Shape Classification and Normal Estimation for Non-uniformly Sampled, Noisy Point Data

Abstract

We present an algorithm for robustly analyzing point data arising from sampling a 2D surface embedded in 3D, even in the presence of noise and non-uniform sampling. The algorithm outputs, for each data point, a surface normal, a local surface approximation in the form of a one-ring, the local shape (flat, ridge, bowl, saddle, sharp edge, corner, boundary), the feature size, and a confidence value that can be used to determine areas where the sampling is poor or not surface-like.

We show that the normal estimation out-performs traditional fitting approaches, especially when the data points are non-uniformly sampled and in areas of high curvature. We demonstrate surface reconstruction, parameterization, and smoothing using the one-ring neighborhood at each point as an approximation of the full mesh structure.

Keywords: Surface reconstruction, normal estimation, shape classification, unlabeled data, point data, one-ring

1. Introduction

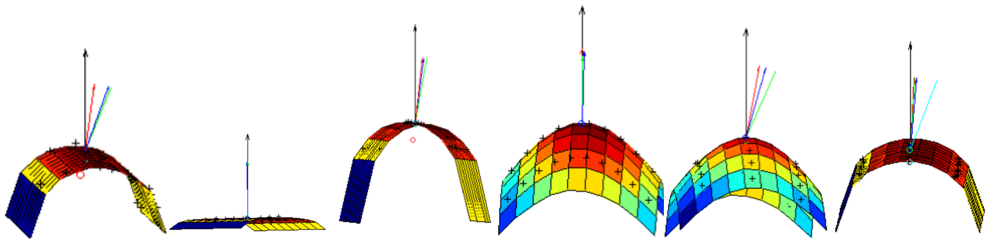
We present an algorithm for estimating surface normals and local shape from point data that is sampled from a 2D surface embedded in 3D space. We show that our algorithm is robust in the presence of both noise and non-uniform sampling. For each data point the algorithm produces a surface normal, a local surface approximation in the form of a one-ring, a local estimate of shape (flat, bowl, saddle, ridge, edge, corner, or boundary), a measure of the feature size, and a confidence value. This value reflects both the quality of the local sampling and noise in the function and can be used to detect places where the points do not represent a surface. The one-ring neighborhood can be used for further mesh processing such as reconstruction, smoothing or spectral mesh processing.

Traditional normal estimation approaches usually rely on fitting a local surface approximation to some subset of the k-nearest neighbors Klasing et al. (2009). This approach works well most of the time, but it has several limitations (see Figure 1). First, if the data are non-uniformly distributed, the fitting error becomes biased — the classic example of this is contour data, where the best planar fit can be *perpendicular* to the surface. Second, the local surface approximation may not have enough flexibility to match the surface, which introduces additional error and increases the sensitivity to non-uniformly distributed data. Increasing the flexibility of the representation, unfortunately, can lead to over-fitting. Third, there is no method for distinguishing between noisy data and poor local fit, or determining if the local samples even represent a surface.

Our fundamental idea is to build three *different* representations of the local surface — a surface normal, a good-quality one-ring, and a local shape model — and cross-validate them. If all three representations are mutually consistent with each other *and* the data, then we can be reasonably confident that they are correct. In this we are closer in spirit to approaches that use robust statistics (Fleishman et al., 2005; Li et al., 2010a). We also gain a lot of information about the samples, namely how much noise is present, how even the sampling is, if there are sharp features or boundaries, the local feature size, and a plausible graph structure. There are several approaches that use one of these pieces of information to *derive* another — for example, normals from the graph structure (Park et al., 2006) — but to our knowledge no-one else has used cross-validation to improve normal estimation and surface analysis.

We show that cross-validation produces better-quality normals than standard fitting approaches, particularly in areas with uneven sampling and high curvature. Not only are the averages better, but the variance is narrower, meaning we are less likely to return a "wrong" result. This is particularly true in saddle and ridge areas. Our approach also behaves well near boundaries and sharp features, and can explicitly identify them.

We make use of three observations. The first is that, locally, smooth surfaces are either flat (zero curvature), ridges (zero curvature in one direction), bowls (positive curvature), or saddles (negative curvature). Given a known surface normal, we evaluate whether or not the data can be plausibly explained by one of these models (Section 5). If it can't, then either the surface normal is wrong or the data points do not, locally, form a surface. Second, a good-quality one-ring approximation of the surface — one that has



(a) Biased (b) Increasing curvature (c) Contour sampling, two views (d) Weighted fit

Figure 1: Black is true normal, red is plane fitting, green is quadratic, and blue cubic. a) More samples on one side than the other pull the normal estimation in that direction. b) Increasing the curvature (same samples) results in error in the normal. c) Two views, contour sampling — more samples on the right contour pull the normal in that direction. d) Weighted fit (cyan arrow) can exacerbate the problem.

roughly equal-sized, equilateral triangles — is the best method for estimating the normal because it does not suffer from inadvertent smoothing or over-fitting. (Smoothing can always be applied in a post-processing step if desired — see Section 8.) Third, the one-ring and the normal should be mutually consistent — projecting the data points onto the tangent plane should yield the same one-ring (Section 4). This is particularly important around ridges with high curvature, where the range of valid normals is small.

