DOI: 10.1111/j.1467-8659.2010.01783.x

COMPUTER GRAPHICS *forum*
Volume 29 (2010), number 8 pp. 2479–2491

3D Surface Reconstruction Using a Generalized Distance Function

R. Poranne[1], C. Gotsman[1] and D. Keren[2]

[1]Computer Science Department, Technion – Israel Institute of Technology, Haifa, Israel
{roip, gotsman}@cs.technion.ac.il
[2]Department of Computer Science, University of Haifa, Haifa, Israel
dkeren@cs.haifa.ac.il

Abstract

We define a generalized distance function on an unoriented 3D point set and describe how it may be used to reconstruct a surface approximating these points. This distance function is shown to be a Mahalanobis distance in a higher-dimensional embedding space of the points, and the resulting reconstruction algorithm a natural extension of the classical Radial Basis Function (RBF) approach. Experimental results show the superiority of our reconstruction algorithm to RBF and other methods in a variety of practical scenarios.

Keywords: surface reconstruction, distance measure, point clouds

ACM CCS: I.3.2 [Computer Graphics]: Graphics Systems (C.2.1, C.2.4, C.3); I.3.5 [Computer Graphics]: Curve, surface, solid, and object representations

1. Introduction

Three-dimensional (3D) scanning is the process of acquiring a digital copy of a physical object. A 3D scanner is used to sample points on the surface of the object (the so-called *underlying surface*) and acquire their Cartesian coordinates, after which a surface reconstruction algorithm is applied to generate a surface based on this sample set.

The problem is, of course, ill-posed, as this is basically an interpolation or approximation problem. Most current methods can be roughly classified into two approaches. The first is based on Voronoi diagrams and their Delaunay duals, starting with [Boi84], and later developed extensively (e.g. [KSO04, ACK01, AB98]). These triangulate the input point set in hope of achieving a triangle mesh approximating the underlying surface. But this method is sensitive to noise and requires a dense point set to give good results.

In the alternative approach of implicit functions, most notably those expressed using Radial Basis Functions [CBC*01, DTS02], a scalar function $d(x, y, z)$ is defined on $\mathbb{R}^3$, such that d approximates the signed Euclidean distance from the underlying surface [CBC*01, SAAY06, SGS05, WCS05]. The construction of d is essentially an interpolation (or approximation) of zero values on the input data points, and the underlying surface is then extracted as the zero set of d. The key to good results using this method is a well-behaved function d, one that achieves zero values on the data and close to it, and whose absolute value increases with distance from the data. A function not having these properties may result in many spurious components in the zero set. Because essentially no additional information, apart from the approximate location of the zeros of d, is present in the input, nothing really prevents points distant from the data having small values. Thus, in practice, to achieve reasonable results, this approach usually requires extra information, beyond the surface samples. This is typically supplied in the form of so-called *inside–outside points*, which provide hints to which points are inside the surface (hence tagged with negative values of d), and which are outside the surface (hence tagged with positive values of d). This type of data is typically not available, and even if it is, it is difficult to decide what precise values of

d to assign these extra off-surface points. Furthermore, the surface may not be closed, or it may have one-dimensional parts protruding from it, in which case it will not have a clear-cut 'inside' or 'outside'.

In the case that normal information is available at the samples, this can be used to generate inside/outside points close to the surface, or provide information on the gradient of an *indicator* function, namely a function that has the value 1 inside the surface and 0 outside. Kazhdan *et al.* [KBH06] show how computing such an indicator function reduces to solving a Poisson equation. Once an indicator function is obtained, extracting the level set $d = 1/2$ yields the desired surface. Alliez *et al.* [ACSTD07] show how to estimate and use unoriented normal information from point sets.

In this paper, we describe a novel construction of an *unsigned* distance function from sample data (without normal information). This function obtains very small values on the data, which increase smoothly and monotonically with distance from the data. Thus, almost no spurious components will be present in the reconstruction. The price we pay for this almost ideal function is twofold: higher computational complexity in its construction and extra difficulty in extracting the surface than when using a signed distance function, because no precise constant value can be considered the representative value of the surface. The common Marching Cubes algorithm is not so useful and we must resort to more sophisticated surface extraction techniques.

2. Radial Basis Functions

We start by reviewing the implicit function approach based on so-called *radial basis functions* (RBF). Consider the problem of fitting a surface to a set of N points $X = \{x_i\}_{i=1}^N \subseteq \mathbb{R}^3$, given no additional information on the surface. A popular approach is to find some approximation to a signed distance function from the surface, and then define the surface as its zero set [CBC*01]. The main (and only) assumption is that the function obtains zero values close to the data. The distance function d is usually represented as a linear combination of RBF

$$d(x) = \sum_{j=1}^{M} \alpha_j \varphi(||x - c_j||),$$

where $C = \{c_j\}_{j=1}^M$ is a set of *centre* points and φ is a positive definite function. This means that d is a linear combination of k basis functions, each obtained using the same 'template' function φ, of one variable, centred at c_j. Its value depends only on the Euclidean distance of x from c_j. The centres are usually selected in some strategic manner, and then the α_j's can be found by solving a linear system derived from its desired values at the data points (usually zero).

One inherent difficulty in this approach is that the zero constraints result in a homogeneous linear system, so the trivial solution $a_j = 0$ will result. The simplest way to avoid this is to provide off-surface points with non-zero values, but this is undesirable, because these non-zero values are somewhat arbitrary, and can have an adverse effect on the solution.

To show how a trivial solution may be avoided without off-surface points, we present a different interpretation of the RBF approach, which will provide a natural non-trivial solution, and also eventually lead to our generalized distance function. Assume first that we have RBF centres at *all* data points. Define a map $\Phi : \mathbb{R}^3 \to \mathbb{R}^N$ by

$$\Phi(x) = (\varphi(||x - x_1||), \ldots, \varphi(||x - x_N||)^\top.$$

This maps the N input points $X \subseteq \mathbb{R}^3$ to N-dimensional points $\Phi(X) = \{\Phi(x_i)\}_{i=1}^N \subseteq \mathbb{R}^N$. Because it consists of N points, the set $\Phi(X)$ will always lie on an $(N-1)$-dimensional hyperplane H in $\mathbb{R}^N$, which is the simplest possible surface imaginable in this space. Thus, we may assume that the underlying surface we seek is the set of *all* points in $\mathbb{R}^3$ which are mapped to H by Φ, namely, all x such that

$$d(x) \triangleq \sum_{j=1}^{N} \alpha_j \varphi(||x - x_j||) + \alpha_{N+1} = 0 \qquad (1)$$

for an appropriate coefficient vector $\alpha = (\alpha_1, \ldots, \alpha_{N+1})^T$. This same assumption is made in the traditional RBF approach. It is easy to see that this vector belongs to the right nullspace of the following $N \times (N+1)$ matrix A:

$$A_{i,j} = \begin{cases} \varphi(||x_i - x_j||) & j \le N, \\ 1 & j = N+1. \end{cases}$$

The dimension of this nullspace is at least one because A has more columns than rows. It will be larger than one only if the basis functions are degenerate in some way (e.g. if two data points coincide). Thus, taking α to be a null vector of A results in Equation (1) defining a non-trivial RBF interpolant d from which we can extract the surface as its zero set. It is also straightforward to see that $d(x)$ is actually proportional to the signed distance of a point $\Phi(x) \in \mathbb{R}^N$ from the hyperplane H. Note that this distance is measured in $\mathbb{R}^N$, as opposed to $\mathbb{R}^3$, so it is not the true Euclidean distance from the underlying surface in $\mathbb{R}^3$, but an approximation.

