View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Washington University St. Louis: Open Scholarship

Washington University in St. Louis

Washington University Open Scholarship

All Computer Science and Engineering Research

Computer Science and Engineering

Report Number: WUCSE-2008-15

2008-01-01

Local Neighborhoods for Shape Classification and Normal Estimation

Cindy Grimm and William Smart

We introduce the concept of local neighborhoods, a generalization of the one-ring on a mesh to unlabeled 3D data points arising from sampling a 2D surface embedded in 3D. The local neighborhood supports both local shape classification and robust normal estimation. In particular, local neighborhoods out-perform traditional approaches in unevenly sampled, curved regions. We show that the local neighborhood can be used in place of a full mesh structure for applications such as smoothing, moving least-squares reconstruction, and parameterization. Longer version of paper submitted to CAGD

Follow this and additional works at: https://openscholarship.wustl.edu/cse_research

 Part of the Computer Engineering Commons, and the Computer Sciences Commons

Recommended Citation

Grimm, Cindy and Smart, William, "Local Neighborhoods for Shape Classification and Normal Estimation" Report Number: WUCSE-2008-15 (2008). *All Computer Science and Engineering Research.*
https://openscholarship.wustl.edu/cse_research/226

Department of Computer Science & Engineering - Washington University in St. Louis
Campus Box 1045 - St. Louis, MO - 63130 - ph: (314) 935-6160.

Department of Computer Science & Engineering

Washington University in St. Louis
School of Engineering & Applied Science

2008-15

Local Neighborhoods for Shape Classification and Normal Estimation

Authors: Cindy Grimm and William Smart

Corresponding Author: cmg@wustl.edu

Abstract: We introduce the concept of local neighborhoods, a generalization of the one-ring on a mesh to unlabeled 3D data points arising from sampling a 2D surface embedded in 3D. The local neighborhood supports both local shape classification and robust normal estimation. In particular, local neighborhoods out-perform traditional approaches in unevenly sampled, curved regions. We show that the local neighborhood can be used in place of a full mesh structure for applications such as smoothing, moving least-squares reconstruction, and parameterization.

Notes:
Longer version of paper submitted to CAGD

Type of Report: Other

Department of Computer Science & Engineering - Washington University in St. Louis
Campus Box 1045 - St. Louis, MO - 63130 - ph: (314) 935-6160

Local Neighborhoods for Shape Classification and Normal Estimation

Cindy Grimm and William D. Smart

Washington Univ. in St. Louis

Abstract

We introduce the concept of local neighborhoods, a generalization of the one-ring on a mesh to unlabeled 3D data points arising from sampling a 2D surface embedded in 3D. The local neighborhood supports both local shape classification and robust normal estimation. In particular, local neighborhoods out-perform traditional approaches in unevenly sampled, curved regions. We show that the local neighborhood can be used in place of a full mesh structure for applications such as smoothing, moving least-squares reconstruction, and parameterization.

Key words: Surface reconstruction, normal estimation, shape classification, unlabeled data, exponential map, one-ring

1 Introduction

We introduce a formal definition of *local neighborhoods* (LNs) for 3D point data sets. Informally, a local neighborhood (LN) is the equivalent of the one-ring neighborhood of a vertex in a mesh. By extending the one-ring concept to point data sets we can apply many mesh-based algorithms, such as smoothing (Taubin, 1995; Jones et al., 2003), edge sharpening (Attene et al., 2005), parameterization (Saba et al., 2005), and advanced feature detection Lai et al. (2007) directly to the point data set without the need for building an intermediate mesh surface. The graph structure induced by the LNs can also be used to infer the global structure of the data set. LNs differ from simply using the k-nearest neighbors in that the neighborhood is both ordered and, typically, form a subset of the k-nearest neighbor graph.

LNs also provide a very robust method for estimating surface normals, even in cases where *any* k-nearest neighbor, SVD-based algorithm (Mitra and Nguyen, 2003) will fail catastrophically due to uneven sampling densities (see figure 1). Moreover, we show that LNs can recover the correct normals for sharp features such as edges and corners.

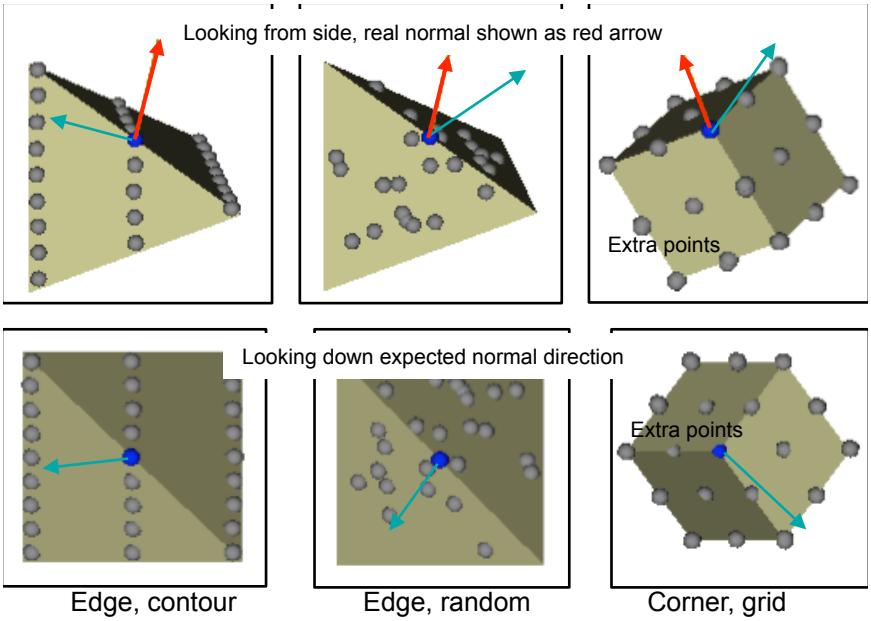

Fig. 1. Three cases where the SVD results in the incorrect normal. For the contour edge case, the density of samples in one direction (which lies diagonal to the edge) results in the fitted plane passing through those samples. In the random sample case, the uneven point distribution pulls the plane towards one face. For the cube this failure is due to the fact that one side of the cube has twice as many samples.

To ensure that a LN is the correct (local) reconstruction of the surface we simultaneously compute a local shape descriptor to verify the LN. This shape descriptor can in turn be used in feature-finding algorithms (Attene et al., 2005; Hildebrandt et al., 2005). The normal plus shape descriptor can also be used to improve surface reconstruction algorithms (Kolluri, 2005; Boissonnat and Cazals, 2000).

The majority of (local) normal estimation approaches rely on some form of surface fitting, which can be biased by uneven sampling. Similarly, discrete, curvature-based feature-finding approaches are also influenced by sampling regularity (Gatzke and Grimm, 2006). In contrast, our approach does not assume regular sampling but instead assumes that the samples arise from a 2D surface embedded in 3D. Locally, we can describe any such surface with a small number of possible shapes (flat, bowl, ridge, saddle, edge, or corner). Given an LN we explicitly build a model for each of these possible shapes and determine how well the samples fit each model. This greatly reduces the influence of the sampling on the model evaluation. The quality of the best model is then used to evaluate the LN. This approach provides additional robustness by insuring that the LN "predicts" a locally-valid 2D surface, and that the shape model accurately reflects the local connectivity. Without this local shape information we can easily end up finding LNs that look good from a triangulation stand-point, but do not conform to the underlying surface.

Although LNs and shape descriptors are conceptually simple, formalizing them requires a non-trivial local shape analysis. The properties that make for a "good"

LN are very similar to the ones that make for a good mesh triangulation (regularly-shaped triangles). The shape descriptors are based on a geometrical description of the idealized local shape; the difficulty is in balancing the numerical evaluation so that the different models can be directly compared.

We begin by defining, at a high level, both a formal set of criteria for defining a LN and sampling criteria for guaranteeing that a valid LN exists (section 3). We follow this with a (very slow) algorithm which is guaranteed to be optimal (section 7). Section 6 provides a much faster, heuristic algorithm for finding LNs. We give explicit equations for the LN criteria and our shape descriptors in sections 4 and 5. In section 8 we describe the test cases which we use to experimentally verify that our LN criteria and shape descriptors are robust to both irregular sampling and noise. We use the experimental data from the test cases, and show that our heuristic algorithm returns a LN which is close to the correct (non-heuristic) answer, with high probability. In section 9 we demonstrate the heuristic algorithm on a variety of real-world data sets and show its applicability to several existing applications.

2 Previous work

One approach to normal estimation is surface reconstruction, either local (Fleishman et al., 2005) or global (Bernardini et al., 1999; Amenta et al., 2001). In the case of noise-free and dense sampling, several global, Delaunay-based techniques exist for accurate normal and feature size approximations (Dey and Goswami, 2003; Boissonnat and Cazals, 2000; Amenta et al., 2001). In the presence of noise, however, these reconstructions can be incorrect. Recent work (Dey and Sun, 2005) extends this approach to noisy data by using an adaptive threshold to cull Delaunay balls that arise due to noise. We compare our local approach to this one and show that, particularly for unevenly sampled data, our normal reconstruction is more accurate (table 4). However, the global approach can be more accurate in cases where the sampling criteria for our approach is not met (section 8.3).

The most common approach to normal estimation is plane fitting. (Mitra and Nguyen, 2003) provide a formula for estimating the best value of K to use based on estimates of the noise and local curvature. They then calculate the normal by plane fitting and show that adaptively choosing K increases the accuracy of the normal estimation. This approach was extended to quadratic and cubic surfaces with normal-based weights (Vančo and Brunnett, 2007) which provide a better approximation in the presence of noise. We compare our approach to this one and show that, particularly for areas of high curvature and uneven sampling, the local neighborhood normal reconstruction outperforms these surface fitting approaches (section 8.2). This is because irregular sampling can easily "pull" the fitted surface away from the average, due to the nature of linear regression. This effect is worse when the underlying shape, such as a corner, can not be approximated by the fitted surface

(plane, quadric, or cubic).

Pauly et. al (Pauly et al., 2003) present a modification of the plane-fitting algorithm that weights points by their distance from the point of interest. This is called locally-weighted regression in the machine learning literature. While this can help in some cases, it actually exacerbates the contour-sampling problem by reducing the influence of the points on the nearby contour. In a recent comparison of these two plane-fitting techniques with a global, Delaunay-based one (Dey et al., 2005), the Delaunay and weighted sampling approaches were comparable, and out-performed the non-weighted, plane-fitting approach. This result is in line with our experiments.

Fleishman et. al. (Fleishman et al., 2005) provide a statistically robust method for locally classifying points around sharp features into clusters, each of which is well-modeled with a smooth polynomial surface. Sharp features are created where the polynomial surfaces intersect. This approach gives much more accurate normals along sharp features. Similarly, we also use intersecting planes to more robustly calculate normals at sharp features. Unlike Fleishman's approach, we do not apply smoothing before calculating the normal; this allows us to better capture small surface detail without precluding the subsequent use of smoothing if desired.

An alternative to local curvature-based feature finding is to use both the surface locations and the normals (Lai et al., 2007) and examine how the surface normals vary in a local patch on the surface. The techniques presented here could easily be applied to this approach to produce better normal estimation, produce a local parameterization when a mesh is not available, and identify edges and corners, which are processed differently.