We use a combination of optimization and validation to find the surface normal, one-ring, and local surface model that are mutually consistent. We define evaluation scores for the shape models (Section 5) and for the one-ring (Section 4) along with cross-validation criteria (Section 3). Note that the search space is discrete — there are an enumerable number of one-rings. We use this fact to develop a heuristic algorithm that generates valid one-ring candidates and then optimizes them (Section 3).

We evaluate our normal construction and shape classification using both real and test data (Section 7). We demonstrate the usefulness of our one-rings for both surface reconstruction and smoothing (Section 8). Our contributions are:

- Robust normals even in the presence of noise in the data and non-uniform sampling.

- Construction of a well-behaved, minimal one-ring neighborhood from the k-nearest neighbors.
- Local shape estimation (flat, bowl, saddle, ridge, corner, sharp edge, or boundary).
- An optimization algorithm to find the above that cross-validates the results.
- Identification of outliers and poorly reconstructed areas.
- Two novel surface reconstruction algorithms based on the constructed one-rings.

Source code is available at `https://sourceforge.net/projects/meshprocessing/`.

2. Related 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 1). However, the global approach can be more accurate in cases where the between sample distance is less than the between surface distance (section 7.2).

The most common approach to normal estimation is plane fitting. (Mitra and Nguyen, 2003) provide a formula for estimating the best number of neighbors, 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. The fitting 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. A recent survey (Klasing et al., 2009) also found quadratic fitting more accurate for moderate noise, although for very high noise planar fitting was better. We compare our approach to both planar, quadratic, and cubic fitting and show that, particularly for areas of high curvature and uneven sampling, our approach outperforms these surface fitting approaches (section 7). 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., 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., 2005) and (Li et al., 2010a) provide a statistically robust method for locally classifying points around sharp features (which are the intersection of smooth surfaces) into clusters, using robust statistical approaches. 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 exacerbated 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 and increases the effect of noise. In our approach we only use a smaller k when the data samples do not constitute a valid surface (end of Section 6), gaining the benefits of using smaller neighborhoods without losing the benefits of larger ones. Alternative graphs over all data points, which could be used to create local neighborhoods, are minimum spanning trees, relative neighborhood graphs, and Gabriel graphs. These are typically too sparse for our purposes, although a recent elliptical adaptation to the Gabriel graph (Park et al., 2006) is denser. We differ from these approaches in that we both optimize for the quality of the one-ring and verify that it is a valid approximation to the surface (but we do not create a consistent, global graph).

Delaunay triangulation also defines a concept of local neighborhood, called the "natural neighbors" (Boissonnat and Cazals, 2000). This concept is used in (OuYang and Feng, 2005) to extract a set of mesh neighbors from a global Delaunay triangulation. Our one ring is usually an ordered subset of the natural neighbors, and is 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 (Sections 5 and 6). 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, particularly for data sets captured from CAD/CAM models.

3. The algorithm

The input to the algorithm is a set of N data points $D = d_1 \dots d_N$, and whether or not the surface has boundaries, corners, or sharp edges. Let $Q = q_1 \dots q_k$ be the k nearest neighbors for the point d ($k = 25$ in our implementation). For each data point d the algorithm outputs the following.

1. A surface normal $\hat{n}$.
2. An ordered one-ring neighborhood $P = p_1 \dots p_m$, with $p_i \in Q$. The one-ring has the following properties:
   (a) The surface normal computed from the one-ring neighborhood is $\hat{n}$.
   (b) The points P, when projected to the tangent plane defined by d and $\hat{n}$, form a non self-intersecting polygon.
   (c) The points Q, when projected to the tangent plane defined by d and $\hat{n}$, lie on, or outside, of that polygon. These two conditions essentially ensure that the one-ring represents (locally) a disk of a manifold surface.
3. The local shape type $s \in S$, which is one of: **flat, bowl, saddle, ridge, corner, sharp edge, or boundary**. The latter three will only be used if the user has explicitly said they are present in the data.
   (a) The local shape has the surface normal $\hat{n}$.
   (b) An estimate of the curvature of the local shape (feature size).
4. A noise score, $e_s(d, \hat{n}, Q)$, which represents the average variation of Q from the fitted shape model. This score is normalized.
5. A quality score, $e_p(d, \hat{n}, P)$, for the one-ring neighborhood. This score is normalized.

```
Iterate over candidate normals
  Find dense, valid one-ring $P$
  Remove points from $P$ to create $P'$
    Compute normal $\hat{n}'$ for $P'$
    If $P'$ still valid
      Compute $e_p(d, \hat{n'}, P')$
      Compute $e_{s \in S}(d, \hat{n'}, Q)$ for all shapes $S$
      Keep if $e_p(d, \hat{n'}, P') + \min_{s \in S} e_s(d, \hat{n'}, Q$, minimum
```

Figure 2: Heuristic algorithm.