3. Generalized RBF

We can generalize the approach described in the previous section to the case where: (1) we use $M \le N$ centres $C = \{c_j\}_{J=1}^M$ (i.e. fewer basis functions) which do not necessarily coincide with the data set and (2) the surface is modelled as a hyperplane H of dimension $M - l$ in $\mathbb{R}^M$. H may be defined as the intersection of l hyperplanes of dimension $M - 1$, each containing the N points of $\Phi(X)$. This dimension reduction can be viewed as obtaining a tighter fit to X. Thus, we need to find l independent coefficient vectors $\alpha^k = (\alpha_1^k, \ldots, \alpha_{M+1}^k)$, $1 \leq k \leq l$ satisfying (1), namely, l independent vectors in the right nullspace of A, which is now the $N \times (M+1)$ matrix

$$A_{i,j} = \begin{cases} \varphi(||x_i - c_j||) & j \leq M, \\ 1 & j = M+1. \end{cases}$$

Unfortunately, because $M + 1 \leq N$, such vectors most likely do not exist, so instead we use l vectors minimizing $||Ax||$, namely, the l lower right singular vectors of A. The quality of fit of the hyperplane represented by a singular vector can be inferred from the respective singular value: smaller values mean the projection of points in $\Phi(X)$ on that vector are closer together or that their distances from the hyperspace defined by that vector are smaller.

To simplify matters a little, we redefine $\Phi(X)$ so that the mapped set is centred in all M dimensions, eliminating the component α_{M+1} in (1)

$$\Phi(x) = (\varphi(||x - c_1||) - \mu_1, \ldots, \varphi(||x - c_M||) - \mu_M)^\top$$

where

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} \varphi(||x_i - c_j||)$$

is the M-vector of the *means* of the basis functions on the data. This allows us to exchange (1) for l equations of the form

$$\sum_{j=1}^{M} \alpha_j^k(\varphi(||x - c_j||) - \mu_j) = 0 \tag{2}$$

where $1 \leq k \leq l$. Now a set of l equations of the form (2) represent a linear subspace V of $\mathbb{R}^M$ of dimension $M - l$ (a hyperplane containing the origin). The associated matrix B (analogous to A) is now

$$B_{i,j} = \varphi(||x_i - c_j||) - \mu_j$$

and V is perpendicular to the solutions α^k of (2) in the least-squares sense, namely, the l right singular vectors of B corresponding to the smallest singular values. Note that the column sums of B vanish, thus, when $M \geq N$, B's smallest singular value also vanishes. V is spanned orthogonally by the $M - l$ right singular vectors of B corresponding to the largest singular values and $V^\perp$ by the l right singular vectors of B corresponding to the smallest singular values. The generalized distance function $D(x)$ will be defined as the squared distance of $\Phi(x)$ to V which is just the sum of the squared lengths of the projections of $\Phi(x)$ on each of the dimensions of $V^\perp$, as illustrated in Figure 1.

4. The Mahalanobis Distance

It is possible to slightly modify the straightforward definition of D as a simple Euclidean distance in $\mathbb{R}^M$ to be a weighted

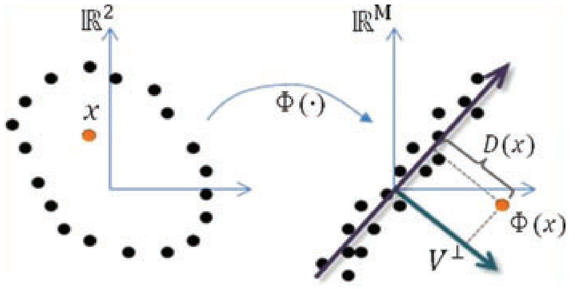

Figure 1: *Embedding the (black) data points in an appropriate higher-dimensional space using Φ makes them 'look' like a linear subspace. The distance D(x) of a point x to the data is then its (possibly weighted) distance to this subspace.*

Euclidean distance from V in $\mathbb{R}^M$ which takes into account the quality of the fit of V to $\Phi(X)$ in each of V's dimensions. Consider the following $M \times M$ covariance matrix Σ of $\Phi(X)$

$$\Sigma = B^\top B. \tag{3}$$

This covariance matrix may be used to define the following Mahalanobis distance function [DeJ00] in $\mathbb{R}^M$, commonly used in statistics:

$$D(x) = (\Phi(x)^{\mathrm{T}} \Sigma^{-1} \Phi(x))^{1/2}.$$

The covariance matrix is frequently used in principal component analysis (PCA) to determine the principal directions of scattered data, and the spread in each such direction. These directions and spreads are given by the eigenvectors and eigenvalues of the covariance matrix, and the Mahalanobis distance weights the standard Euclidean distance by the inverse variances along the principal directions. Intuitively, this means that given a point cloud with covariance Σ, and a new point p, the distance of p to the point cloud is computed such that the distance along a more variable direction contributes less to the total distance (because variation in this direction is an inherent property of the point cloud itself).

This distance function is intimately related to the spectrum of Σ. Because Σ is symmetric positive semi-definite, its eigen decomposition and singular decomposition are equivalent, and we may write

$$\Sigma = USU^{\mathrm{T}}, \qquad S = \mathrm{diag}(\sigma_1, \ldots, \sigma_M) \tag{4}$$

where $\sigma_1 > \cdots > \sigma_M$ are the eigenvalues and U is the matrix of eigenvectors of Σ. Thus, $\Sigma^{-1} = US^{-1}U^{\mathrm{T}}$ and

$$D(x) = (\Phi(x)^{\mathrm{T}} U S^{-1} U^{\mathrm{T}} \Phi(x))^{1/2} = \left(\sum_{k=1}^{M} \sigma_k^{-1} \langle \Phi(x), \alpha^k \rangle^2\right)^{1/2},$$

where α^k are the columns of U, and $\langle ., . \rangle$ denotes inner product. Because $\Sigma = B^{\mathrm{T}}B$, the eigenvectors of Σ are identical to the right singular vectors of B and the squares of the singular values of B are equal to the eigenvalues of Σ.

As in the previous section, we may choose to model the point cloud as the subspace of $\mathbb{R}^M$ of dimension $M - l$. Similarly to PCA, this is the subspace formed by the $M - l$ eigenvectors of Σ having the largest eigenvalues. Thus, the l eigenvectors of Σ having the smallest eigenvalues form an orthogonal basis of the complementary $V^\perp$, and we call it the *approximate nullspace* of Σ. All this is equivalent to using the rank-l approximation to Σ^{-1} minimizing the Frobenius norm. By the Young–Eckart theorem [GVL96], this approximation is obtained by eliminating (i.e. setting to zero) the $M - l$ smallest eigenvalues of Σ^{-1} (which are the largest eigenvalues of Σ) Thus, $D(x)$ can be approximated as

$$D(x) = \left(\sum_{k=M-l+1}^{M} \sigma_k^{-1} \langle \Phi(x), \alpha^k \rangle^2 \right)^{1/2}. \tag{5}$$

We call this distance function the 'MaD distance function' (MaD is short for *Ma*halanobis *D*istance), and remind the reader again that the traditional RBF method is the special case $l = 1$ (with the slight caveat that $D(x)$ will be the absolute value of the RBF function). Note also that the simple Euclidean distance function defined in the previous section is obtained when all σ_k in (5) are simply taken to be ones.