Nearly all point-based methods define the concept of a neighborhood, typically the K closest points as measured by Euclidean distance. This approach can cause problems when the surface "folds back" on itself because nearby points in Euclidean space may not be close from a geodesic measure. This problem is exascerbated by large K. One solution to this is to use an approximation of geodesic distance to build large neighborhoods from small ones (Vančo and Brunnett, 2007). We have experimented with this approach but have found it to *decrease* the quality and accuracy of the normal reconstruction because a smaller neighborhood reduces the chance of "bridging" a gap in the sampling. Except for the case where the distance between the folded surface is less than the sampling distance (which causes the one-ring to bridge the gap), the addition of points on the nearby surface tends to increase the error of all of the models but does not change the set of available normals, which are determined by the one-ring.

Delaunay triangulation also defines a concept of local neighborhood, called the "natural neighbors" (Boissonnat and Cazals, 2000). Our local neighborhoods are usually an ordered subset of the natural neighbors, and are computed locally, as

opposed to requiring a global construction of the Delaunay triangulation.

As part of our analysis we build local approximations of the surface (plane, ridge, bowl, saddle, edge, corner) to determine if the K-nearest neighbor points could have come from that surface (section 5). An alternative to explicitly representing the surface is to determine if there exists a rigid motion entirely in the tangent space of the points (Gelfand and Guibas, 2004). This can be used to identify points that lie on a plane, cylinder, or sphere. This approach is unsuitable for initial shape estimation purposes because it only handles a subset of the possible surface types, requires surface normals, and a relatively large number of points. However, it could be a useful secondary processing step for determining normal consistency across larger surface patches.

3 Formal definitions

We assume that the sample points $\{D\}$ come from some smooth surface S, possibly with noise, and that the surface is sufficiently sampled (see below). Given a point $P \in \{D\}$, its valid local neighborhood is an ordered subset of n points $Q_1 \dots Q_n$, $Q_j \subset \{D\}$ such that the following hold:

- The local neighborhood $\{Q\}$ contains at least three points ($n \geq 3$).
- Connect each point Q_j to $Q_{j+1 \mod n}$ with a geodesic in S to form a closed curve c. The curve c does not intersect itself.
- The region R bounded by the curve c is a disk in S, i.e., R is homeomorphic to a unit circle (x,y), $x^2+y^2<1$ in the plane.
- P is contained in the region R.
- No points in $\{D\}$ are inside of the region R.

A good local neighborhood is one which is evenly sampled both in angle and length:

- Connect each point Q_j to P via geodesics g_j. The lengths of all of the g_j are the same.
- The angle between g_j and g_{j+1} is the same for all j.

The surface is considered to be sufficiently sampled if:

- The edges PQ_j and Q_jQ_{j+1}, are ε approximations to the corresponding geodesics. By ε approximation we mean that the maximum distance from any point on the geodesic to the closest point on the edge is less than ε.
- The region R projects onto the tangent plane at P without folding.
- Project the edges Q_jQ_{j+1} onto the tangent plane to form a polygon q. No other points that are within $\max_j ||Q_j - P||$ distance of P project into q. (The sampling

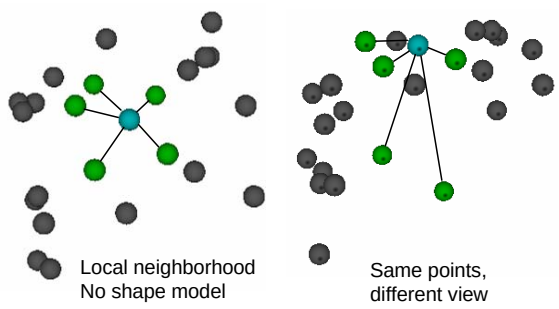

Fig. 2. A good local neighborhood which is a poor approximation to the local shape. (Point is on the side of a corner.)

on the surface is denser than the sampling between two separate parts of the surface.)

To extend these definitions to surfaces with boundary we relax the inside constraint for P when P is on the boundary. In this case the region R is a half disk with P lying on the half disk's boundary.

Note that a data set that satisfies the ε-sampling criterion (Dey and Goswami, 2004) will also satisfy our requirements. The local neighborhood is also related to the so-called "natural neighbors", or the neighbors that arise from a Delaunay triangulation (Boissonnat and Cazals, 2000). Note, though, that our construction is a purely local one whereas the Delaunay approach is global.

Suppose we have a set of samples on S that meet the above criteria. When constructing a local neighborhood (picking the $\{Q\}$) we can easily evaluate the quality by replacing the geodesics with their corresponding edge approximations (section 4). This error will closely approximate the true one, *provided* the edges are good approximations of the geodesics. If we have S, this verification is simply a matter of comparing the geodesics to their linear approximations and ensuring that they meet the epsilon sampling requirement. If we do not know S then we need some other mechanism for measuring the quality of S. This is the role of our *shape models* (section 5) which serve as a stand-in for S and ensure that the local neighborhood accurately captures the shape (see figure 2).

Our key observation is that the samples are not random, but represent a 2D surface embedded in 3D. A good local neighborhood is one which predicts, or best explains, the local shape. Fortunately, differential geometry tells us there are only a small number of different possible local shapes, given by the signs of the principal curvatures (flat, bowl, ridge, and saddle). We define a function for evaluating what each local shape *should* look like, given a specific local neighborhood. These shape model functions are carefully designed so that their errors are directly comparable.

When evaluating a local neighborhood we simultaneously evaluate the shape models on the K nearest samples around P (the local neighborhood should be a subset

of this larger set of samples). The shape models use the LN (in particular, the estimated surface normal) in their evaluations. The LN's total score then depends on both how good the LN itself is and the best shape model fit. This helps to ensure that the LN makes sense, and it also provides us with an estimate of the local shape. By adding in specific shape models for edges, corners, and boundaries we also get robust normal estimation for these cases.

We can phrase our approach as a Bayesian estimation. If we hypothesize an error distribution, normal to the fitted surface, for the error points, we can calculate $P(s|Q_j)$, the probability of the shape model, given the local neighborhood from the residuals (the distance from the sample points to the fitted surface). If we assume that these errors are independent and normally distributed, with mean zero and variance σ, we get

$$P\left(s|Q_j\right) = \prod_j \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{e_j^2}{2\sigma^2}\right), \tag{1}$$

where e_j is the residual for the jth point in the LN, for model s. Further, if we have prior probabilities for each likelihood of the possible shapes, we can weight $P(s|Q_j)$ by these probabilities. However, in the current work, we make no assumptions, and use maximally uninformed priors (which weight each shape equally).

We next show how to compute the LN score (...)

4 Local neighborhood

Our local neighborhood (LN) evaluation consists of three pieces. The first is a validation test; these are point configurations which can be rejected because they do not form a LN. The second piece is optimization criteria for what it means to be a "good" LN. The third piece is how we calculate the surface normal from the LN. This consists of three different schemes, one each for smooth, edge, and corner neighborhoods. We also take into account potential boundary cases.

Terminology: Due to the large number of symbols used in this paper, we summarize the most-used ones in table 1. In general, capital letters denote points in 3D, and lower case letters denote those 3D points projected onto, e.g., the tangent plane. The symbol $\perp q_j$ indicates the distance to the tangent plane, and $\angle q_j \in [0, 2\pi)$ the angle in the tangent plane (using an arbitrary tangent-plane vector as the zero axis). Subscripts on 2D and 3D points indicate elements of sets, sets are denoted as $\{\}$, and vertical bars around sets indicate size. For example, $\{D\}_K \in \{D\}$ represents the $|\{D\}_K| = K$ closest points to P, and $Q_1 \ldots Q_k$, $Q_j \subset \{D\}$ represents the k points making up the LN. The $\{Q\}$ are assumed to be ordered by their angle $\angle q$ in

P	point for LN
$\hat{N}_L$	normal for LN
$\{D\}_K$	K nearest points to P (shape neighborhood)
$\{Q\}$	k LN points, sorted by angle
q_j	projection of Q_j into tangent plane
$\angle q_j$	angle of q_j in tangent plane, in range $[0, 2\pi)$
$\perp q_j$	(signed) distance to tangent plane
$\|\|q_j\|\|$	distance of q_j from origin
$\max_j \|\|q_j\|\|$	maximum width of LN polygon
$\max_j \|\|Q_j - P\|\|$	maximum width of LN in 3D
$\max_i \|\|D_i - P\|\|$	maximum width of shape neighborhood
$\mathscr{N}$	Local neighborhood metric
$\mathscr{M}$	Shape model metric
$\mathscr{F}, \mathscr{B}, \mathscr{R}, \mathscr{S}$	Primary shape model $\mathscr{M}$ metrics (flat, bowl, ridge, saddle)
$\mathscr{E}, \mathscr{C}, \mathscr{D}$	Secondary shape model $\mathscr{M}$ metrics (edge, corner, boundary)

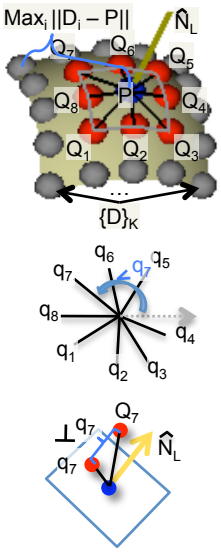

Table 1
Notation summary.

the tangent plane. Therefore, $q_1 \dots q_k$ represents the polygon formed by projecting $Q_1 \dots Q_k$ onto the tangent plane at P.

The normal of the LN is denoted $\hat{N}_L$. Script letters ($\mathscr{N}$) are used for evaluation metrics. Epsilons ε are used for constant bounds, while Ws are used for constant weights. These are summarized in appendix A.

Let d_i be the projection of D_i onto the tangent plane and $\perp D_i$ be the (signed) distance from d_i to D_i. We define a point as being on the plane if:

$$\frac{|\perp D_i|}{\sum_K ||D_i - P||/K} < \varepsilon_p \tag{2}$$

where $\varepsilon_p = 0.1$. Dividing by the average distance to P normalizes the measure.

4.1 Validation

We have three validation criteria which correspond to the formal criteria given in section 3. The first checks that the LN normal $\hat{N}_L$ induces the correct ordering of the Q_j. This is a mutual consistency check, because the LN normal is based on the ordering of the Q_j (section 4.3). Essentially, if we use the currently constructed LN normal to project the Q_j onto the tangent plane then we should recover the original ordering.

The second criterion ensures that we did not "skip over" any interior points. We take all of the points in $\{D\}$ that are closer than the 3D width of the LN $(\max_j ||Q_j - P||)$ and project them into the tangent plane. If any of these points lie inside of the polygon q_j then the neighborhood is not valid. If the sample points lie very close to the polygon boundary then this can be unduly restrictive, so we soften this restriction slightly. Let d_i be the projection of D_i onto the tangent plane, and d_i' the projection of d_i onto the boundary of the polygon q_j. If

$$\frac{||d_i - d_i'||}{||d_i - (0,0)||} < \varepsilon_I \tag{3}$$

where $\varepsilon_I = 0.05$ then we reject the LN. We do not look at points beyond the maximum size of the LN because, particularly if $K = |D_i|$ is large, the shape neighborhood may curve around on itself.

Our third criterion simply checks that the edges of q_i do not cross. This can happen when the neighborhood points do not surround P.

4.1.1 Boundary validation

If we are evaluating a boundary model then we alter the validation criteria slightly. If $\angle q_{j+1} - \angle q_j > \varepsilon_d$ then we mark that wedge as being a boundary ($\varepsilon_d = 0.8\pi$). We do not include this edge in the self-intersection test. This wedge is also excluded from the polygon inside check.

We also check to see if any points project past the boundary (see figure 3). Let $q_j p$ and $p q_{j+1}$ be the two line segments formed by joining the projection of P to the two boundary points. Then no points should project onto those line segments from outside the polygon.