The algorithm optimizes for the P, $\hat{n}$, and s that minimize $e_s(d, \hat{n}, Q) + e_p(d, \hat{n}, P)$ while ensuring that the properties in items 2 and 3 hold. The difficulty is that the one-ring is used to compute the surface normal, which in turn is used to find a one-ring neighborhood that satisfies the given properties (and minimizes $e_p(d, \hat{n}, P)$). This is addressed by guessing a surface normal, then iterating between computing the one-ring and the normal until both stabilize (if they do). This initial one-ring is dense; we further optimize it by removing points. Removing points might change the computed normal; we check that this candidate one-ring is still valid by re-projecting and checking the properties in item 2. If the candidate one-ring passes this test then we calculate $e_p(d, \hat{n}, P)$ and the shape model scores $e_{s\in S}(d, \hat{n}, Q)$ for each model. The score for the candidate one-ring is $e_p(d, \hat{n}, P) + \min_{s\in S} e_s(d, \hat{n}, Q)$. We return the candidate one-ring, normal, and shape which have the lowest combined score. This algorithm has the following components, described in more detail in the following sections and in Grimm and Smart (2008):

- Calculate an initial, cross-validated surface normal and one-ring pair (Section 4).
  - Calculate a surface normal $\hat{n}$ from a one-ring P
  - Calculate an initial one-ring $P \subset Q$ from a surface normal $\hat{n}$.
  - Calculate candidate surface normals.
- Optimize surface normal and one-ring pairs using the score $e_p(d, \hat{n}, P)$ (Section 4.1).
  - Compute $e_p(d, \hat{n}, P)$.
  - Generate candidate one-rings P' from P.
- Compute shape models $e_{s\in S}(d, \hat{n}, Q)$ (Section 5).
  - Fit shape models (bowl, ridge, flat, saddle) to Q given d and $\hat{n}$.
  - Evaluate resulting fitted shape model.
- Specialized shape models (sharp edge, corner, boundary) (Section 6).
  - Compute additional candidate normals $\hat{n}$ from the corner and sharp edge models.
  - Fit corner, sharp edge, and boundary models to Q given d, P, and $\hat{n}$.
  - Evaluate resulting fitted specialized shape models.
- Optimizing in the presence of boundaries, corners, and sharp edges (Section 6).

The outlined heuristic algorithm in Figure 2 works, but can be made more efficient by not recomputing shape models when the normals are sufficiently similar (dot products within 0.98 in our implementation). Additionally, we can make an initial pass with a handful of candidate neighborhoods and rule out any shape models with scores substantially bigger than the best-fit models.

We use the angle-weighted normal averaging but we could just as easily use other normal calculation techniques (Meek and Walton, 2000; OuYang and Feng, 2005). Because we are optimizing for one-ring quality, which includes regular triangles, the specific weighting scheme has less effect than it would for a general mesh with fixed (possibly bad) one-ring topology.

Role of cross-validation: As mentioned in the introduction, our goal is not just to calculate any normal, one-ring, and local shape type, but to also ensure that they are mutually consistent. We ensure consistency between the normal and the one-ring by calculating the normal from the one-ring, then checking that the one-ring is still valid when created with that normal (criterion 2 above). This consistency check particularly helps with anisotropic sampling, where simpler approaches often result in normals that are orthogonal to the correct one. Any projection of the local neighborhood using any normal will result in a few of the points in the sparse sampling direction being included in the one-ring, which results in a one-ring normal that points (roughly) in the right direction. Projecting with this new normal results in a one-ring that has an even better sampling of the sparse direction — a few iterations and the normal and one-ring will agree.

In areas of high curvature or in the presence of outliers simply checking the normal and the one-ring is not sufficient. It is possible in these cases for there to be more than one stable normal and one-ring configuration — a sharp ridge, when viewed sideways, looks like a noisy boundary. For this reason, we cross-validate with the larger local neighborhood, i.e, the k nearest neighbors. Most of the time we cross-validate in only one direction — given the normal, see if the k nearest neighbors form a "valid" shape. For high-curvature areas this helps to disambiguate a ridge from a boundary, because the ridge will fit better than the boundary model. Outliers are also less of a problem because they tend to get removed from the one-ring (if possible) because the resulting normal is not as good at predicting the local shape.

In the case of edges, corners, and boundaries we also cross-validate in the other direction by trying the normal predicted by the model. Like most sharp feature detection approaches, we partition the local neighborhood into

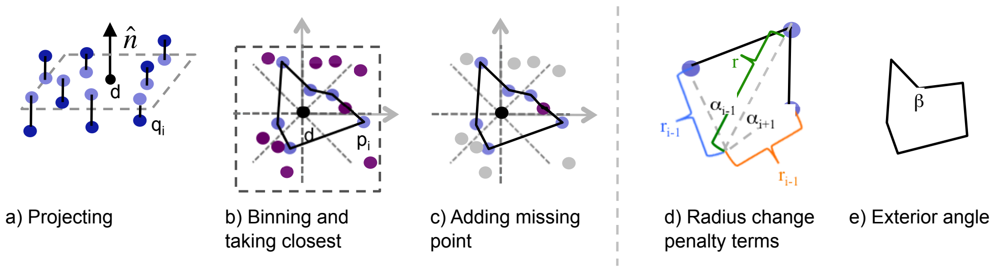


Figure 3: From left to right: a) Projecting the points to the tangent plane, then b) binning by angle, to construct a one-ring neighborhood. c) The highlighted point in the first quadrant is inside the polygon; this point is added to the polygon. d) Calculating the radius change term. e) An example of an exterior angle.

two (edge) or three (corner) sets where each set is well-explained by a plane. Unlike these approaches, we cross-validate by checking that the normal made by averaging the plane normals agrees with the normal of the one-ring, that the one-ring can also be partitioned into those planes, and that the normal produces that one-ring. This is both faster (the one-ring tells us which planes to try) and more robust (see Figure 7).