The observant reader will note the similarity between our method and kernel PCA methods [SGS05, SSM98, WCS05]. In kernel methods, each point in the data is mapped into a high-dimensional space, called the feature space. By choosing the mapping correctly, various relations might be found within the data in the feature space that would be hard to find otherwise. For example, PCA can be used on the feature space to gather information about linear relations in the high-dimensional space, which are non-linear relations in the ambient data space. Our approach can be viewed as mapping the data into a 'feature space', where the features are some form of closeness ('affinity') to each of the given data points. There is also a connection between our approach and Support Vector Machines (SVM) [SC08], where linear classifiers are sought in a high-dimensional feature space.

There is also an alternative way to derive the MaD function as the weighted average of relevant positive functions, where larger weight is given to those functions obtaining small values on the data x_i. See the Appendix for details.

5. The Reconstruction Algorithm

Having covered the basic theory behind our method, we are now in a position to describe our reconstruction algorithm. It consists of two main steps: First, the MaD is sampled on a hierarchical grid covering the volume of interest. Secondly, the local minimum surface of the function is extracted. The first operation is quite straightforward, albeit having relatively high complexity. The latter operation is not as straightforward, and requires some sophistication. In the sequel, we describe the details of each of these steps.

5.1. MaD calculation

The algorithm involves first choosing the number of centres, M, and the dimension of the approximate nullspace, l. Then the $M \times M$ covariance matrix Σ, as defined in (3), is constructed. This costs $O(M^2N)$, where N is the size of the input data set. Next, the l eigenvectors of Σ with smallest eigenvalues must be computed. This costs $O(lM^2)$. Computing the MaD over the sample grid involves evaluating (5) at each grid point. The complexity of this is $O(lM^2G)$, where G is the number of grid points. A typical number for G would be $512^3 = 1.4 \times 10^8$. However, it is impractical and quite unnecessary to sample the entire grid at full resolution. Because high accuracy is required only near the surface, we employed the following refinement technique. Initially, the MaD is sampled on a coarse square grid, forming a voxel space. Voxels whose centres have function values which fall under a certain threshold are then further sampled on a finer grid. This can be continued recursively on an octree-type structure. Moreover, the sampling at the higher levels may make do with just a few eigenvectors [a small value of k in (5)]. Extra eigenvectors can be added while refining the grid, making the computation even more efficient.

5.2. The watershed transform

The most difficult part of the algorithm is extracting the surface. In 2D, we can imagine the MaD as an elevation map of mountains and ridges. The curve we seek is a collection of points for which the MaD is small, so must lie in the ridges of the map. As a start, we would like detect these ridges in the map. In image processing, this is done using a technique known as the *watershed transform*. It is used to segment an image by intensity to different valleys. Intuitively, if a drop of water falls on a mountain it will slide down away from the *watershed*, or drainage divide, and not cross any other such line. If we apply the watershed transform to the negative of the MaD, the mountains become valleys and watersheds become ridges, so the watershed transform of the negative of the MaD will reveal the set of ridges. We outline here the mathematical definitions, and then adapt to the 3D case. See [RM00] for a more detailed treatment. Let f be an image. Define the *topographical distance* between two points p and q in the image as:

$$T_f(p, q) = \inf_{\gamma} \int_{\gamma} ||\nabla f(\gamma(s))|| \, \mathrm{d}s,$$

where the infimum is over all paths γ from p to q. Now let m_i be a minimum of f. Then the catchment basin of m_i, $CB(m_i)$, or *watershed segment*, is the set of all points, which are topographically closer to m_i plus the absolute height of

m_i than to any other minimum of f

$$CB(m_i) = \{x | \forall j \neq i,\ f(m_i) + T_f(x, m_i) < f(m_j) + T_f(x, m_j)\}.$$

The watershed lines are defined as the boundaries of all watershed segments. We will call the 3D equivalent of a watershed line a *watershed surface*.

If we take the negative of the MaD on the region of interest, then the area or volume bounded by the underlying curve or surface is the union of some watershed segments. The main question is which segments exactly. We elaborate on this next.

5.3. Identifying the segments

We distinguish between two types of segments according to their relationship to the underlying surface: interior segments, which reside within the surface, and exterior segments, which reside outside the surface. We seek this classification of the segments.

Identifying the interior segments is not a trivial task. We use the following heuristic: We maintain a score for the segments. The segments touching the boundary of the region of interest are the *hard* exterior, and assigned a score of zero. Positive score will mean the segment is an interior segment, while negative score will mean exterior segment. We search next for points that we know, with high degree of certainty, lie between exactly two segments. This is done by examining a sphere with a small radius around the point and counting the segments that intersect that sphere. After finding these points we iterate over them, and examine their neighbouring segments. If one of the segments is an exterior segment we increment the score of the other segment by one, and if one of them is an interior segment we lower the score of the other by one. Hard exterior segments do not have a score and will always be exterior segments.

This process is illustrated in Figure 2. We start by classifying just the hard exterior segments (light blue). Points 1 and 2 then raise the score for some of the interior segments (green), but point 3 lowers the score for an interior segment, erroneously classifying it as exterior (yellow). However, points 4, 5, 6 later raise the score of this segment enough to become positive, correcting the classification error. Finally, points 7 and 8 correctly lower the score of two exterior segments.

After several iterations, all of the interior segments touching the surface will have been identified, while the other interior segments will remain unclassified. The outer boundary of the known interior segments is output as the reconstruction. Although this approach is somewhat heuristic, it appears to give satisfactory results in many cases.

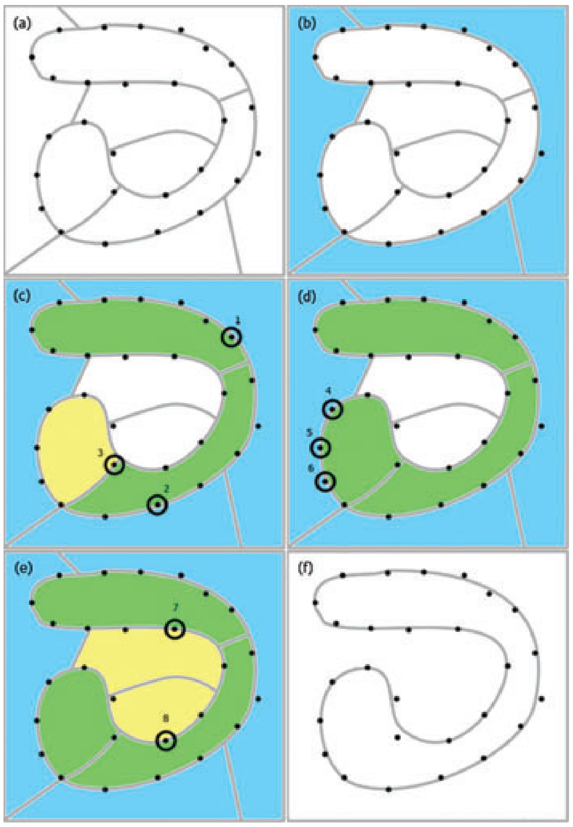

Figure 2: *An example of the steps in identifying the interior segments. Unidentified segments are shown in white, hard exterior in light blue, positive score in green and negative in yellow.*

6. Experimental Results

In the following section, the RBF method will refer to the idea described in Section 3, that is using only the first eigenvector of the matrix A as the coefficient vector for the interpolant.