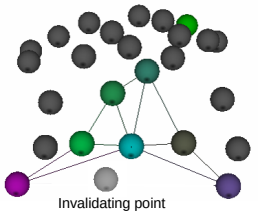

Fig. 3. Checking for points projecting outside of the polygon in the boundary case.

4.2 Optimization

The ideal local neighborhood is one that surrounds P, with P roughly in the center, the q_i evenly distributed around P and roughly an equal distance from P. Both hexagonal and grid sampling patterns tend to produce nice, even neighborhoods, the former with six neighbors, the latter with four or eight. Evenly distributed, but random, sampling patterns produce neighborhoods that lie somewhere inbetween the two. Contour sampling, on the other hand, produces elongated neighborhoods with denser sampling on the long, between-contours axis (see figure 8). Random sampling, or missed samples, can result in arbitrarily-shaped neighborhoods, with between six and twelve neighbors, on average.

Geometrically, a good local neighborhood depends on even angle spacing in the tangent plane (E_a), minimal variation in radius length (E_d), and how centered the point is (E_m). We define metrics for each of these elements, normalizing the metrics so they can be combined into a single score. We add an additional convexity term, (E_c) which particularly penalizes LNs with concave polygons.

$$\alpha_k = 2\pi/k \tag{4}$$

$$E_a = 1/k \sum_j \Big(\frac{\alpha_k - (\angle q_{j+1} - \angle q_j)}{\alpha_k}\Big)^2 \tag{5}$$

$$w_j = \frac{\angle q_j - \angle q_{j-1}}{\angle q_{j+1} - \angle q_j} \tag{6}$$

$$l_j = W_d||q_j|| + (1 - W_d)/2(||q_{j-1}|| + ||q_{j+1}||) \tag{7}$$

$$E_d = 1/k \sum_j \Big(\frac{||q_j|| - ((1-w_j)||q_{j-1}|| + w_j||q_{j+1}||)}{l_j}\Big)^2 \tag{8}$$

$$E_m = ||\frac{1/k \sum_j q_j}{\max_i ||D_i - P||} - (0,0)|| \tag{9}$$

$$v_j = \frac{q_{j+1} - q_j}{||q_{j+1} - q_j||} \tag{10}$$

$$E_c = 1/k \sum_j \Big(\big(\frac{\cos^{-1}(< v_{j-1}, v_j >)}{\pi} \big)^2 \quad \text{if convex, i.e., } \; v_{j-1} \times v_j < 0 \Big) \tag{11}$$

$$\mathscr{N} = E_a + ((1 - W_m)E_d + W_m E_m) + E_c \tag{12}$$

where all indices are taken mod k and all angle differences mod 2π. l_j is a normalization factor ($W_d = 0.2$) that blends the point q_j's length with its neighbors'. The weight $W_m = 0.2$ is how much to weight the centroid error over the length error. Note that E_a, E_d, and E_m are all normalized with respect to each other, and hence it makes sense to combine them. The E_c term is an additional error term that grows as the neighborhood becomes concave.

We have also experimented, using the test cases in section 8, with several other optimization criteria, none of which performed as well. These included distance to a fitted ellipse (tended to favor neighborhoods that zig-zagged across the ellipse), average distances (unfairly punishes elongated neighborhoods), average angle, length of polygon boundary relative to its radius, regularity of the polygon angles q_{j_1}, q_j, q_{j+1}, and barycentric coordinates. For the barycentric approach, we calculated the barycentric coordinates β_j of the point $(0,0)$ (the projection of P) with respect to the polygon q_j. The error was the measure of how close each β_j was to $1/k$ and how close $\sum_j \beta_j q_j$ was to $(0,0)$. We tried both Laplacian and Floater's shape-preserving weights (Floater, 1997). Although this approach promised to combine both angle and distance in a single metric, it unfortunately is not very discriminating for $k > 4$ and is not stable (the β_j can change rapidly for small changes in q_j), particularly when P is close to the boundary of Q_j. This is probably because the system is largely unconstrained.

4.3 Normal calculation

For most cases, the normal is computed by an angle-weighted average of the normals of the triangles $Q_j P Q_{j+1}$. Note that other weighting schemes would work equally well (Max, 1999). Unfortunately, normals for edge and corner neighborhoods are, generally, not well-approximated by any simple averaging scheme. Instead, we find the planes forming the edge or corner and average those plane normals.

For the smooth and boundary cases:

$$\hat{N}_j = \frac{(Q_{j+1} - P) \times (Q_j - P)}{||(Q_{j+1} - P) \times (Q_j - P)||} \tag{13}$$

$$\alpha_j = \angle q_{j+1} - \angle q_j \tag{14}$$

$$w_j = \begin{cases} \alpha_j & \alpha_j \leq \varepsilon_b \\ \max(0, 2\varepsilon_b - \alpha_j) & \alpha_j > \varepsilon_b \end{cases} \tag{15}$$

$$\hat{N}_j = \frac{\sum_j w_j \hat{N}_j}{||\sum_j w_j \hat{N}_j||} \tag{16}$$

The scale factor w_j decreases the influence of the corresponding triangle normal as the angle goes from $\pi/2$ to π ($\varepsilon_b = \pi/4$). These triangles represent a potential boundary, and so that normal should not unduly influence the final normal.

For the edge case we assume that P actually lies on (or very close to) the edge. Therefore, for each of the two planes, there should be at least one triangle of the LN lying in that plane. We use this fact to initialize the plane fitting. We first look at pairs of triangles with disparate normals ($| < \hat{N}_i, \hat{N}_j > | < \varepsilon_e = \pi/3$), leaving out any triangles with $\alpha_j > \pi$. If there are no such pairs then we do not try to fit planes (the LN is flat or folded). From this set of pairs (i, j) we find the pair that represents the most triangle normals:

$$\max_{i,j} \sum_k \max(< \hat{N}_k, \hat{N}_i >, < \hat{N}_k, \hat{N}_j >) \tag{17}$$

We next build two approximate planes from the triangles i and j, using the mid-point of the edge opposite P : e.g., $(1/2(Q_j + Q_{j+1}), \hat{N}_j)$. We then assign all of the *shape neighborhood* points, $\{D\}_K$, to the plane the point is closest too. If the point D_i lies on both planes, we assign it to both groups. We then least-squares fit a plane to each group to get the final planes. Again, if the angle between the plane normals is not in the range $(-\varepsilon_e, \varepsilon_e)$ we mark the data set as not an edge. The final normal is the average of the two plane normals, with the relative orientations correctly determined (section 5.5), i.e., both plane normals should point out. This is true if the points in group one lie below plane two and vice-versa.

The corner normal is found in a similar manner, except we look for three triangles with disparate normals. We relax the angle criterion to be $\varepsilon_e = \pi/4$ instead of $\pi/3$.

In the final analysis, we chose which normal to use based on the best shape model (section 5).

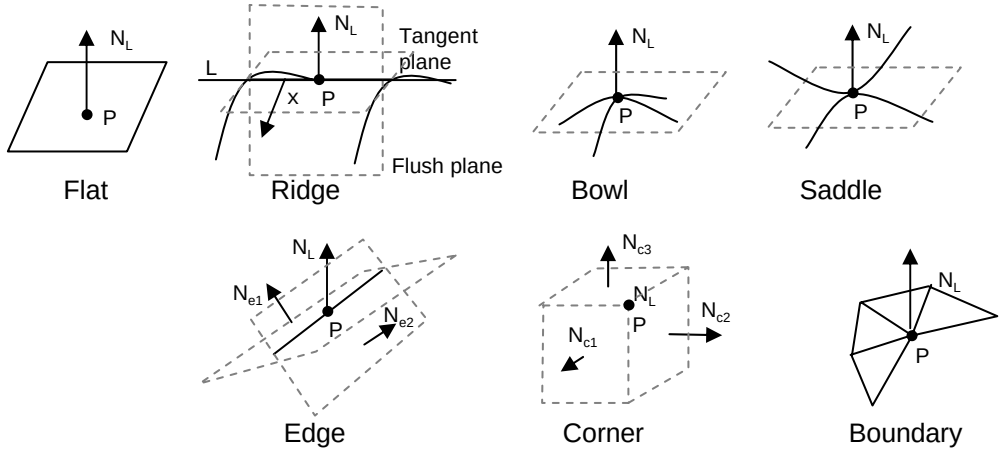

Fig. 4. Idealized shape models.

5 Shape Models

For each shape model we provide the evaluation criteria and the specific algorithm used to calculate it. The evaluation is over the entire nearby neighborhood (the K closest points $\{D\}_K$) and is based on the tangent plane formed by the point P, the surface normal $\hat{N}_L$ of the local neighborhood, and the LN itself. Changing the surface normal will change the model evaluations. We use the smooth normal (section 4.3) for all models except the edge and corner ones. We calculate each of the individual shape models ($\mathscr{F}, \mathscr{B}, \mathscr{R}, \mathscr{S}, \mathscr{E}, \mathscr{C}, \mathscr{D}$) and return the lowest one as the shape model $\mathscr{M}$ score. A key factor of our evaluation criteria is that the scores for each model are numerically normalized with respect to each other so that they can be directly compared.

Note that these models are a very straightforward numerical evaluation of the desired geometric shape, and may have appeared in one form or another in previous work. Figure 4 shows idealized versions of the models. There are three mildly tricky issues to deal with. The first is ensuring that the magnitudes of the models are all on a similar scale. In general, our metrics are such that a good fit has an error around 0.001, rising to 0.2 for a tolerable fit. For data sets that could be, e.g., a noisy plane or a ridge, the crossing point is around 0.02. These numbers rise linearly as noise is introduced (for an average 10 percent error a score of 0.001 rises to 0.01).

The second issue is ensuring that the metrics evaluate correctly regardless of how the data points are distributed, so long as there are sufficient data points to recognize the shape. For example, a ridge or edge may contain points which are near, but not exactly on, the ridge line itself. This type of data distribution causes algorithms which, e.g., find edges using dihedral angles (Jiao and Heath, 2002; Attene et al., 2005), to fail.

The third issue is ensuring that the shape model evaluation is dependent upon the

current local neighborhood. This is particularly important for non-uniform sampling of noisy data, where pure shape fitting can give incorrect results. For example, fitting a plane to contour data is likely to give a plane perpendicular to the actual surface (see figure 1).

We use a fall-off weight to extend the models to regions which are well-modeled locally around P, but possibly adjacent to features that are different. We base the fall-off f on the average and maximum radius of the D_i:

$$f(D_i) = \max(1, \varepsilon_f - \frac{(||D_i - P|| - 1/K \sum_i ||D_i - P||)}{\max_i ||D_i - P||}) \tag{18}$$

The $\varepsilon_f = 1.1$ ensures that every point D_i contributes to the sum ($f(D_i) > 0$). $f(D_i)$ is clamped to one inside of the average distance.

5.1 Flat model

The flat model $\mathscr{F}$ measures the average distance of the D_i from a plane. Because P may contain noise, we use a plane which is parallel to the tangent plane, but positioned to minimize the projected distances $| \perp (D_i \bigcup P)|$. We find the plane's location using a least-squares solver, weighting the points by the fall-off weight $f(D_i)$.

$$\mathscr{F} = \frac{1}{K+1}(| \perp P| + \sum_i f(D_i)| \perp D_i|) \tag{19}$$

5.2 Bowl model

The bowl model assumes that the surface lies entirely on one side of the tangent plane and curves away from the plane. The bowl model $\mathscr{B}$ is therefore a combination of how well the data's distance from the plane versus its distance from P is modeled by a curve (E_f) and whether or not the data does curve away from the plane (E_c). The latter ensures that we do not mark flat regions as bowls.

