, we used 15 intervals in the range of the filter with a* 50% *overlap.*