6.1. Implementation

Our software, implemented in MATLAB, first computes the covariance matrix Σ, using one centre per data point (unless specified otherwise). As we shall show later, we need only a few vectors from the approximate nullspace of Σ, which we computed using a variant of the 'inverse iteration' method described by Toledo and Gotsman [TG08]. Next, we evaluate the MaD on a voxel grid of low resolution represented by a multi-dimensional array. Each member of the array is then duplicated four or eight times (depending on the dimension) resulting in a grid with twice the resolution. The MaD is then recalculated for voxels on the finer grid, which have values below a certain threshold. The process is then repeated until the desired resolution is reached. We usually choose the threshold as the lowest quartile for the 2D case and lowest

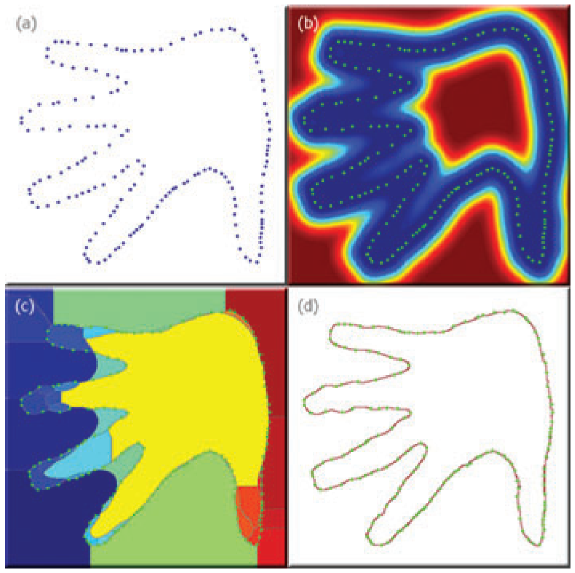

Figure 3: *Reconstructing the 'hand' curve from a 2D data set of 162 points: (a) The input point set. (b) The colour-coded MaD where blue and red colours indicate lower and higher values, respectively. (c) Segmentation of the watershed transform. (d) The reconstructed curve as the boundary of the interior segments.*

octile for the 3D case of the recalculated values. This makes the number of recalculated values for each level roughly the same. However, lower values could also be safely used, with shorter running times.

Once the MaD is computed, the program uses the watershed function from MATLAB's image processing toolbox to obtain the watershed transform of the MaD on the grid. An example of the output for the hand data set is shown in Figure 3(c). The program then creates a new grid where voxels inside an interior segment will have the value 1 and voxels in an exterior segment will have the value 0. This grid acts as a sampled indicator function for the object.

The surface can now be extracted as the 1-isosurface of the indicator function. This, however, will result in a very jagged surface. Faced with the same problem, Hornung and Kobbelt [HK06] applied Laplacian smoothing to the resulting mesh. We chose instead to smooth the *indicator function* with a simple, small averaging filter and extract the 0.5-isosurface. This seems to produce better results and also does not shrink the mesh.

6.2. MaD versus RBF

To better illustrate the different steps of the reconstruction algorithm, we first apply it to the reconstruction of curves

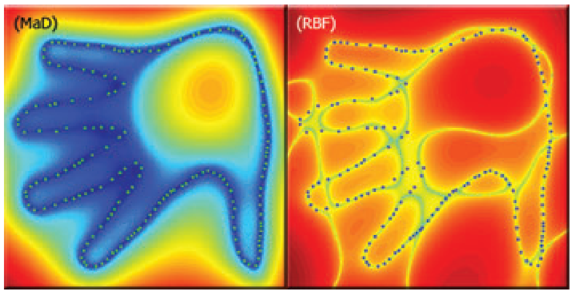

Figure 4: *Comparison of (left) MaD and (right) absolute value of RBF on hand dataset using very wide Gaussians centred at all data points, with standard deviation containing all the points.*

from 2D data sets. Figure 3 shows the three steps of the algorithm for the *hand* data set, containing 162 data points. Using radial Gaussians centred at all the data points, we were able to obtain a perfect reconstruction.

In this particular example, we used 'narrow' Gaussians which makes the reconstruction more local, and the first eigenvector very dominant. Thus, if only the absolute values of RBF are used, then MaD and RBF will produce very similar results. However, as MaD can use any number of eigenvectors, it is less sensitive to the width of the Gaussians. Figure 4 compares between the MaD and RBF unsigned distance functions using the same centres as in the previous example, but with wider Gaussians. A good heuristic for Gaussian width (standard deviation) is five to six times the average distance between a point and its nearest neighbour.

6.3. Manifold dimension

Our approach provides a significant advantage over *signed-distance* approaches when the dimension of the underlying manifold surface is not standard. For example, the zero set of the RBF distance function will always have co-dimension 1. This means that even if the data set is sampled from a curve in 3D, the RBF method will always result in a *surface* containing that curve. In contrast, the MaD method can adapt to this situation by setting $l > 1$. Figure 5 shows the 'helix' data set with a 2D slice of the MaD and absolute value of RBF. As evident in the figure, RBF can produce only a 2D manifold as its zero set, while MaD has small values only very close to the helix. The surface in this case was obtained by extracting a 'very close to zero'-isosurface because there is no easy way to extract one-dimensional data. Using the watershed transform will produce the same result for both cases. This data set has $n = 100$ points and we used all possible eigenvectors for MaD ($l = 100$).

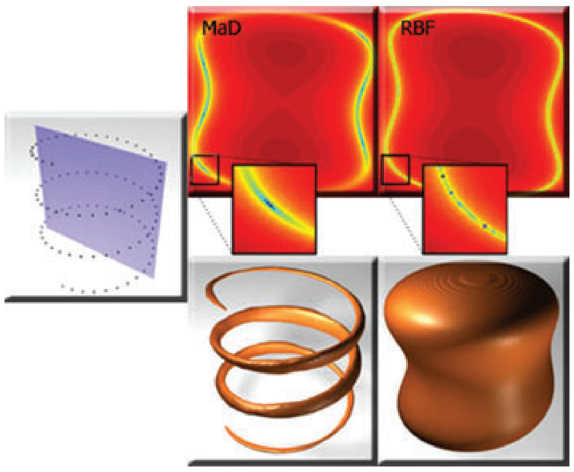

Figure 5: *Reconstructing the 'helix'. (Left) the input data set. (Top) Coded-coded MaD (left) and RBF (right) distance functions in the cross section plane shown in purple in the top image. Note that the MaD has significant minima only close to the input data points. (Bottom) Surface extracted from the MaD (left) and the zero set of the RBF (right).*

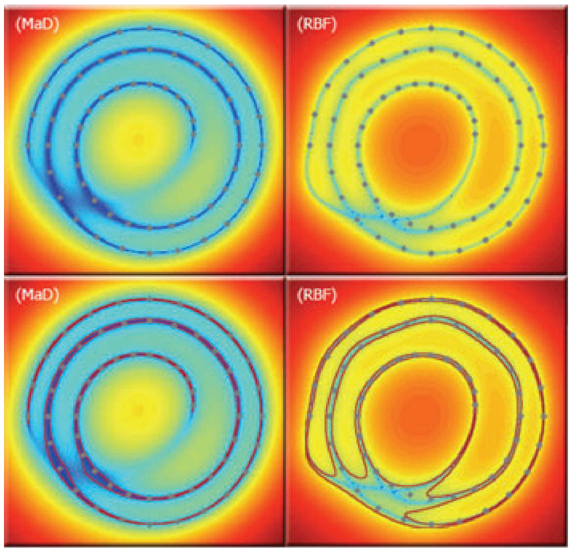

Figure 6: *Distance measure from the open 'spiral' using MaD (left) and RBF (right). A 'close to zero'-isoline is marked in red.*