We assume the curving can be modeled by a simple polynomial, $y = ax^2 + c$, where x is the distance in the tangent plane from P and y is the distance from the plane. We first fit the polynomial to the data, using the distance $||D_i - P||$ for x and the distance from the plane $\perp D_i$ for y, keeping the sign of $\perp D_i$. We solve for a, c using the standard least-squares approach, with one row for each D_i and an additional row for P, weighting the rows by the fall-off value $f(D_i)$. We measure the

curvature by evaluating the polynomial at both the closest ($x = 0$) and the furthest ($x = \max_i ||D_i - P||$) distance. If the angle with the plane is less than $\varepsilon_C = \pi/8$ we add in a penalty term that grows, reaching one at $\varepsilon_c = \pi/16$, as the angle shrinks:

$$\alpha = |\tan^{-1} \frac{(c) - (a(\max_i ||D_i - P||)^2 + c)}{\max_i ||D_i - P||}| \tag{20}$$

$$E_c = \frac{\varepsilon_C - \alpha}{\varepsilon_C - \varepsilon_c} \tag{21}$$

$$E_f = \frac{1}{K+1}(c + \sum_i f(D_i)| \perp D_i - (a||D_i - P||^2 + c)|) \tag{22}$$

$$\mathscr{B} = E_f + W_c \max(E_c, 0) \tag{23}$$

$W_c = 0.1$ weights the curvature error, and we use the $f(D_i)$ fall-off function to weight a point's error by its distance from P. Note that we could just simply set c to be zero, enforcing interpolation at P, but this may not yield the best total error if P has noise.

5.3 Ridge or cylinder model

The difference between the ridge model and the bowl one is that the surface is flush with the tangent plane in one direction, and the points fall away from the tangent plane in the perpendicular direction (see figure 4). The ridge model $\mathscr{R}$ is a weighted combination of how flush the points are in the first direction (E_l), how they curve away from the plane in the perpendicular direction (E_f), and whether or not they do curve away (E_c).

We represent the flush direction as a plane that passes through P and is perpendicular to the tangent plane. We try three plane normals. The first is found by taking all of the points that are on the tangent plane, if any (eq. 2), plus P, and fitting a line to them. This line is then projected into the tangent plane; the flush plane's normal is perpendicular to this line. The other two normals we try are the plane normals that arise from fitting two planes to the data when calculating the edge normal (section 4.3). Note that simply fitting a line is not sufficient because there may not be points in the flush direction.

All points D_t on the flush plane (eq. 2) plus P are used to calculate E_l. We measure the distance of these points to the line L formed by the intersection of the tangent and flush planes.

The calculation of E_f and E_c are exactly the same as the bowl model $\mathscr{B}$, except we replace the distance $||D_i - P||$ with the distance from the line L to the projection of D_i onto the tangent plane.

Let d_t be the closest point on L to D_t, and similarly for P:

$$E_l = \frac{1}{|D_t|+1}(||P-p|| + \sum_t f(D_t)||D_t - d_t||) \tag{24}$$

$$\mathscr{R} = \frac{|D_t|+1}{|D_i|+1}E_l + \big(1 - \frac{|D_t|+1}{|D_i|+1}\big)E_f + E_c \tag{25}$$

Note that we leave the "is curved" term E_c outside of the average; this is, again, to prevent the ridge model from incorrectly matching flat regions.

5.4 Saddle model

A saddle is formed by radially alternating upward curving areas with downward curving ones, when viewed from the normal direction. The saddle shape is distinguishable by four features: It has points both above and below the tangent plane, it curves away, for any given radial direction, the points are all either above, below, or on the plane, and there are at least four changes of direction. [1] The saddle model $\mathscr{S}$ is built from two bowl models, one that curves up and another that curves down, plus penalty terms for the last conditions (E_o and E_s).

We first split the points into three groups, one group above the plane (D_u), one below (D_d), and one group on the plane (eq. 2). If either D_u or D_d has fewer than three points than we do not evaluate this model. The fit and curvature of D_u and D_d is then measured using the bowl model. Note that a polynomial only poorly approximates the curved shape of the saddle; a more flexible shape model, however, results in over-fitting.

The E_o term checks that the points in any given radial direction are all in the same group, D_u or D_d. The E_s term checks to see that there are at least four radial directions where the points on one side belong to D_u and to D_d on the other side. This is primarily what distinguishes the saddle model from a flat or ridge model with lots of noise.

$$\mathscr{S} = (1-W_g)(E_f(D_u) + E_c(D_u) + E_f(D_d) + E_c(D_d))/2 + W_g(E_o + E_s) \tag{26}$$

where $W_g = 0.3$ is a weighting of the clustering versus the fitting.

Calculating E_o and E_s correctly is somewhat difficult; a saddle has four alternating regions while a monkey saddle has six. Searching for the optimal partition lines that

[1] This leaves out saddles that are flat spots in a monotonically non-increasing function. These shapes are very unstable and best captured with the flat model.

minimize the overlaps would be prohibitive. We instead use a binning approach, adding up the percentage overlap in each bin. The bin size is a balance of two factors. Too many bins, and it becomes unlikely that any two points will share a bin. Too few bins, and points which are widely separated will be marked as overlapping. We therefore choose a bin size $\alpha_b = \lceil 2\pi/(K/s_b) \rceil$ which results in $s_b = 2$ points, on average, falling into any given bin. We then place the bin boundaries at four locations ($b\alpha_b/4, b \in [0,3]$), calculate E_o for each, and take the smallest value. Let $\angle D_i \in [0, 2\pi)$ be the projected angle of D_i in the tangent plane. We place D_i in bin $\lfloor (\angle D_i + b\alpha_b/4)/\alpha_b \rfloor$. We average the percentage overlap in each bin, skipping the bin if there are no points in it. $E_s = \max(4-s, 0)$ simply checks the number of times s the bins change from being more above than below, and vice-versa.

We also use the binning to assign some of the "flat" points to one of D_u or D_d for the bowl fitting calculation. If a flat point lies in the middle of a wedge that has more points in D_u (or D_d) then we assign that point to D_u (or D_d).

5.5 Edge model

A point is on an edge if it lies at the intersection of two planes which have sufficiently different orientations. We could use a K-means (Duda et al., 2000) clustering algorithm ($K = 2$) to find the two planes, but this does not reflect the choice of the local neighborhood. Instead, we use the edge-normal estimation technique of section 4.3 to find the two plane normals. The edge model error $\mathscr{E}$ is a weighted combination of how well the two planes fit the data (E_p), the angle between the planes (E_d), whether or not P lies on the intersection of the two planes (E_i), and if the groups fall on either side of a line in the tangent plane (E_t).

We first split the points into two groups, D_{e1}, D_{e2}, depending on which plane D_i most lies on. For each group D_e we calculate the flat error metric ($\mathscr{F}(D_e)$). E_p is then the weighted blend of each group's error. E_i is found by projecting P onto each plane, to get p_{e1} and p_{e2}. E_t is found by taking all the points in D_{e1} which lie above plane 2, and vice-versa.

$$E_p = \frac{|D_{e1}|}{|D_i|}\mathscr{F}(D_{e1}) + \frac{|D_{e2}|}{|D_i|}\mathscr{F}(D_{e2}) \tag{27}$$

$$E_d = \Big(\frac{\cos^{-1}(<\hat{N}_{e1}, \hat{N}_{e2}>) - \pi/2}{\pi/2}\Big)^2 \tag{28}$$

$$E_i = \frac{||p_{e1} - P|| + ||p_{e2} - P||}{\frac{2}{K}\sum_K ||D_i - P||} \tag{29}$$

$$E_s = \frac{\sum_{e1} \max(0, \perp_2 D_{e1}) + \sum_{e2} \max(0, \perp_1 D_{e2})}{\frac{2}{K}\sum_K ||D_i - P||} \tag{30}$$

$$\mathscr{E} = W_e E_p + (1 - W_e)(E_d + E_i + E_s)/3 \tag{31}$$

$W_e = 0.25$ weights the fit over how much this looks like an edge. The E_s term, plus the weighting in E_p, both serve to prevent the model from finding a good match by pushing most of the points into one group.

5.6 Corner model

The corner model $\mathscr{C}$ is nearly identical to the edge one, except we use three planes instead of two.

5.7 Boundary model

Unlike the other shape models, there is no natural, simple shape for boundaries because they arise when when the sampling process is incomplete, leaving a gap in the data. Therefore, the "correct" shape could be any one of the smooth models. We use the flat model score for the boundary model score and additionally mark all smooth models as being boundaries if they satisfy either of the following:

- The point $(0,0)$ is not in the polygon q_i.
- Any angle $\alpha = \angle q_{j+1} - \angle q$ is bigger than $\varepsilon_d \pi$.

5.8 Shape model score $\mathscr{M}$

We return the lowest score as the model score $\mathscr{M}$ unless one of the shape models was marked as being a boundary. If a possible boundary exists we use a slightly more complex heuristic to determine $\mathscr{M}$. Note that sharp edges and corners can also masquerade as boundaries with noise. If the best smooth shape model is flagged as being a boundary then the point is marked as a boundary. If the best shape model is an edge or corner one we check for sufficiently different planes; if the planes are close to flat we return a boundary. Otherwise, we return the best shape model as usual.

5.9 Feature size

Our feature size is based on how fast the surface curves away. For the ridge, bowl, and saddle models this is α value computed for the E_c term. For edges and corners, this is the angle between the tangent plane and the planes making up the edge or corner. See figure 17.

6 Heuristic algorithm

The heuristic algorithm begins with a very dense neighborhood and greedily culls out points which contribute the most to the LN score. Each model is scored independently; different models will, usually, have different best LNs. The algorithm returns the model, and corresponding LN, with the best shape model (SM) score. We use the SM score alone because it is a better measure of how well the LN approximates the surface normal.

We use the three orthogonal Eigenvectors of the SVD as starting normals for the primary models (flat, ridge, bowl, saddle). The LNs snap into stable configurations very quickly, even if the initial normal is off by as much as 45 degrees. We experimentally checked using just three normals versus the twelve evenly-spaced ones from section 7 and found that the additional normals only rarely affected the result (< 5 percent), and then only the LN score, not the SM score.

Since the points' contributions to the overall LN score are correlated, a greedy culling algorithm is not guaranteed to find the best set of points. We address this by adding some randomness to the selection mechanism. Instead of greedily selecting the point that contributes most to the LN scored, we use importance sampling (Duda et al., 2000) to pick points for culling. Points are selected with a probability proportional to their contribution to the LN score. The LN score and probabilities are recomputed after each cull. Extremely bad points, which have a high influence due to the least-squares fit, are selected with high probability, while selections between similarly bad points are essentially randomized. Although we still cannot be guaranteed to find the best set of points, importance sampling can be shown to reduce the variance of the final LN score, and has proven to be highly effective in our experiments.

For the secondary models (edge, corner, boundary) we first check that one of these models might exist because there is a projection direction which leaves a large gap in the LN (section 6.4). If this is the case, we use the estimated best boundary normal and K-means clustering (Duda et al., 2000) to find an initial guess for the edge and corner normals. If not, then we do not search for a boundary and we use the smooth model normals (ridge, bowl) as a starting point for the edge and corner ones.