4. One-ring computation

To find an initial, valid P and $\hat{n}$ pair we iterate between computing a one-ring from a normal, then computing the one-ring from the normal. We iterate at most four times for a given starting normal; if no valid pair is found then we either mark the point as invalid or try other normals as directed by the user (Section 6).

Validating P, $\hat{n}$: To cross-validate a one-ring P with a normal $\hat{n}$ we perform the following checks: 1) Calculate the new normal $\hat{n}'$. Check that $< \hat{n}, \hat{n}' >> 0.95$, ie., the normals are essentially the same. 2) Re-project the points Q (which includes P) onto the $d, \hat{n}'$ tangent plane. Verify that all of the projected points lie on, or outside of, the polygon formed by projecting the P points. Check that the re-projected polygon P does not cross, or fold back on, itself. These checks ensure that our one-ring is a “tight” disk around d (Section 3).

One-ring from normal: To construct a one-ring from a normal we project all of the points Q to the tangent plane centered at d with normal

$\hat{n}$ (this places the projection of d at the origin). We bin the points into 32 even wedges, keeping only the closest point found in the wedge (if any). This forms an initial, dense one-ring P with at most 32 points. We then check that all of the projected points lie inside of this polygon, adding in any that do not – this can happen when points in neighboring bins are at greatly differing distances from d (see Figure 3c). This ensures that our one-ring meets criterion 2c) in Section 3.

Normal from one-ring: We use the standard angle-weighted normal averaging (Meek and Walton, 2000) with one exception; we use a decreasing weight for angles bigger than 25% of the total. These represent gaps in the data.

Initial normals: We compute the singular value decomposition of the $k \times 3$ matrix $Q_j - d$ and use all three vectors as initial, candidate normals. Other candidate normals may be added if we detect edges, corners, or boundaries (Section 6).

4.1. One-ring evaluation

The ideal one-ring neighborhood P is one that surrounds d, with d roughly in the center and the points in P evenly distributed around d. We evaluate the quality of P not in 3D, but using the 2D polygon created by projecting the points P onto the tangent plane. Our metric $e_p(d, \hat{n}, P)$ is the average of three terms: even angle spacing, minimal change in radius length, and how centered the point d is. We then add to $e_p(d, \hat{n}, P)$ an additional term which penalizes concave polygons (see Figure 3e).

Let n be the number of points in the projected polygon P, and p_i be the projection of the i^{th} point of P. The angle term $e_a(i)$ is the difference, squared, of the angle α_i from the average, divided by the average. The radius change term $e_r(i)$ is the difference between the radius at p_i and the average of the previous and next radii (weighted by angle) and scaled by the overall radius. The centered term e_c is the difference between the polygon's centroid and the origin, divided by the maximum polygon radius R. The convexity term e_x is the exterior angle (divided by π) for any convex boundary point. The first three terms are normalized so they can be combined; the convexity term grows as the one-ring becomes concave.

$$e_a(i) \quad = \quad \left(\frac{\alpha_i - \frac{1}{n}\sum_j \alpha_j}{\frac{1}{n}\sum_j \alpha_j} \right)^2 \qquad (1)$$

$$
\begin{aligned}
e_r(i) &= \left(\frac{\left(r_i - \frac{1}{2}\left(\frac{\alpha_{i-1}}{\alpha_{i-1}+\alpha_{i+1}} r_{i-1} + \frac{\alpha_{i+1}}{\alpha_{i-1}+\alpha_{i+1}} r_{i+1}\right)\right)}{r_i + r_{i-1} + r_{i+1}} \right)^2 && (2)\\
e_c(P) &= \frac{\|\frac{1}{n}\sum_i p_i\|}{R} && (3)\\
e_x(P) &= \sum_i \frac{\beta_i}{\pi}^2, \beta_i > \pi && (4)\\
e_p(d, \hat{n}, P) &= e_x(P) + \left(e_c(P) + \frac{1}{n}\sum_i e_a(i) + \frac{1}{n}\sum_i e_r(i)\right)/3 && (5)
\end{aligned}
$$

This combination was arrived at experimentally using the test cases (Section 7).

We have also experimented with other error metrics, including a subset of the ones listed, distance to a fitted ellipse (tended to favor neighborhoods that zig-zagged across the ellipse), average distances (unfairly punishes elongated one-rings), length of polygon boundary relative to its radius, and barycentric coordinates. For the barycentric approach, we calculated the barycentric coordinates of the projected point d. The error was the measure of how close the barycentric coordinates were to $1/n$. We tried both Laplacian and 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 more than 4 points and is not stable.

4.2. One-ring optimization

Once we have an initial valid one-ring, we optimize it by picking a point that reduces e_p when removed. We score each point by the change in $e_p(d, \hat{n}, P)$ when that point is removed. We use importance sampling (Duda et al., 2000) to actually pick the point to remove; this introduces stochasticity into the algorithm and helps prevent local minimum. In particular, we can replace step 3 in algorithm 2 with a routine that removes several points from P at once before evaluating, but restarts from the initial candidate one-ring several times (three in our implementation).