6.4. Manifold topology

Because, in the RBF method, the surface is extracted as the zero set of a signed distance function, it must be a closed manifold, that is have no boundaries. The MaD method is not limited to closed surfaces. A demonstration of the distance from a 2D spiral data set is shown in Figure 6. For

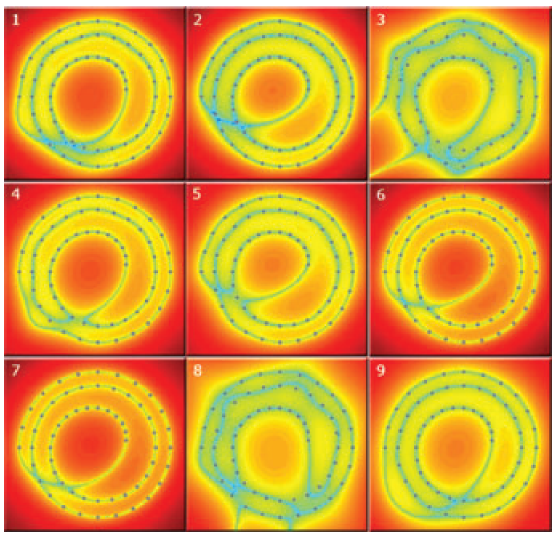

Figure 7: *The distance map produced by each of the first nine eigenvectors of Σ, from top to bottom, left to right. Notice how the spiral closes differently in some cases.*

MaD, the values close to zero (shown in blue colours) follow the spiral, in contrast to RBF, where the result is a closed curve. To understand this better, we plot in Figure 7 the first few eigenvectors of Σ independently. It seems that in each of these plots the spiral is closed in some different way. However, when summed to form the distance function, they cancel out to produce a clean result.

6.5. The optimal dimension of V

The dimension of V is determined by M—the number of centres (which determines the size of the matrix Σ) and l—the dimension of the approximate nullspace (the number of smallest eigenvalues of Σ) used

$$\dim\ V = M - l$$

so aiming at a target dimension, we must decide on M and l. On one hand, a small M will result in a smaller Σ, which can significantly reduce runtime. On the other hand, the mapped points will not be guaranteed to reside on a hyperplane, which is our basic assumption. Larger values of M causes the mapping in $\mathbb{R}^M$ to be 'flatter' to a degree that when M is equal to the number of points n, the mapping lies on a single hyperplane. Because the quality of the reconstruction is more important than the runtime, we typically prefer a larger value of M and, consequently, also a larger value of l.

We have noticed that in some cases the spectrum of Σ has a noticeable 'bend', as evident in Figure 8. We presume that the eigenvectors corresponding to the eigenvalues beyond the bend are in fact a basis of the true 'nullspace' of Σ (the

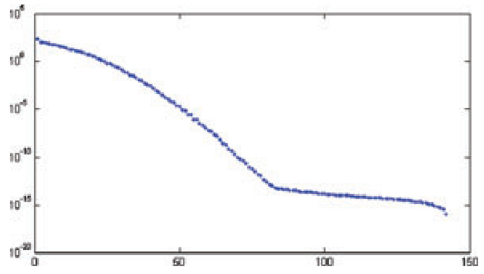

Figure 8: *The spectrum of Σ for the data set in Fig. 9 (on a logarithmic axis). Note the 'bend' around the 80th eigenvalue, meaning that the remaining $l = 60$ eigenvectors having smallest eigenvalues form a basis for $V^{\perp}$.*

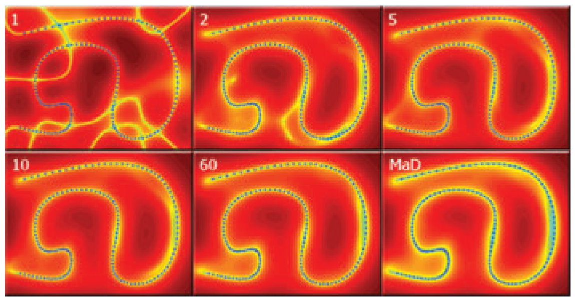

Figure 9: *The MaD distance using an unweighted sum of the first eigenvectors. From left to right on the first row are the maps using 1, 2, and 5 and on the second row are the maps using $l = 10$ and $l = 60$ eigenvectors which, according to Fig. 8 form a basis for $\mathbf{V}^{\perp}$. The rightmost map is the weighted MaD.*

linear space $V^{\perp}$ mentioned in Section 3). Hence, there is no point in weighting these eigenvectors and the calculation can be made easier. In fact, usually only a few eigenvectors are needed to produce a satisfying result, as demonstrated in Figure 9.

6.6. 3D examples

Figure 10 shows the reconstruction of the 3D *skull* data set, containing approximately 10 000 points. It is not closed and is sampled non-uniformly. For the MaD reconstruction we used the first unweighted 100 eigenvectors, a number which was chosen somewhat arbitrarily, and the result does not critically depend on this number. We compared the MaD reconstruction to simple RBF, and to a number of state-of-the-art reconstruction methods, including RIMLS [OGG09], IPSS [GG07, OGG09] and Poisson reconstruction [KBH06]. The implementations used for the latter three were those included in the MeshLab [ML] software package. Despite the fact that these methods require, and use, normal data in their algorithms, and MaD does not, so the comparison is

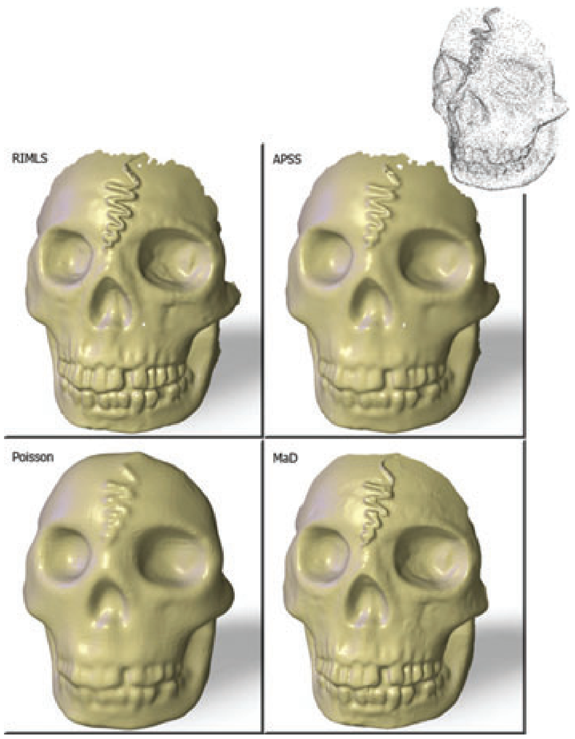

Figure 10: *Reconstruction of the 'skull' data set using different algorithms.*

somewhat biased against MaD, Figure 10 shows that we were still able to achieve results comparable, or even superior to the competition. The RBF result was so bad that there was no point in showing it. Figure 11 shows a closeup on the forehead area. None of the methods is perfect, but MaD seems to have achieved, all in all, the best reconstruction.