6.1 Initial ordering

Given a normal, we produce an initial set Q_j by projecting the D_i into the tangent plane, then binning the d_i by their angle (32 bins, minimum angle $\varepsilon_s = \pi/16$). Within each bin we keep only the closest point. As a precaution, after we recalculate the normal we check to see if any of the $d_i \notin q_j$ lie inside of the polygon

Iterate over three SVD normals $\hat{N}_s$

  Set normal to $\hat{N}_s$

  Iterate until stable

    Find maximum local neighborhood Q_j

    Order Q_j

    Calculate normal $\hat{N}$ from P and Q_j

  While can remove point from Q_j

    Order local neighborhood Q_j

    Calculate normal $\hat{N}_L$ from P and Q_j

    Calculate $\mathscr{N}$

    Calculate $\mathscr{M}$ for flat, ridge, bowl, saddle using D_i

    Keep Q_j and $\hat{N}_L$ with best $\mathscr{N} + \mathscr{M}$ for each model

If valid (edge,corner,boundary)

  Set initial (edge,corner,boundary) normal

  While can remove point from Q_j

    Order local neighborhood Q_j

    Calculate normal $\hat{N}_L$ from P and Q_j

    Calculate $\mathscr{N}$

    Calculate $\mathscr{M}$ for (edge,corner,boundary) using D_i

    Keep Q_j and $\hat{N}_L$ with best $\mathscr{N} + \mathscr{M}$

  Redo (up to 5 times) all good, valid models with importance sampling

Return model with best SM score or if boundary (section 5.7)

Fig. 5. Heuristic algorithm.

q_j; if so, we add them to the LN. We loop four times, stopping if the normal stabilizes ($< N_i, N_{i-1} >> \varepsilon_n = 0.95$). Typically, the normal stabalizes within one to two iterations.

6.2 Removal of points

Points are sorted for removal by scoring them according to the LN evaluation criteria of section 4. We do not remove points that would cause the LN to become

invalid, or would introduce an incorrect boundary. Note that the scoring criteria need to be recalculated every time the surface normal changes. Although the shape model $\mathscr{M}$ score depends on the LN, it primarily depends on the LN surface normal and the set of all points D_i, neither of which change (much) when points are removed from the LN.

Validation criteria: We do not remove the LN point q_j if the resulting angle in the tangent plane ($\angle q_{j+1} - \angle q_{j-1}$) is bigger than $\varepsilon_b = \pi/2$. We also do not remove the point if the 3D angle between $Q_{j-1} - P$ and $Q_{j+1} - P$ is bigger than ε_b (the tangent plane angle can be made smaller than the 3D one by tilting the plane). Finally, we see if q_j is inside the polygon made by removing q_j (eq. 3). Note that we do not need to explicitly build this polygon, we only need to check if the segment $\overline{q_j p}$ intersects $\overline{q_{j-1} q_{j+1}}$.

The score S for a point q_j is basically q_j's contribution to the LN evaluation (eq. 12). Because the desired angle changes as the number of points changes, we alter that part of the calculation to avoid creating very large, very small, or unbalanced angles. Specifically, q_j's score is formed by multiplying a desired average angle score S_a by a combined length S_l, convexity S_c, balanced angle S_b, and small angle S_s score. We multiply rather than add to better balance the angle and distance metrics. The S_a multiplier shrinks to zero as the angle gets bigger than the ideal one. The S_b term favors removing points which are not equally between their neighbors, and the S_s term favors removing points with very small ($\varepsilon_s < \pi/16$) angles. The S_l and S_c terms are the same as the evaluation ones, except we scale down S_l's contribution if the point is shorter (rather than longer) than its neighbors (we want to remove far away points).

$$\alpha_k = 2\pi/k \tag{32}$$

$$\alpha_j = \angle q_{j+1} - \angle q_{j-1} \tag{33}$$

$$S_a = \begin{cases} 1 - \min 1, \frac{\alpha_j - \alpha_k}{\alpha_k} & \alpha_j > \alpha_k \\ 1 & \alpha_j \leq \alpha_k \end{cases} \tag{34}$$

$$w_j = \frac{\angle q_j - \angle q_{j-1}}{\angle q_{j+1} - \angle q_j} \tag{35}$$

$$l_j = W_d ||q_j|| + (1 - W_d)/2(||q_{j-1}|| + ||q_{j+1}||) \tag{36}$$

$$a_j = (1 - w_j)||q_{j-1}|| + w_j ||q_{j+1}|| \tag{37}$$

$$S_l = \begin{cases} \frac{||q_j|| - a_j}{l_j} & ||q_j|| > a_j \\ 1/10 - 1/10 \frac{a_j - ||q_j||}{\max_j ||Q_j - P||} & ||q_j|| \leq a_j \end{cases} \tag{38}$$

$$v_j = \frac{q_{j+1} - q_j}{||q_{j+1} - q_j||} \tag{39}$$

$$c_j = (\frac{\cos^{-1}(< v_{j-1}, v_j >)}{\pi})^2 \quad \text{if convex, i.e. } v_{j-1} \times v_j < 0 \tag{40}$$

$$S_c = (c_{j-1} + c_j + c_{j+1})/3 \tag{41}$$

$$S_b = |(\angle q_j - \angle q_{j-1}) - (\angle q_{j+1} - \angle q_j)|/(2\alpha_j) \tag{42}$$

$$S_s = 2\big(\max(1, \frac{\varepsilon_s}{\angle q_j - \angle q_{j-1}}) + \max(1, \frac{\varepsilon_s}{\angle q_{j+1} - \angle q_j})\big) \tag{43}$$

$$S = S_a(S_l + S_c + S_b + S_s) \tag{44}$$

We can also remove more than one point at a time. Currently, we remove two points at a time in the initial model calculation loops and one in the subsequent importance sampling loop.

6.3 Importance Sampling

We run the greedy point removal again with importance sampling instead of deterministic removal. This can help remove biases in the greedy procedure, particularly for points which have very similar removal scores. We only repeat the models which have a score within a factor of five from the best score found.

For data sets with relatively uniform sampling distributions this additional sampling is unnecessary. However, for uneven sampling, particularly contour sampling, it can greatly increase the local neighborhood score.

6.4 Implementation issues, timings, and analysis

Avoiding unnecessary model evaluations: We can eliminate evaluating the boundary, edge, and corner model calculations for points that do not have a projection direction that results in a gap when all of the points are projected into the tangent plane. We take twenty uniformly distributed normals N_j and take the normalized dot product of those normals with every vector $(< N_j, (D_i - P)/||D_i - P|| >)$. We assign each vector to the closest N_j, and keep a running sum of the absolute value of the dot products for each normal.

$\|\{D\}_K\|$	All	Smooth	All, no sampling	Smooth, no sampling	SVD	K
15	11.3	3.1	3.4	2.3	0.8	0.10
25	12.2	7.5	8.6	5.5	1.8	0.15
30	14.7	10.0	11.5	7.2	2.5	0.18
45	25.6	17.3	20.2	12.5	5.0	0.26
60	36.6	24.1	27.7	17.0	9.6	0.40

Table 2
Average time (in ms) to run the algorithm on a single data point for the Gargoyle data set (129,721 points). $|\{D\}_K|$ is the size of the shape neighborhood. **All** is trying all shape models and using importance sampling. **Smooth** is leaving out the edge, corner, and boundary models, but using importance sampling. The second two collumns leave out importance samplig. **SVD** is the time to compute a plane fit. **K** is the time to get the nearest $|\{D\}_K|$ neighbors. Minimum size was six and maximum size 65 for all runs. Platform: Pent-M 1.5GHz, 768 MB Ram.

For each normal N_j which does *not* have a vector assigned to it, we determine if the points D_i, when projected to N_j's tangent plane, have a gap bigger than $\varepsilon_d \pi$. If the number of normals without gaps is twice as big as the number of normals with gaps then that point is not a potential boundary.

The normal N_j with the lowest accumulated score is used as the initial guess for the boundary normal.

Running time: The running time of the algorithm is linear in the size of the shape neighborhood, for $|D_i| > 32$, because the maximum starting local neighborhood size is capped at 32 (the number of bins). This also bounds the number of times the Remove Points loop is called, the local neighborhood score calculation, and the smooth normal calculation. The model, evaluation, and edge/corner normal calculations are all linear in $|D_i|$.

We can optionally run the algorithm leaving out the edge and corner models, or not doing the additional importance sampling, or both. Table 2 lists the time savings for each of these cases for different neighborhood sizes. Total running time for the Gargoyle data set ranges from a few minutes (smooth models only, no importance sampling, small neighborhood size) to a half an hour (all models, importance sampling, large neighborhood).

Shape neighborhood size: The user provides a minimum, average, and maximum shape neighborhood size. If the heuristic algorithm fails to find a valid non-boundary model, then it is run again with either a smaller (if no valid boundary model found) or bigger neighborhood size, until the minimum or maximum size is reached. Shrinking helps when the shape neighborhood has wrapped around a feature, or included points from adjacent surfaces. Growing helps bridge gaps. Typical neighborhood size triplets for our data sets range from (6,15,25) on sparse data sets

to (15,25,35) on laser-scan data. Invalid cases only rarely arise, and usually only near noisy boundaries, sharp edges, or where the data is not locally 2D.

Re-calculating the models: We only re-calculate the models as points are removed if the surface normal changes by more than $\varepsilon_m = 0.98$, since the shape model depends on the normal, not the local neighborhood.

7 Exhaustive search algorithm

The exhaustive search algorithm searches all of the possible neighbor combinations Q_j contained in the K closest data points to determine which one is the best. For each shape model we keep the best combination of shape model and local neighborhood shape found over all Q_j. We then return the best Q_j for the best shape model. This prevents a good local neighbor score from over-riding a poor model fit.

There are four parts to this algorithm: Enumerating the possible neighbor sets, determining an ordering (if any) of the points Q_j such that the criteria in section 3 hold, evaluating the shape model, and evaluating the quality of the local neighborhood itself.

Ordering: A local neighborhood at P consists of a set of ordered points $Q_j \subset D_i$. The ordering is found by projecting the Q_j into the tangent plane at P, then sorting by angle relative to an arbitrary vector in the tangent plane. Once we have an ordering, we can estimate the surface normal by taking a weighted combination of the triangle Q_jPQ_{j+1} normals (section 4.3). This presents a chicken-and-egg problem, since we need a surface normal to project the points in the first place. We get around this by alternating projection and normal calculation until the normal stabalizes and the calculated surface normal gives the same ordering as the projection one.

We initialize the chicken-and-egg search with twenty evenly-spaced normals. For each normal, we order the Q_j and calculate a new surface normal. We then use this new surface normal to re-order the Q_j and then re-calculate the surface normal. We verify that this final normal produces the same ordering of Q_j in the validation step. Each unique ordering of Q_j (usually only one) is tested.

Evaluation: To evaluate a local neighborhood we use the following:

- A set of routines which implement linear versions of the validation requirements (section 4.1). The local neighborhood must pass these validation tests.
- A metric $\mathscr{N}$ for measuring both the angle and the distance regularity of the local neighborhood (section 4.2).
- A shape model metric $\mathscr{M}$ which, given a local neighborhood, tries all possible

shapes and returns the one with the lowest error (section 5).

Enumeration: Let D_i be the set of K closest points to P. To find the local neighborhood, we search over all of the K choose k possible neighbors combinations in D_i. K must be big enough to include the local neighborhood, but not so big that the D_i do not form a simple shape (bowl, ridge, saddle). The algorithm is summarized in figure 6.