5. Shape models

The shape models are shown in Figure 4. Each shape model is fit to the points Q by optimizing the free parameters, given that we know what the surface normal must be. These models are deliberately simplistic with few free parameters; their role is to determine what the local surface shape is, and provide a noise estimation. To this end, the model error functions (e_s) are carefully constructed so that the measured error is consistent between models. The error rises linearly with noise (eg, if the original fit score was 0.01, introducing 10% noise increases the error to 0.1. In general, a good fit has an error around 0.001, rising to 0.2 for a tolerable fit. Errors bigger than that indicate problems or outliers.

The free parameters are chosen so that the distribution of the points has little, or no, effect, provided there are enough points to recognize the shape (ie, for a ridge there must be points both along the ridge and on either side). Because the shapes are simplistic they tend not to hold for large regions; for that reason we use a fall-off weight when solving for the parameters. This weight is

$$l_a = \frac{1}{k}\sum_i ||q_i - d|| \tag{6}$$

$$w_i = \max\left(1, 1.1 - \frac{||q_i - d|| - l_a}{\max_i ||q_i - d|| - l_a}\right) \tag{7}$$

We include the point d when solving for the parameters.

A point Q_i is considered to be on the tangent plane through d with normal $\hat{n}$ if it's projected distance is less than 0.1 of the average distance:

$$\frac{1}{10k}\sum_i ||Q_i - d|| \tag{8}$$

Flat model: For the flat model we know the surface normal; all that's left to find is where the tangent plane is located along $\hat{n}$ (one degree of freedom). We find this location using a weighted least-squares solve (Eq. 7). The error e_s is the residual error of the solve.

Bowl model: The bowl model assumes that the surface lies entirely on one side of the tangent plane passing through d with normal $\hat{n}$, and that points further from d are further below the plane (this ensures that a flat region is not marked as a bowl). We assume the curving can be modeled by a simple polynomial, $y = ax^2 + b$, where x is the distance in the tangent plane from d and y is the distance below the tangent plane. We solve for a, b using a weighted least-squares solve. The reason we include b is to allow the tangent plane to slide up or down, accounting for noise in the normal direction at d.

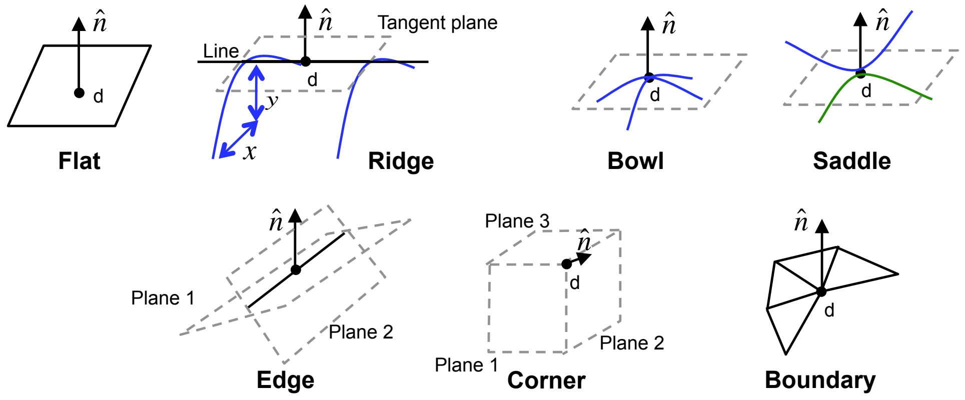


Figure 4: Idealized shape models. $\hat{n}$ is the normal used to fit the shape model at the point d. All curves (ridge, bowl, saddle) represent a polynomial $y = ax + b$. The flat model has one free parameter, the location of the plane along $\hat{n}$. The ridge model has three free parameters, the orientation of the line L in the tangent plane and two parameters for the curve (a and b). The bowl model has two free parameters (a and b). The saddle model has four — two each for the up and down curves. The edge model has eight parameters (the two intersecting planes). The corner has twelve (three planes). Any one-ring that is incomplete (the gap is bigger than a threshold) is marked as a potential boundary.

We measure how much the bowl curves down by calculating the angle α between the starting point of the polynomial ($x = 0$) and the ending point $x = \max_i ||q_i - d||$. Angles less than $\pi/16$ indicate a plane, not a bowl. For angles between $\pi/16$ and $\pi/8$ we introduce a penalty term $0.1(\pi/8 - \alpha)/(\pi/16 - \pi/8)$. This penalty is added to the residual fit error to give e_s.
Ridge model: The difference between the ridge model and the bowl one is that the surface is flush with the tangent plane in one direction. Given the direction $\vec{t}$ in which the surface is flush, we calculate a fall-off polynomial as in the bowl model, except we use the distance to the line $d + \vec{t}$ for x.

We try three different $\vec{t}$ directions, taking the one with the best fit value. 1) Take all of the points that lie on, or near, the tangent plane and fit a line to them; project this line onto the tangent plane. 2,3) Calculate a sharp edge model (Section 6). Intersect each of the two resulting planes with the tangent plane.
Saddle model: A saddle is formed by radially alternating upward curving areas with downward curving ones, ie, there are points both above and below the tangent plane, and the further from d the point Q_i is, the further from the tangent plane it is. To measure the curving, we divide the points into two groups (above and below), putting points that are close to the tangent plane into both groups. We then fit a bowl model to each group. The e_s term is the average of the two bowl models.