Another example is shown in Figure 12, where we reconstructed the *screwdriver* from a point cloud containing 2700 *unoriented* points (a random 10% sample from an original 27 000 points). For RIMLS and Poisson we had to estimate normals by local PCA. The results of both contained artefacts for every set of parameters we chose. Although there might be a combination of parameters that generates a good result, our MaD reconstruction gave perfect results with practically no tuning. Note that the missing tip of the screwdriver is also not present in the input point cloud.

6.7. Performance

We used the method of Toledo and Gotsman [TG08] to compute the l vectors in the approximate nullspace of Σ. For example, the computation of $l = 100$ eigenvectors for the skull data set of Figure 10, which contains about 10^4 points, requires 390 s in MATLAB on a 2.4 GHz machine with 4 GB

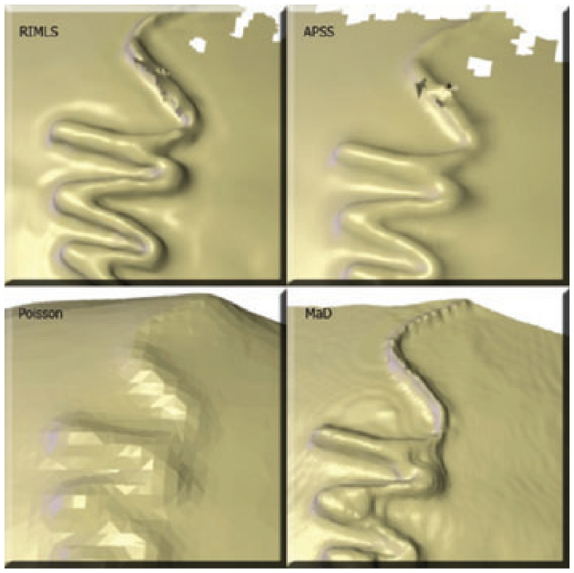

Figure 11: *Closeup on the forehead region of the 'skull' model. Both RIMLS and APSS have issues and Poisson reconstruction loses features. MaD reconstruction seems to be best.*

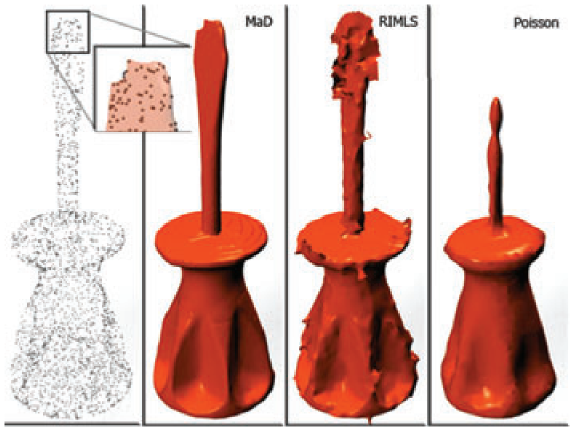

Figure 12: *Reconstruction of the 'screwdriver' data set. The close-up shows the tip of the screwdriver with the MaD reconstruction overlaid.*

RAM. Computation of the MaD took 270 s for each level of resolution with octile threshold (i.e. the lower eighth of the range), starting with a resolution of 100^3 and refining twice to reach a resolution of 400^3. Although the total runtime of 20 min is not very fast, it should be noted that our implementation was not optimized. Furthermore, this part can be easily rewritten to run on the GPU. The next steps (watershed transform, etc.) require only a few seconds.

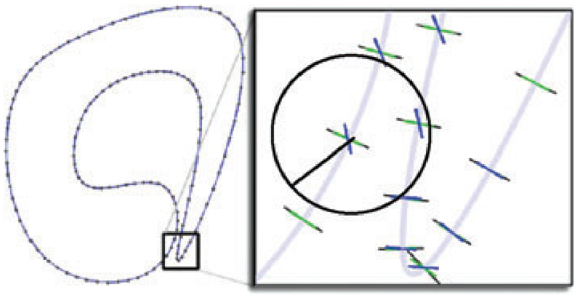

Figure 13: *Normal estimation using PCA and MaD. The green lines show the MaD estimate and blue lines the PCA estimate. The black line is the true normal. The circle shows the region of influence used for the PCA procedure.*

6.8. Normal estimation

Estimating surface properties, such as normal vectors, from a point cloud, is an interesting and important application in its own right. This can be done locally, using techniques such as PCA on a neighbourhood of points, without actually reconstructing the entire surface, or from the surface after reconstruction. We now show that it is possible to obtain a robust estimate for a surface normal also from our MaD distance function. Because the MaD takes into account global properties of the data as well as local properties, it is especially effective in regions where different parts of the surface come close to each other, where a local method may easily get confused. MaD seems to be able to distinguish between the different parts of the surface. A similar observation was made by Luo *et al.* [LSW09] using their diffusion metric for estimating gradients. The normal direction is the principal direction of the Hessian of the MaD function [DoC76]. We compare this normal estimation procedure with simple PCA estimation: for each point we compute the direction of minimal spread of all the points contained in a circle cantered at the point. Although this is not an optimal estimate, its weaknesses are a characteristic of all local approaches. The comparison can be seen in Figure 13. For the most part, the PCA and MaD estimates agree with the true normals. However, in the (zoomed in) area where the curve almost intersects itself, the PCA estimation is completely off.

We compare normal direction computed with MaD and k-nearest-neighbour PCA on the dancer sets of Figure 15. The error relative to the normals obtained from the original triangle mesh are shown in Table 1.

6.9. Noise and outliers

The interpolation properties of our method can be controlled by the number and position of centres used. When all the points of a data set are used as centres (which is the default), the mapped points will all lie exactly on a

Table 1: *Normal estimate error of two algorithms on the 'dancer' data set.*

No. of points	5000	2500	1250	625
MaD	0.0259	0.0421	0.0428	0.0716
PCA	0.0490	0.0771	0.1236	0.1827

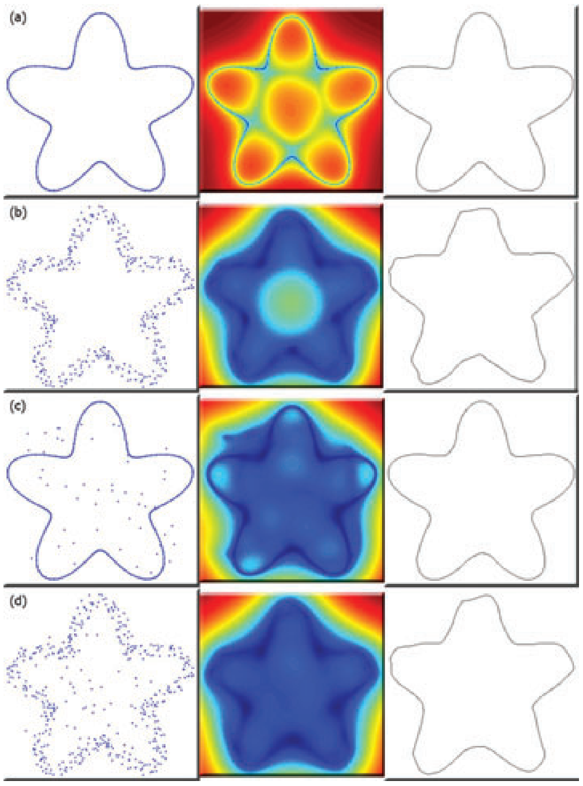

Figure 14: *Reconstruction with noise and outliers. The left image in each row shows the data set. The centre image shows the MaD and the right image shows the reconstructed curve. (a) Clean dense data set, (b) Noisy data set, (c) dense data set with outliers and (d) noisy data set with outliers.*

($(n-1)$-dimensional hyperplane (represented by Σ's first eigenvector). Hence, the extracted surface will most likely pass through them, and therefore each point will lie on some watershed surface. The resulting surface will interpolate the points, and in the presence of noise this is undesirable. In these cases, we randomly choose a proper subset of the points as centres and proceed as usual. Our experiments show that selecting half of the points as centres will provide satisfying results for most cases.