Iterate over K choose k, $k \in 4 \ldots K$ (unordered Q_j)
  For each of 20 evenly-spaced normals $\hat{N}$
    Iterate until calculated $\hat{N}$ same as projection $\hat{N}$ (max 4 iterations)
      Order local neighborhood Q_j using current $\hat{N}$
      Re-calculate normal $\hat{N}$ from P and Q_j
    If already tried Q_j or Q_j not valid, go to next one
    Calculate $\mathscr{N}$ from $\hat{N}_L$, Q_j
    Calculate $\mathscr{M}$ for each shape using D_i and $\hat{N}_L$
    For each shape model
      Keep Q_j if $\mathscr{N} + \mathscr{M}$ smallest so far
Return Q_j from best shape model.

Fig. 6. Calculating the local neighborhood.

Figure 7 shows plots of $\min_{Q_j}(\mathscr{N} + \mathscr{M})$ for each value of k for all sampling patterns from our test cases (section 8.1). Note that small k tend to have the smallest error, particularly for samples on a grid, and that there is a dip in the curve after which the error is monotonically increasing.

Obviously, if we have normals for the points then we do not need to search for the best initial normal. Moreover, we can eliminate all points from the neighborhood which have normals in the opposite direction.

7.1 Evaluating the heuristic algorithm

To verify that the heuristic algorithm returns near-optimal values, we ran 30 trials over each of the test cases with error rate 0.1 (for both parameter and function error), and recorded the scores found by the heuristic algorithm. We then calculated the large-sample 95% upper confidence bound on each set of 30 scores, and compared these to the exact scores, which we generated exhaustively. In all cases, the heuristic score was within 2% of the optimal score, with a mean of 0.88%, a median of 0.72%, and a variance of 0.50% (all at a confidence level of 95%).

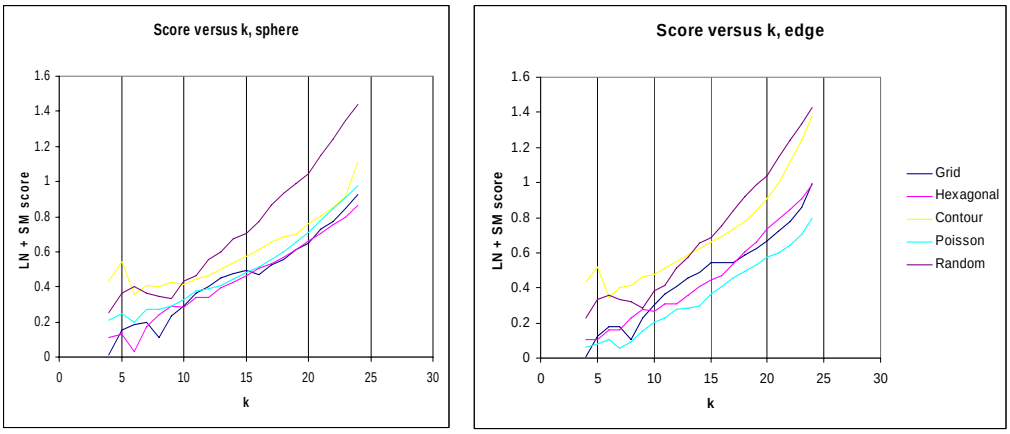

Fig. 7. Evaluation score as k increases. Note the dips in the curve. Left graph: bowl shape. Right graph: edge shape.

Total	Heuristic	Evaluation	Normal	Neighborhood	Model
17.6 σ 5.1	3.2 σ 1.2	4.3 σ 1.6	4.5 σ 4.0	4.0 σ 1.1	1.5 σ 2.3

Table 3
Running times, in ms, with standard deviation. Test set size is $6 \times 5 \times 121 = 3630$ samples, varying both parameter and function noise from 0 to 1 in increments of 0.1. Platform: Pent-M 1.5GHz, 768 MB Ram.

Running times for the heuristic algorithm, broken down by algorithm stages, are given in Table 3.

8 Test cases and evaluation

We have created a suite of test cases which vary the sampling distribution and noise but have a known (if noisy) shape and normal. We use this test suite both to validate the LN and SM criteria, and to test our normal reconstruction.

8.1 The test suite

We generate five $n \times n$ sampling patterns in the range $(-0.5, 0.5) \times (-0.5, 0.5)$, shown in figure 8. The patterns are a grid, a hexagonal grid, contours ($3n$ sampling rate in one direction), random but evenly distributed (Poisson distribution), and random. We introduce noise by randomly jittering the initial samples up to $\pm(1/n)/2$ of the original spacing distance.

We lift the samples onto one of six shapes, flat, cylinder, sphere, edge, and corner (figure 9). Noise is added in the z direction, except for the corner, where noise is

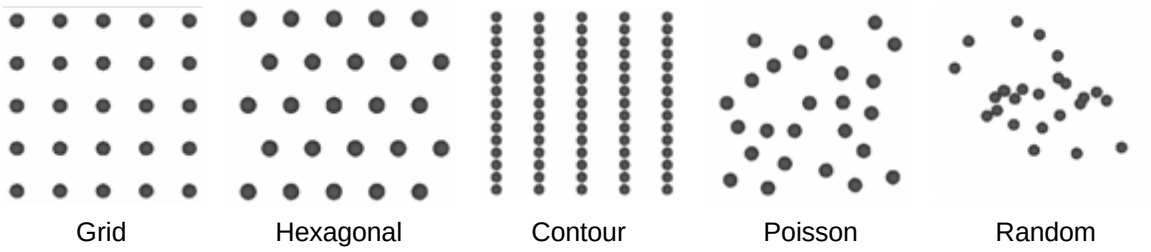

Fig. 8. Sampling patterns.

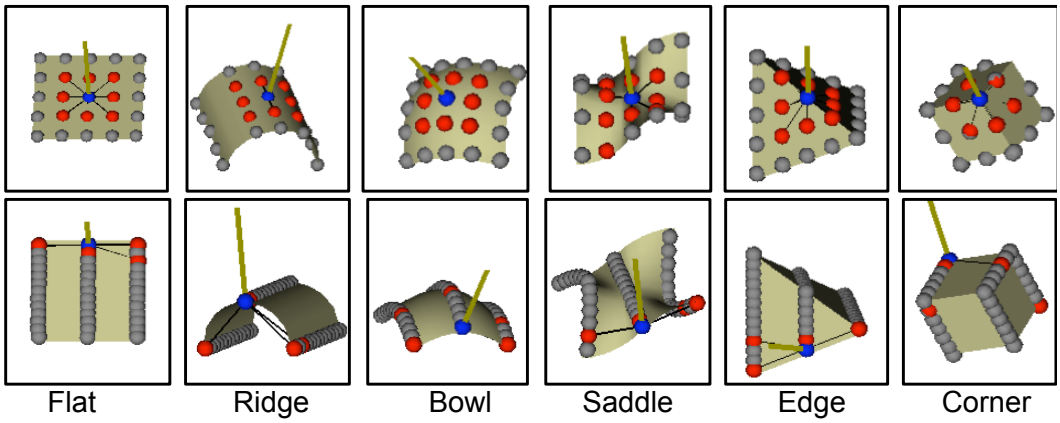

Fig. 9. Test shapes with example local neighborhoods. Top row: interior points. Bottom row: boundary points.

added perpendicular to the plane the sample point lies on. The added noise is in the range ± 0.1.

The cylinder equation is the top half of a unit cylinder, scaled down by 1/2 and translated so that the center point is at $(0,0,0)$. The saddle equation is $1.5s(s^2 - 3t^2)$. The sphere is a reverse stereo projection, with the result translated down by 1 so the top of the sphere is at $(0,0,0)$. The edge is placed diagonally through $(0,0,0)$, with the two planes at right angles to each other and folding down. The corner model places the lower quadrant on the plane $z = 0$, folds the upper left quadrant onto the plane $y = 0, x < 0$, and the lower right quadrant to the plane $x = 0, y < 0$. The remaining quadrant is distributed over the top half with some noise.

8.2 Test case evaluation

The default number of samples we used was $K = 24$ (a 5 by 5 grid), with the contour pattern having 10×3 samples. The point P is either the point closest to $(0,0,0)$ (interior tests) or a point on the boundary (see Figure 9).

For the grid and hexagonal sampling patterns with noise at 0.1 and 0.4 we know that the LN should have eight and six points, respectively. For the remaining test cases,

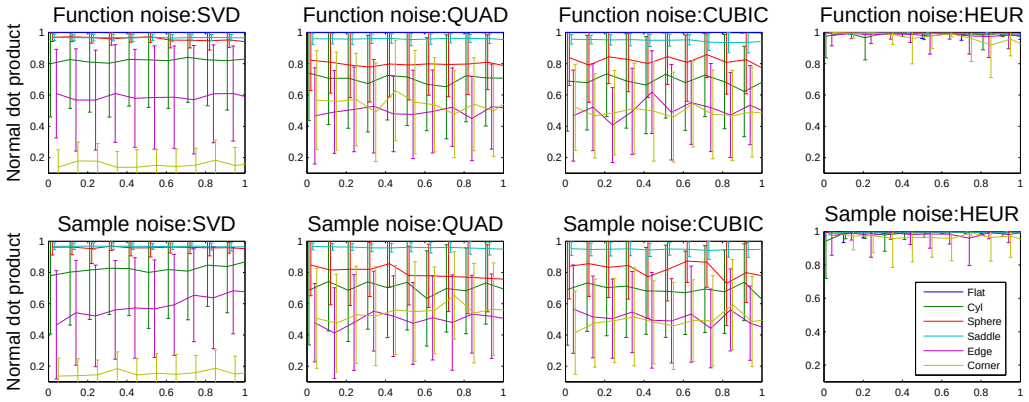

Fig. 10. Behaviour of normals as function (top row) and sample (bottom row) noise is varied for (from left to right) SVD, quadratic, and cubic fitting, and our approach. Each curve corresponds to a specific test case shape (flat, ridge, bowl, saddle, edge, and corner). The data is averaged over the different sampling distributions (figure 8) and the sample (top) or function (bottom) noise. Vertical bars show the standard deviation. Note that bars for each shape are offset slightly to make them visible.

we have verified the neighborhoods by hand which is, unfortunately, a somewhat subjective evaluation. Some examples are shown in figure 9; more examples are shown in the supplemental materials.

We know the surface normal and shape for each of our test cases. We check the returned shape model and normal against the known answer (normal comparison uses a dot product). Note that noise can cause the edge and corner models to degenerate into the ridge and bowl models, respectively, and that sampling noise can cause the saddle model to look flat, locally. Figures 10 and 11 (right) plot the accuracy of the recovered normal as the noise levels increases for each shape (heuristic algorithm). The normal reconstruction is fairly insensitive to increased sampling and function noise.

In comparison, we show the equivalent plane-fitting normal reconstruction (Mitra and Nguyen, 2003) and quadratic and cubic fitting (Vančo and Brunnett, 2007) in figures 10 and 11 (interior and boundary point, respectively). Fitting accuracy for each shape is shown. Note that our algorithm performs better for non-flat shapes.