To prevent a noisy local neighborhood from looking like a saddle, we add an additional penalty term to ensure that, in any given radial direction, the points all curve up (or down). We bin the points by angle, then count the number of bins that have points that belong to both the up and down group. We use $4\pi/k$ bins. The penalty term is 0.1 times the number of bad bins.
Feature size: For the curved models we use the angle α to provide an absolute measure of the local feature size. For flat models we use the angle α of the best-fit smooth model.

6. Edge, corner, boundary models

If there are sharp edges or corners in the data than we can explicitly fit a model to them, which results in better normal estimation.The boundary model assumes that the one-ring is incomplete and is used to flag boundary points in the data set. It is safe to include these models even if these features are not present in the data set; however, it increases the computation cost, which is why they are optional.

An edge is formed when two planes with sufficiently different orientations meet. Similarly, a corner is formed when three planes meet. To determine if we have an edge (or corner) we need to first find those two (or three) planes. The observation we use is that the triangle wedges of the one-ring (formed by d, P_i, P_{i+1}) are a good approximation to the planes. We therefore look at all pairs (triplets) of wedges who's normals are sufficiently different (angle difference greater that $\pi/3$ for edges, $\pi/4$ for corners). If there are no pairs (triplets) then there is no edge (or corner).

Normal calculation: To calculate the normal, we assign each point in Q to one of the candidate planes, assigning it to both planes if it's close enough (Eq. 8). We then fit planes, as in the flat model. To ensure the normals are oriented correctly, they are flipped if they do not point in the same direction as the candidate planes. The normal is the average of the two (three) plane normals. We only keep normals that pass the validity check in Section 4.

If we find a valid edge or corner model we add that normal to the list of candidate normals. We cross-validate edge and corner models slightly differently. We use the normal found by the candidate planes and only try one-rings which are valid when projected by that normal. We additionally check that, when walking around the one-ring, the triangle wedges split into two consecutive groups (three for corner) where the normals are approximately that of the fitted planes.

Edge model: The edge model is a combination of how well the planes fit (flat model $e_f(d, \hat{n}, Q)$), the angle α between the planes, whether or not the points lie below the tangent plane (h), and how close d is to each plane. Let Q_1 be the group of points assigned to plane one, and similarly for Q_2. Let d_1 be the distance from d to plane one, divided by the average size of the neighborhood ($\frac{1}{k}\sum_i ||q_i - d||$), and similarly for d_2.

$$h = \sum \max 0, \frac{l_i}{\frac{1}{k}\sum_i ||q_i - d||} - 0.1 \tag{9}$$

$$e_s(d, \hat{n}, Q) = 0.25\frac{|Q_1|e_f(d, \hat{n_1}, Q_1) + |Q_2|e_f(d, \hat{n_2}, Q_2)}{|Q_1| + |Q_2|} \tag{10}$$

$$+0.75\left(\left(\frac{\alpha - \frac{\pi}{2}}{\frac{\pi}{2}}\right)^2 + \frac{d_1 + d_2}{2} + h\right) \tag{11}$$

where $|Q|$ is the number of points in the set Q. l_i is the signed distance of q_i to the tangent plane.

Corner model: The corner model is identical to the edge model, except we place the points in three groups, sum over three planes, and compare the normals pair-wise.

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, with a piece missing. 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 projected polygon, i.e., the one-ring is less than half a circle.
- Any wedge of the polygon is bigger than 0.8π.

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 at we return a boundary. Otherwise, we return the best shape model as usual.

6.1. Modifications to heuristic algorithm

The regular algorithm uses just three initial, orthogonal vectors. If, after trying these three vectors, all of the smooth models are marked as boundaries we do a more exhaustive search to see if there is a projection direction that results in no gaps. We try 12 normal directions, evenly distributed on the sphere; if any of these pass the no boundary test (items listed above) we add it as a candidate normal in the outer loop.

If we get a boundary model, or the best model score is greater than 0.1 we try both adjusting the neighborhood size down (minimum size 6) and up (maximum size 50) in increments of 25% of the current size. Shrinking the neighborhood is helpful where the surface passes close to another piece of surface, and growing the neighborhood helps with bridging sampling gaps. Provided the inter-surface distance is greater than the intra-sample distance, the shape model that comes from only one surface will have a better score than one that has samples from both surfaces.

7. Evaluation

We evaluated the algorithm in three ways: 1) known surfaces, 2) real data, 3) comparison to exhaustive search.

7.1. Known data sets

Our test suite consists of six shapes (flat, cylinder, sphere, saddle, edge, corner) cross five sampling patterns (grid, hexagonal grid, contours, random but evenly distributed, and random, see Figure 5). We can adjust the curvature of each shape, and add both tangential noise and normal noise to the data points. By changing which point is d we create a boundary case as well (see Figure 5). C++ and Matlab code to generate these shapes is provided. For our tests we used a neighborhood size of 24 (a 5 x 5 grid), with the contour pattern having 10×3 samples. For each test shape and grid pattern we generated 10 x 10 examples with tangential and normal noise each incrementing independently in steps of 0.1. Figures 6 and 7 summarize the results of these tests across all 3,000 examples. The graphs show the effects of increasing tangential (top) and normal (bottom) noise, broken out by shape type. Tangential noise is relative to the spacing of the data points; a value of 1 means the points can be moved up to 1/2 of the average spacing between the points. A value of 1 for the normal noise means the points can be displaced up to 1/10 of the width of the shape.