Outliers will have a different impact on the MaD. Once points are mapped to the high-dimensional space there is no way to distinguish between outliers and other points, since the MaD of both will be close to zero. Therefore, in theory, any treatment of outliers should be done before using our method. However, in practice, outliers will cause additional watershed surfaces to appear, but will not change the watershed surfaces which should be part of the final surface. Selecting the correct interior segments will then eliminate these extra surfaces, resulting in a correct clean surface. So in fact, no additional handling is needed when our method is confronted by outliers. Figure 14 shows some 2D examples with various amount of noise and outliers.

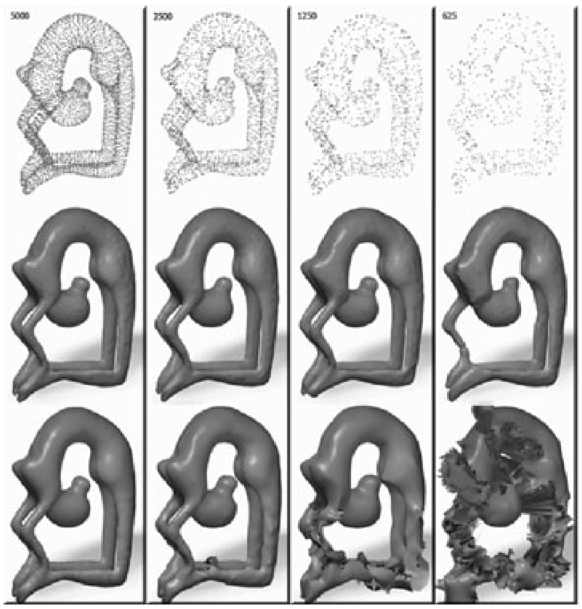

Figure 15: *Reconstruction with different densities. (Top) Input points, (middle) reconstruction using MaD and (bottom) reconstruction using RIMLS.*

6.10. Sparse and irregular data sets

Combining the power of the extra eigenvectors and the selection of segments allows our method to reconstruct surfaces also from quite sparse data sets. We demonstrate this with the *dancer* model in Figure 15, starting with 5000 points and halving the number of points until reconstruction fails. As evident in the figure, the number of points can be reduced to 1250 without much damage to the reconstruction. At 625 points, the selection step will not select part of the hand and the reconstruction fails. However, we can still use MaD to estimate normals for other algorithms which will better utilize them.

As mentioned in Section 6.2, the MaD is much less sensitive to the choice of Gaussian widths. This allows for an irregularly sampled data set to be reconstructed without the need to consider local density of points. We show an extreme

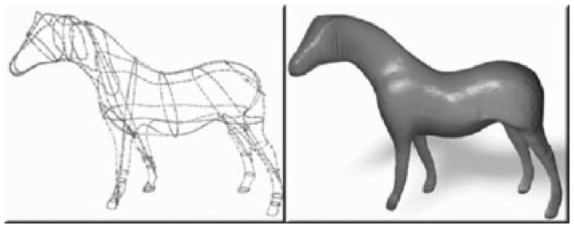

Figure 16: *MaD reconstruction of an irregularly sampled data set.*

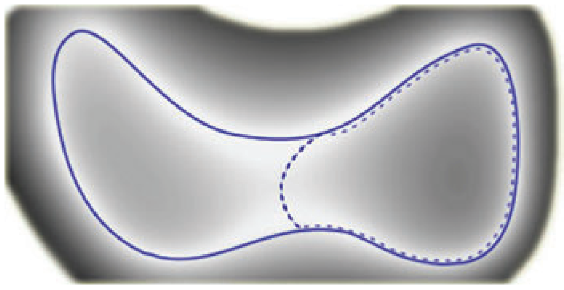

Figure 17: *The shortcut problem with min-cut. Regions coloured in a brighter grey intensity represent lower values of the MaD. The continuous blue line represents the desired cut, while the dotted line shows the cut that might result.*

case in Figure 16. The input data in this experiment was obtained by first taking planar slices of the horse model and then densely sampling the slices' boundaries.

6.11. Watershed versus Min-Cut

An alternative method of extracting a surface from an unsigned distance function was proposed by Hornung and Kobbelt [HK06]. This is based on the minimal cut of a regular grid graph using a discrete distance function and may be adapted to use our MaD distance function instead. The main advantage of the min-cut method over the watershed method is that segment selection, a difficult phase in the watershed method, is avoided. However, in certain situations, especially where the object is 'thin', the cut may pass through the thin part and skip part of the point set entirely. This is illustrated in Figure 17, where the continuous line represents the desired curve, and the dashed line represents the cut that the min-cut algorithm might generate. In contrast, the watershed transform of the map will result in two interior segments, which will be easily classified as such in the next phase of the algorithm.

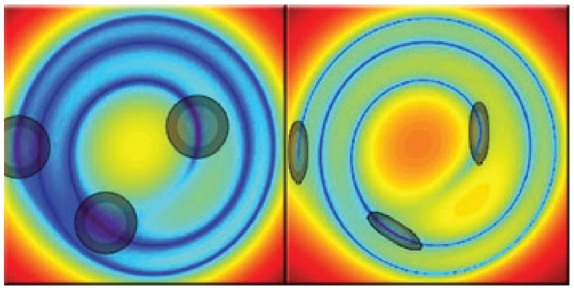

Figure 18: *Reconstruction of the 'helix' data set (left panel) without normal information and (right panel) with normal information. The Gaussians (coloured in grey) used in the latter are squeezed by 66% in the direction of the normal relative to the Gaussians used in the former.*

6.12. Using normal information

Our MaD method does not require normal information, and does not even estimate any normal information as an indirect part of the process. However, when reliable normal information is available, we would like to be able to put it to use. A simple way would be to orient the Gaussian basis functions such that they are aligned with the complementary tangent directions. This hints to the MaD that, locally, distances are less important in the tangent directions. An example of the difference this makes is shown in Figure 18, where it is clear that the additional normal information contributes to a sharper MaD function. However, we have seen cases where normal information could have the adverse effect on noisy and sparse sets, as points which are close to each other on a surface might end up far away from each other when mapped to $\mathbb{R}^M$. So care must be taken in those cases and sometimes it is better to ignore the normals.

7. Conclusion

We have presented a method to compute a distance function derived from a given point data set, based on embedding the data in a higher-dimensional space. In this space, the point set, and the underlying surface it is sampled from, may be modelled as a linear subspace, making all subsequent processing very easy and intuitive. The resulting MaD distance function is just a Mahalanobis (weighted) distance from this subspace. The more conventional distance function resulting from approximating the surface as a zero-set of a single linear combination of RBF functions, is shown to be a very special case of the MaD distance.

Embedding data in a higher-dimensional space using so-called *kernels*, to enable easier processing, is a common technique in machine learning. It remains to be seen whether other machine-learning techniques may be borrowed and applied to geometry processing.