8.3 Real-data comparison

We verify the normal error results observed in the test cases with real data sets. Figure 12 shows comparisons of the real versus estimated normals for two models, a low-resolution version of the bunny and a marching-cube sampled dragon. We categorize the normals based on feature shape in order to more clearly show that the

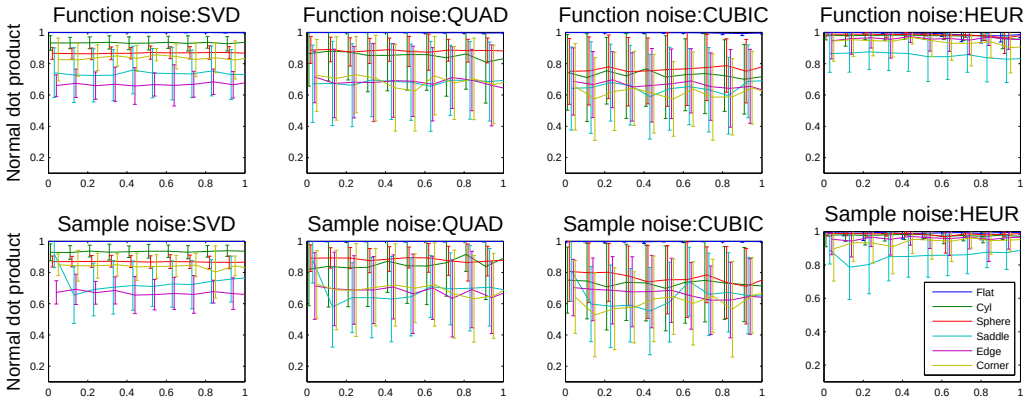

Fig. 11. Same as figure 10, except for a point located on the boundary of the test shape.

	bunny	2-holed mug	dragon	gargoyle	figure	radius
LN	.981, .057	.909, .215	.996, .033	.999, .063	.984, .063	.989, .057
SVD	.935, .149	.758, .348	.990, .053	.995, .019	.974, .086	.973, .116
Quad	.880, .204	.747, .321	.968, .102	.974, .073	.928, .147	.961, .115
Cubic	.860, .230	.678, .341	.968, .105	.977, .071	.924, .162	.956, .131
Del	.952, .121	.951, .102	.985, .032	.991, .022	.974, .080	.979, .083

Table 4
Comparison to known normals. Bunny is a sparse (2002) downsampling of the original data set, 2-holed mug is a contour, undersampled smooth surface, dragon is a Marching Cubes mesh, gargoyle is a full laser scan data set, figure is an evenly-sampled, C^4 surface with known normals, and radius is a contour data set. Given is average and standard deviation of the dot product of the calculated normal with the known one.

performance depends on the local shape. We compare with the (local) SVD (Mitra and Nguyen, 2003) and quadratic reconstruction (Vančo and Brunnett, 2007) and with the (global) Delaunay approach (Dey and Sun, 2005). The SVD and quadratic fits were calculated using one round of normal-based weighting (Vančo and Brunnett, 2007).

Average and standard deviation values for several different types of models are given in table 4, corresponding images in figure 13. Note that the first two data sets do not meet our sampling criteria; in particular, the two-holed mug case has many places where the surfaces are closer than the corresponding sampling rate. In this case, the global algorithm out-performs the local one, although our algorithm does successfully identify where there are problems (see figure 13). Further examples are provided in the supplemental materials.

Figure 14 illustrates the behaviour of the different algorithms under the addition of Gaussian noise. Each point was perturbed by up to 0.2 percent of the average

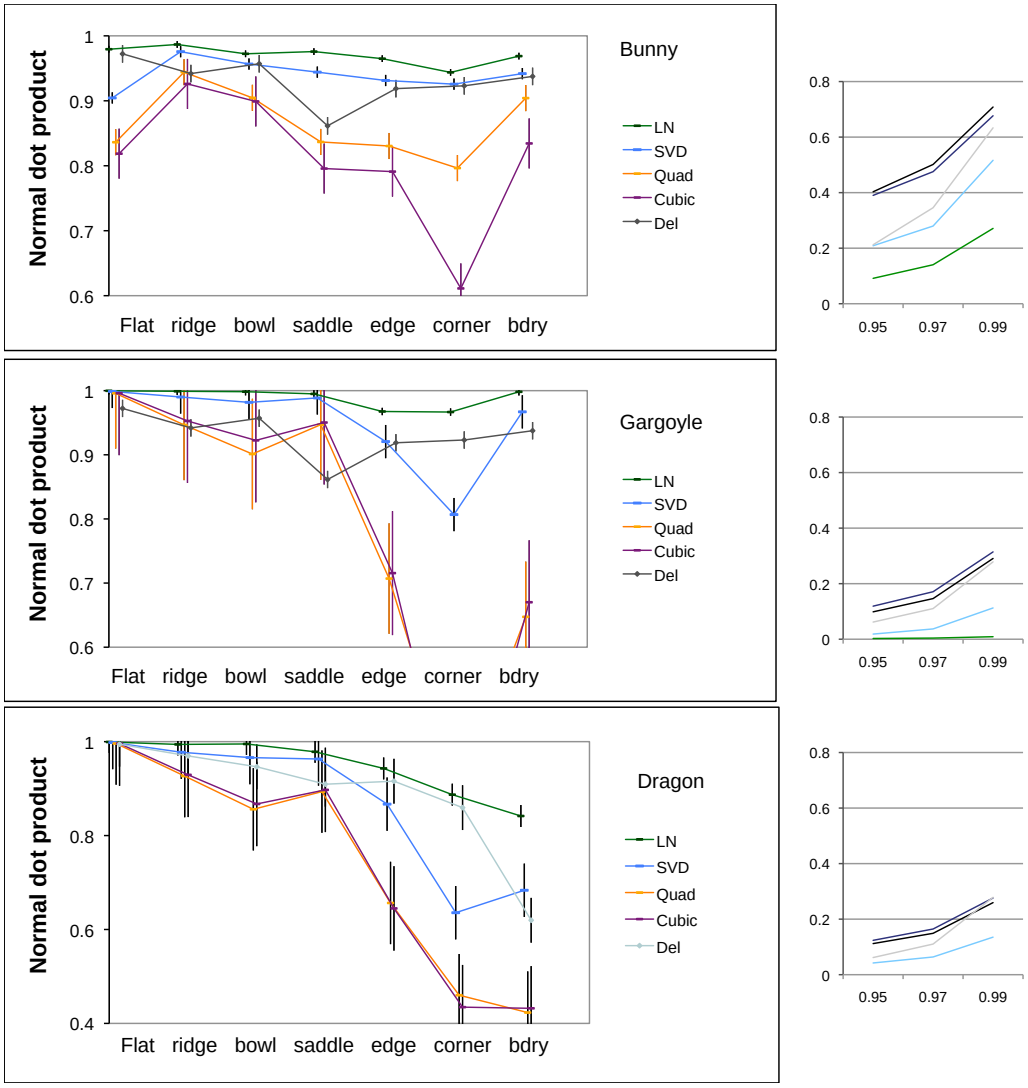

Fig. 12. Comparison to SVD, quadratic, cubic, and a Delaunay approach on real data sets (top: low-res bunny, middle: Gargoyle, bottom: Marching cubes dragon). Normals are grouped by shape classification (from left to right: Flat, ridge, bowl, saddle, edge, corner, boundary). Standard error bars are shown slightly separated for clarity. On the right are plots of the number of points for which the dot product is below 0.95, 0.97, and 0.99.

edge length at each iteration. The comparison was to the normals from the original data set. As the graph in figure 14 shows, the local neighborhood out-performed both the SVD and the Delaunay approach. This is because the average local shape, even with noise, remains similar, and the algorithm picks the local neighbors (and hence surface normal) that best predict the local shape. This is particularly true for non-flat regions; we plot the reconstruction error for the ridge points to illustrate this.

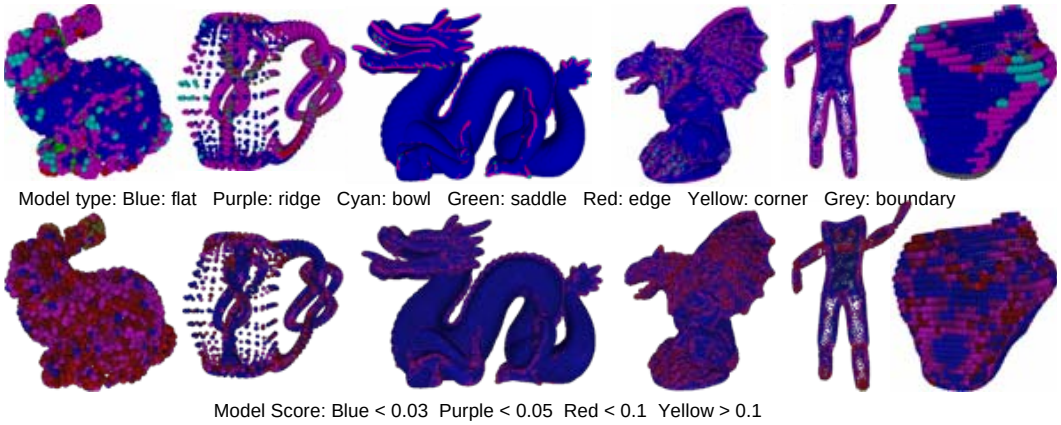

Fig. 13. Model type and model score for the bunny, 2-holed mug, dragon, gargoyle, figure, and radius data sets.

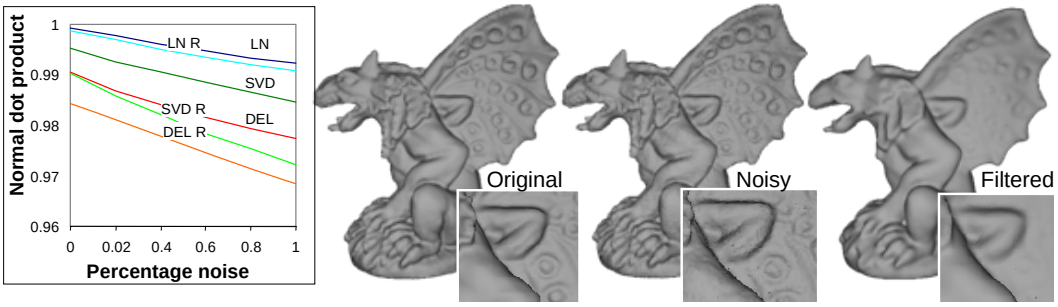

Fig. 14. Behaviour under the addition of noise for local neighbor, SVD, and Delaunay. Shown are average normal error for all data points (LN,SVD,DEL) and for ridge points (LN R, SVD R, DEL R) (roughly 40,000 of 130,000 total points). Model error as noise increases: 0.035, 0.042, 0.047, 0.051, 0.055, 0.058. Right: Surface reconstruction result for original, maximum noise, and filtered data sets.

9 Applications

We demonstrate using local neighborhoods for two applications, surface reconstruction of surfaces with spherical topologies and smoothing. Figure 15 shows several genus zero, manifold, water-tight mesh reconstructions from point clouds and figure 14 shows the result of applying simple isometric smoothing on the point sets before reconstruction.

Our surface reconstruction is a novel approach which first parameterizes the data set using a modified spherical parameterization algorithm Saba et al. (2005). Once the point data set is embedded on the sphere we run a convex hull algorithm (Barber et al., 1996) to triangulate it. This is guaranteed to produce a water-tight, manifold mesh of spherical topology. The mesh is a Delaunay triangulation on the sphere; when we move the vertices back to their original positions in 3D the triangles will no longer be Delaunay. We therefore run an edge-swap optimization to improve the triangulation.