We used the test cases to validate the shape, one-ring, and normal behavior under increasing levels of noise for both the boundary and the interior case. Except for some extreme noise cases, the algorithm returned the expected shape. In high-noise cases (> 0.8 noise) the method had trouble distinguishing saddles from flat areas, ridges from edges, bowls from corners, and missed some boundary cases. For the grid and hexagonal sampling patterns with interior points we know that the LN should have eight and six points, respectively. The algorithm returned the correct one-ring neighborhood in these cases. For the remaining test cases, we have verified the neighborhoods by hand which is, unfortunately, a somewhat subjective evaluation. Some examples are shown in Figure 5.

Figures 6 and 7 (right-most) plot the accuracy of the recovered normal for our algorithm as the noise levels increase for each shape. The normal reconstruction is fairly insensitive to both increased sampling and function noise. Except for the boundary corner and saddle cases, the dot product of the expected and found normal is > 0.95. We compare our results to three

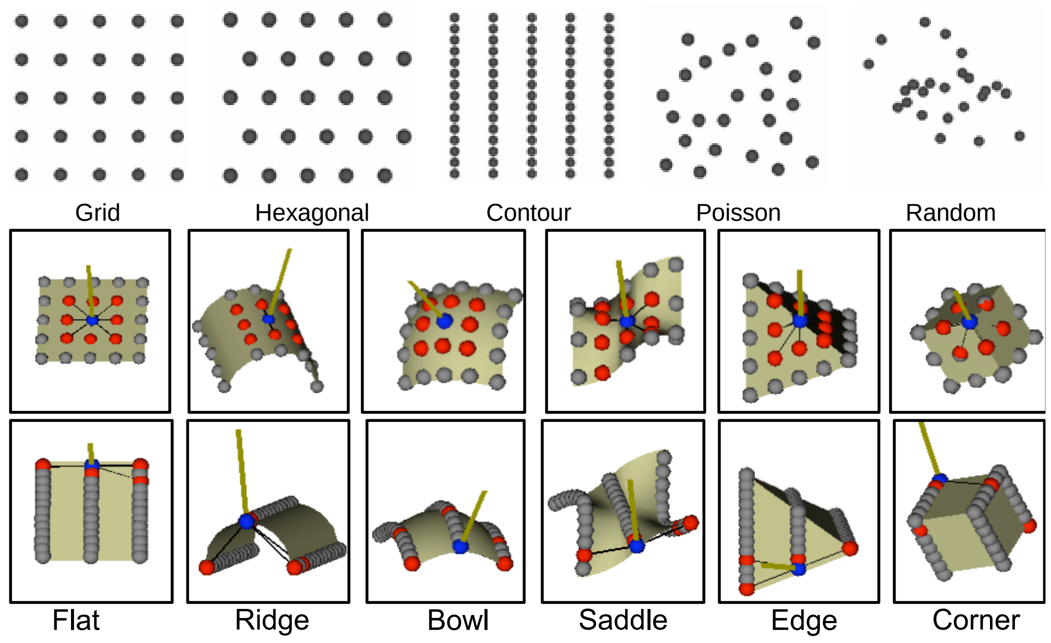


Figure 5: Top row: Sampling patterns. Test shapes with example local neighborhoods. Middle row: interior points. Bottom row: boundary points.

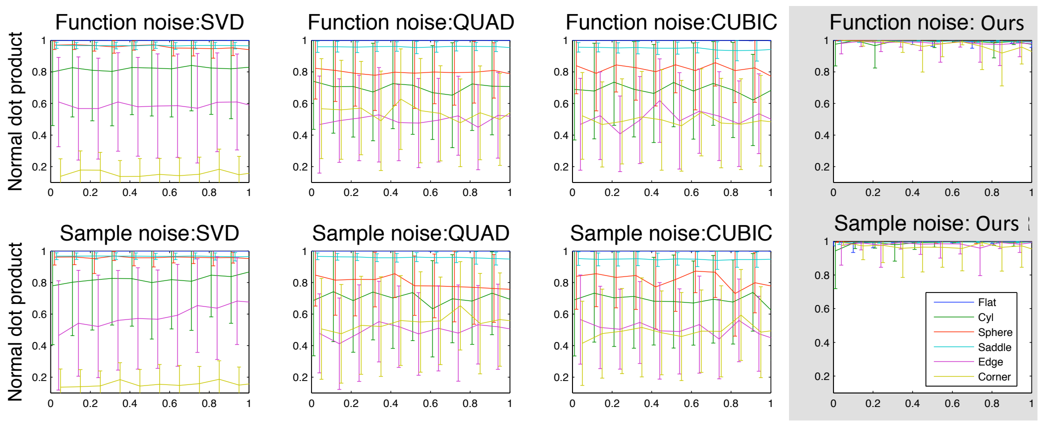


Figure 6: Behaviour of normals as function (top row) and sample (bottom row) noise is varied for (from left to right) SVD, quadratic, and cubic fitting, and our approach. Each curve corresponds to a specific test case shape (flat, ridge, bowl, saddle, edge, and corner). The data is averaged over the different sampling distributions (figure 5) 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.

approaches: plane-fitting normal reconstruction (Mitra and Nguyen, 2003) and quadratic and cubic fitting (Vančo and Brunnett, 2007). These three approaches perform well on flat regions, but deteriorate as the curvature increases, as the plots show. Note that adding sampling bias can also cause these methods to deteriorate, even when there is no noise (see figure 1). The only shape for which these algorithms have comparable results is the flat one. For all other shapes the average is worse, and the standard deviation higher.