The MaD method has proven to be quite versatile and robust, justifying the extra computational overhead compared to more traditional methods such as RBF. However, a number of issues require further research. First and foremost, the surface extraction procedure is still somewhat cumbersome, and could benefit from some simplification. Secondly, the computational complexity of the method should be reduced in order for it to be more practical. We are confident this can be done with some further optimization. Perhaps this can be combined with the surface extraction procedure.

Appendix: A Different Interpretation of the MAD Function

We present here an alternative derivation of the MAD function.

Given a set of data points $\{x_i\}_{i=1}^N$ and basis functions $\{\phi_j\}_{j=1}^M$, we seek a positive distance function related to the subspace spanned by these basis functions (denoted by Φ), ideally having a very small absolute value on the data points. The following function seems to have these properties:

$$D = \frac{\int_\phi f^2 \exp(-\sum_{i=1}^N f^2(x_i))\mathrm{d}f}{\int_\phi \exp(-\sum_{i=1}^N f^2(x_i))\mathrm{d}f}.$$

This is because D is the weighted average of all functions, which are the square of a function in Φ, such that the weight exponentially decreases in the average value of the function on the data points.

Now we show that D is precisely our MaD distance function by integrating over Φ through its coefficient space

$$\begin{aligned} D &= c\int_\phi f^2 \exp\left(-\sum_{i=1}^N f^2(x_i)\right) df \\ &= c\int_{\mathbb{R}^n} \left(\sum_{j=1}^M a_j\phi_j\right)^2 \exp\left(-\sum_{i=1}^N \left(\sum_{j=1}^M a_j\phi_j(x_i)\right)^2\right) da \\ &= c\int_{\mathbb{R}^n} (a^{\mathrm{T}}\phi)^2 \exp(-a^{\mathrm{T}}\Sigma a)\, da \\ &= \phi^{\mathrm{T}}\Sigma^{-1}\phi, \end{aligned}$$

where $c^{-1} = \int_\phi \exp(-\sum_{i=1}^N f^2(x_i))\,\mathrm{d}f$ and the last equality is by application of the standard Gaussian integral for vector a and matrix Σ

$$\int_{\mathbb{R}^N} a_i a_j \exp\left(-a^\top\Sigma a\right)\mathrm{d}a = c^{-1}\Sigma_{ij}^{-1}.$$

References

[AB98] Amenta N., Bern M. W.: Surface reconstruction by voronoi filtering. In *Proceedings of the Symposium on Computational Geometry* (Minneapolis, MN, USA, 1998), pp. 39–48.

[ACK01] Amenta N., Choi S., Kolluri R. K.: The power crust, unions of balls, and the medial axis transform. *Computational Geometry* 19, 2–3 (2001), pp. 127–153.

[ACSTD07] Alliez P., Cohen-Steiner D., Tong Y., Desbrun M.: Voronoi-based variational reconstruction of unoriented point sets. In *Proceedings of the Symposium on Geometry Processing* (Barcelona, Spain, 2007), Eurographics Association, pp. 39–48.

[Boi84] Boissonnat J.-D.: Geometric structures for three-dimensional shape representation. *ACM Transaction on Graphics 3*, 4 (1984), 266–286.

[CBC7*01] Carr C., Beatson R. K., Cherrie J. B., Mitchell T. J., Fright W. R., Mccallum B. C., Evans T. R.: Reconstruction and representation of 3D objects with radial basis functions. In *Proceedings of the SIGGRAPH'01* (Los Angeles, CA, USA, 2001), ACM, pp. 67–76.

[DEJ00] De Maesschalck R., Jouan-Rimbaud D., Massart DL. The Mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems 50*, 1 (2000), 1–18.

[DoC76] Do Carmo M. P.: *Differential Geometry of Curves and Surfaces*. Prentice-Hall, Upper Saddle River, NJ, 1976, Chap. 3

[DTS02] Dinh H. Q., Turk G., Slabaugh G.: Reconstructing surfaces by volumetric regularization using radial basis functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24* 10 (2002), 1358–1371.

[GG07] Guennebaud G., Gross M.: Algebraic point set surfaces. In *Proceedings of the SIGGRAPH'07* (San Diego, CA, USA, 2007), ACM, p. 23.

[GGG08] Guennebaud G., Germann M., Gross M.: Dynamic sampling and rendering of algebraic point set surfaces. *Computer Graphics Forum 27*, 2 (2008), 653–662.

[GVL96] Golub G. H., Van Loan C. F.: *Matrix Computations* (3rd edition). Johns Hopkins University Press, Baltimore, MD, 1996.

[HDD*92] Hoppe H., Derose T., Duchamp T., Mcdonald J., Stuetzle W.: Surface reconstruction from unorganized points. In *Proceedings of the SIGGRAPH'92* (Chicago, IL, USA, 1992), ACM, pp. 71–78.

[HK06] Hornung A., kobbelt L.: Robust reconstruction of watertight 3D models from non-uniformly sampled point clouds without normal information, In *Proceedings of the Symposium on Geometry Processing* (Cagliari, Sardinia, Italy, 2006), Eurographics Association, pp. 41–50.

[KBH06] Kazhdan M., Bolitho M., Hoppe H.: Poisson surface reconstruction. In *Proceedings of the Symposium on Geometry Processing* (Cagliari, Sardinia, Italy, 2006), pp. 61–70.

[KSO04] Kolluri R., Shewchuk J. R., O'Brien J. F.: Spectral surface reconstruction from noisy point clouds. In *Proceedings of the SIGGRAPH* (Nice, France, 2004), ACM, pp. 11–21.

[LSW09] Luo C., Safa I., Wang Y.: Approximating gradients for meshes and point clouds via diffusion metric. *Computer Graphics Forum*, *28*, 5 (2009), pp. 1497–1508.

[ML] MeshLab. http://meshlab.sourceforge.net/

[OGG09] Oztireli A. C., Guennebaud G., Gross M.: Feature preserving point set surfaces based on nonlinear kernel regression. *Computer Graphics Forum 28*, 2 (2009), 493–501.

[RM00] Roerdink J., Meijster A.: The watershed transform: Definitions, algorithms and parallelization strategies. *Fundamenta Informaticae 41*, 1–2 (2000), 187–228.

[SAAY06] Samozino M., Alexa M., Alliez P., Yvinec M.: Reconstruction with Voronoi-centered radial basis functions. In *Proceeding of the Symposium on Geometry Processing* (Cagliari, Sardinia, Italy, 2006), Eurographics Association, pp. 51–60.

[SGS05] Schölkopf B., Giesen J., Spalinger S.: Kernel methods for implicit surface modeling. In *Advances in Neural Information Processing Systems 17*. L. K. Saul, Y. Weiss, L. Bottou (Eds.). MIT Press, Cambridge, MA (2005), pp. 1193–1200.

[SSM98] Schölkopf B., Smola A., Müller K.: Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation 10*, 5 (1998), 1299–1319.

[TG08] Toledo S., Gotsman C.: On the computation of the nullspace of sparse rectangular matrices. *SIAM Journal of Matrix Analysis and Applications*, *30*, 2 (2008), 445–463.

[SC08] Steinwart I., Christmann A.: *Support Vector Machines*. Springer-Verlag, New York, 2008.

[WCS05] Walder C., Chapelle O., Schölkopf B.: Implicit surface modelling as an eigenvalue problem. In *Proceedings of the Machine Learning* (Bonn, Germany, 2005), ACM, pp. 936–939.