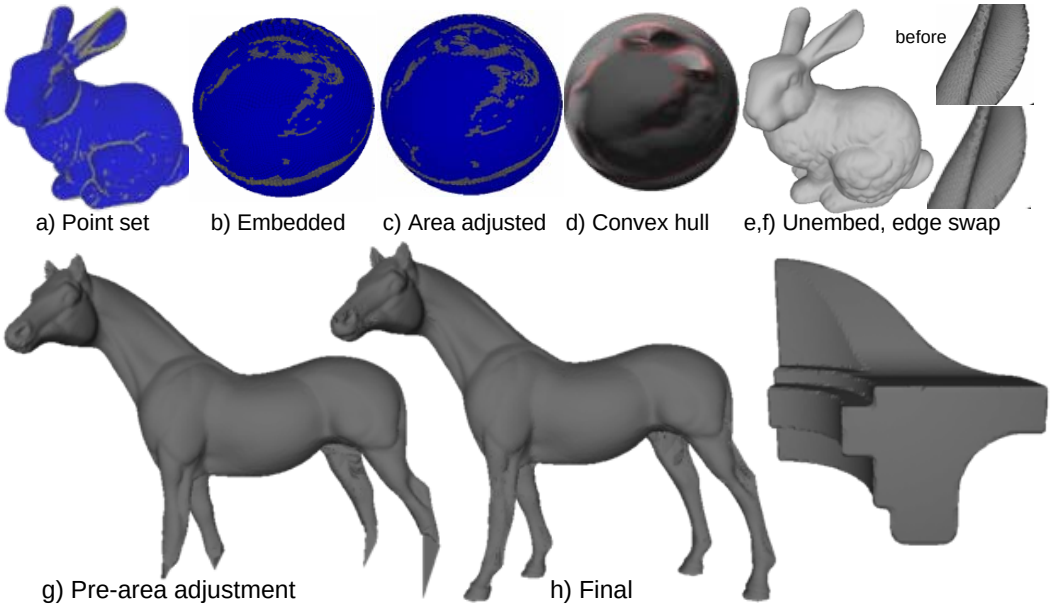

Fig. 15. Top: Reconstructing a surface. From left to right: a) The initial point set, colored by feature size. b) The points embedded on the sphere. c) Applying area adjustment. d) Tessellating the sphere using a convex hull algorithm. e) Moving the mesh vertices back to their original positions and performing edge swaps. f) Ear before and after edge swaps. Bottom: g) Horse before area normalization. h) Horse after area normalization. Right: Fan-disk model.

9.1 Modifying the sphere embedding

We make two changes to the original spherical parameterization. We modify the initial hemisphere partition step and introduce a set of weights to more evenly distribute the points on the sphere. To create the initial hemispheres we grow two disks starting with two widely-separated points, as in the original algorithm, using the local neighborhoods as the graph structure. If a point's local neighborhood lies in both disks we mark it as belonging to the boundary. This results in a band of points, instead of a single, closed curve, around the equator. To sort these points we use a modified version of Isomap (Pless and Simon, 2002) which maps the points to a circle. From here, the algorithm proceeds as before, placing each point at the (weighted) center of its local neighborhood instead of the one ring of a mesh.

One issue with embedding the points on the sphere is that, particularly for long, skinny regions like the horse's legs, the points at the ends of the legs lie so close together on the sphere that the Delaunay triangulation considers them to be coplanar. To address this, we first embed the points using Floater's shape preserving weights (Floater, 1997). We follow this with several (60 to 200) area-expanding passes where the weights are based on the area of the local neighborhood *on the sphere*. Specifically, for each point P we calculate an approximate area on the sphere for P's local neighborhood by summing the area of the spherical triangles

Fig. 16. Using Polymender to produce a water-tight surface from a set of triangles.

Fig. 17. Using the same local neighborhoods, but increasing shape neighborhoods (10, 35), for labeling features. Blue is flat, yellow is a sharp fold.

formed by $q_i p q_{i+1}$. The weight for the point q_i is Q_i's area divided by P's total area. Note that these weights will change as the embedding locations for the points change; we recalculate the weights every 5 iterations.

For the filtering examples we use a very simple filter (80% of own location summed with 20% of average of local neighborhood locations) to adjust the 3D point locations. Figure 14 shows the gargoyle model with Gaussian noise added in then filtered out (20 iterations of filtering).

9.2 Alternative model construction

The local neighborhood can be treated as a collection of triangles and sent to Polymender (Ju, 2004). All Polymender reconstructions (see figure 16 and additional material) were run with a depth of 8 and a 70% volume fill.

9.3 Feature size

We can vary the size of the shape neighborhood in order to get features at different scales. Figure 17 shows an example of this for the dragon and the fan-disk models. Feature size is an absolute measure (see section 5.9) based on how fast the surface curves away from the tangent plane.

10 Remarks and conclusion

Local neighborhoods and shape descriptors are a robust approach to surface normal estimation, local feature analysis, and determining where the sampling is (potentially) insufficient or incorrect. Although specifically design to handle non-uniform sampling cases, it also performs better on uniformly sampled data sets than previous algorithms.

Although the local neighborhood approach by itself can not bridge large gaps in the data set or correctly disambiguate when surfaces are closer together than their sampling rate, it does provide feedback (in the model score) indicating where potential problems lie. The average model scores also correlate directly with the noise in the samples, particularly when restricted to data points labeled as flat.

A Weights

We summarize all of the "magic numbers" used here. Constants (the εs) were chosen to be very conservative, are generally based on absolute geometric considerations, and are mostly used to cull out obvious bad cases. Weights (the Ws) are used to balance the different terms of the evaluation metrics. These were selected largely by experimentation using the test cases. The results are fairly insensitive to small $(-0.05, 0.1)$ changes to the weights.

References

Amenta, N., Choi, S., Kolluri, R. K., 2001. The power crust. In: SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications. ACM Press, New York, NY, USA, pp. 249–266.

Attene, M., Falcidieno, B., Rossignac, J., Spagnuolo, M., 2005. Sharpen&bend: Recovering curved sharp edges in triangle meshes produced by feature-insensitive sampling. IEEE Transactions on Visualization and Computer Graphics 11 (2), 181–192.

Barber, C. B., Dobkin, D. P., Huhdanpaa, H. T., Dec 1996. The quickhull algorithm for convex hulls. ACM Trans. on Mathematical Software 22 (4), 469–483, http://www.qhull.org.

Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., Taubin, G., /1999. The ball-pivoting algorithm for surface reconstruction. IEEE Transactions on Visualization and Computer Graphics 5 (4), 349–359.
URL `citeseer.ist.psu.edu/bernardini99ballpivoting.html`

Boissonnat, J.-D., Cazals, F., 2000. Smooth surface reconstruction via natural neighbour interpolation of distance functions. In: SCG '00: Proceedings of the

$\varepsilon_I = 0.05$	Perc inside, LN (4.1)
$\varepsilon_b = \pi/2$	Maximum allowed angle, heuristic (6,4.3)
$\varepsilon_s = \pi/16$	Smallest desirable angle, heuristic (6)
$\varepsilon_d = 0.8\pi$	Boundary angle, SM (4.1.1)
$\varepsilon_c = \pi/16$	Minimum fall-off angle, SM (5.2,5.3,5.4)
$\varepsilon_C = \pi/8$	Maximum fall-off angle, SM (5.2,5.3,5.4)
$\varepsilon_l = 0.05$	Maximum percentage distance to boundary, LN validation (4.1)
$\varepsilon_p = 0.1$	Point on plane, LN, SM (4,5)
$\varepsilon_e = \pi/3, \pi/4$	Angle between edge,corner normals, LN (4.3)
$\varepsilon_n = 0.95$	Normal is stabalized, LN (6.1)
$\varepsilon_m = 0.98$	Recalculate the shape model, SM (6.4)
$W_d = 0.2$	Percentage of own length, LN (4.1,6)
$W_m = 0.2$	Centroid weight, LN (4.1,6)
$W_c = 0.1$	Curvature weight, SM (5.2,5.3,5.4)
$W_g = 0.3$	Clustering weight, SM (saddle) (5.4)
$W_e = 0.25$	Edge-like and corner-like weight, SM (5.5,5.6)

Table A.1
Summary of constants and weights.

sixteenth annual symposium on Computational geometry. ACM Press, New York, NY, USA, pp. 223–232.

Dey, T. K., Goswami, S., 2003. Tight cocone: a water-tight surface reconstructor. In: SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications. ACM Press, New York, NY, USA, pp. 127–134.

Dey, T. K., Goswami, S., 2004. Provable surface reconstruction from noisy samples. In: SCG '04: Proceedings of the twentieth annual symposium on Computational geometry. ACM Press, New York, NY, USA, pp. 330–339.

Dey, T. K., Li, G., Sun, J., 2005. Normal estimation for point clouds : A comparison study for a voronoi based method. In: Eurographics Sympos. on Point-Based Graphics. pp. 39–46.

Dey, T. K., Sun, J., Jul. 2005. Normal and feature estimations from noisy point clouds. Tech. Rep. OSU-CISRC-7/50-TR50, Ohio State.

Duda, R., Hart, P., Stork, D., 2000. Pattern Classification, 2nd Edition. Wiley-Interscience.

Fleishman, S., Cohen-Or, D., Silva, C. T., 2005. Robust moving least-squares fitting with sharp features. ACM Trans. Graph. 24 (3), 544–552.

Floater, M. S., 1997. Parametrization and smooth approximation of surface triangulations. Computer Aided Geometric Design 14 (3), 231–250, iSSN 0167-8396.

Gatzke, T., Grimm, C., June 2006. Estimating curvature on triangular meshes. Inter-

national Journal of Shape Modeling 12 (1), 1–29, how best to compare curvature metrics on meshes.

Gelfand, N., Guibas, L. J., 2004. Shape segmentation using local slippage analysis. In: SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing. ACM, New York, NY, USA, pp. 214–223.

Hildebrandt, K., Polthier, K., Wardetzky, M., 2005. Smooth feature lines on surface meshes. In: SGP '05: Symposium on Geometry Processing.

Jiao, X., Heath, M. T., June 2002. Feature detection for surface meshes. In: Numerical Grid Generation in Computational Field Simulations. pp. 705–714.

Jones, T. R., Durand, F., Desbrun, M., 2003. Non-iterative, feature-preserving mesh smoothing. ACM Trans. Graph. 22 (3), 943–949.

Ju, T., 2004. Robust repair of polygonal models. ACM Trans. Graph. 23 (3), 888–895.

Kolluri, R., 2005. Provably good moving least squares. In: SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, pp. 1008–1017.

Lai, Y.-K., Zhou, Q.-Y., Hu, S.-M., Wallner, J., Pottmann, H., Jan./Feb. 2007. Robust feature classification and editing. IEEE Transactions on Visualization and Computer Graphics 13 (1), 34–45.

Max, N., 1999. Weights for computing vertex normals from facet normals. J. Graph. Tools 4 (2), 1–6.

Mitra, N. J., Nguyen, A., 2003. Estimating surface normals in noisy point cloud data. In: SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry. pp. 322–328.

Pauly, M., Keiser, R., Kobbelt, L. P., Gross, M., 2003. Shape modeling with point-sampled geometry. ACM Trans. Graph. 22 (3), 641–650.

Pless, R., Simon, I., 2002. Embedding images in non-flat spaces. In: Proc. of the International Conference on Imaging Science, Systems, and Technology.

Saba, S., Yavneh, I., Gotsman, C., Sheffer, A., June 2005. Practical spherical embedding of manifold triangle meshes. Shape Modelling International, 256–265.

Taubin, G., 1995. A signal processing approach to fair surface design. In: SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques. ACM Press, New York, NY, USA, pp. 351–358.

Vančo, M., Brunnett, G., Mar 2007. Geometric preprocessing of noisy point sets: an experimental study. Special Issue on Geometric Modeling (Dagstuhl 2005), 365–380.