There are several parameters in our heuristic algorithm (such as when a one-ring neighborhood should be considered a boundary). These parameters were set by experimentation with the complete test data set.

7.2. Real-data comparison

We verify the normal accuracy results observed in the test cases with real data sets. Figure 8 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 performance depends on the local shape. We compare with the (local) SVD (Mitra and Nguyen, 2003) and quadratic

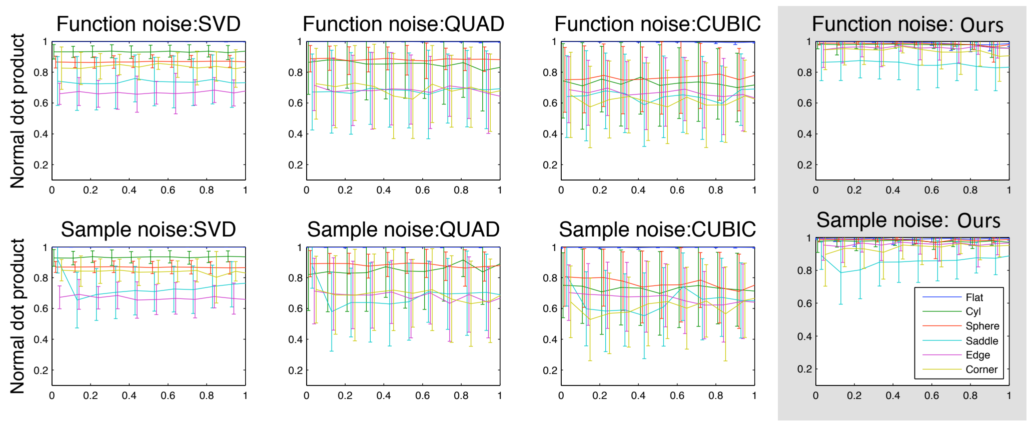


Figure 7: Same as figure 6, except for a point located on the boundary of the test shape.

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). The sampling of the dragon is much denser than the bunny, with a corresponding increase in accuracy for all algorithms. The "real" normals for the meshes were deliberately calculated using an area-weighted average, not the angle-weighted one we use, in an attempt to reduce bias (if we find the same one-ring in the mesh we would get an exact match otherwise). Note that, not only does our technique produce more accurate normals on average, but that the standard deviation is less, indicating fewer cases where the normal is substantially wrong.

Average and standard deviation values for several different types of models are given in table 1, corresponding images in Figure 9. Note that the first two data sets are problematic because there are many places where the surfaces are closer than the corresponding sampling rate. In general, our algorithm is the best, followed by the global algorithm. In the mug case, the global algorithm out-performs ours, although our algorithm does successfully identify where there are problems (see Figure 9). Although the global algorithm does use one-rings (essentially) to compute normals, the one-rings it creates must be consistent across the entire mesh; we do not have this restriction, which allows our algorithm to pick the "optimal" one-ring for each point. As in the test cases, all approaches do well when the local neighborhood is flat. It is when the surface curves that our approach is better.

	Bunny	Mug	Dragon	Gargoyle	Figure	Radius
Ours	**.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 1: Comparison to known normals. Bunny is a sparse (2002) down-sampling of the original data set, 2-holed mug is a contoured, 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 from a CT SCan. Given is the average and standard deviation of the dot product of the calculated normal with the known one. Best is in bold.

Figure 8 illustrates this by breaking out the normals by local shape type. Figure 10 shows this trend gets worse when noise is introduced.

Figure 10 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 edge length at each iteration. The comparison was to the normals from the original data set. As the graph in figure 10 shows, our approach out-performed both the SVD and the (global) Delaunay approach. This is because the average local shape, even with noise, remains similar, and the algorithm picks the one-ring (and hence surface normal) that best predict the local shape. This is particularly true for non-flat regions; we plot the reconstruction error separately for the two categories to illustrate this.

7.3. Optimization evaluation

The algorithm is heuristic and therefore not guaranteed to find the optimal solution. We implemented an exhaustive search algorithm for the test cases to compare to. In all cases the heuristic score was within 2% of the optimal, with a mean of 0.88%, variance 0.5%. This is largely because the normal "snaps" into place for most cases — the normal for all valid one-rings is very similar, even if their "quality" varies. The cases when the normal does not snap or there are more than one stable position are 1) high curvature areas, such as edge of the bunny ears, 2) the existence of multiple outliers, either because of noise or samples that come from a nearby surface, 3) highly

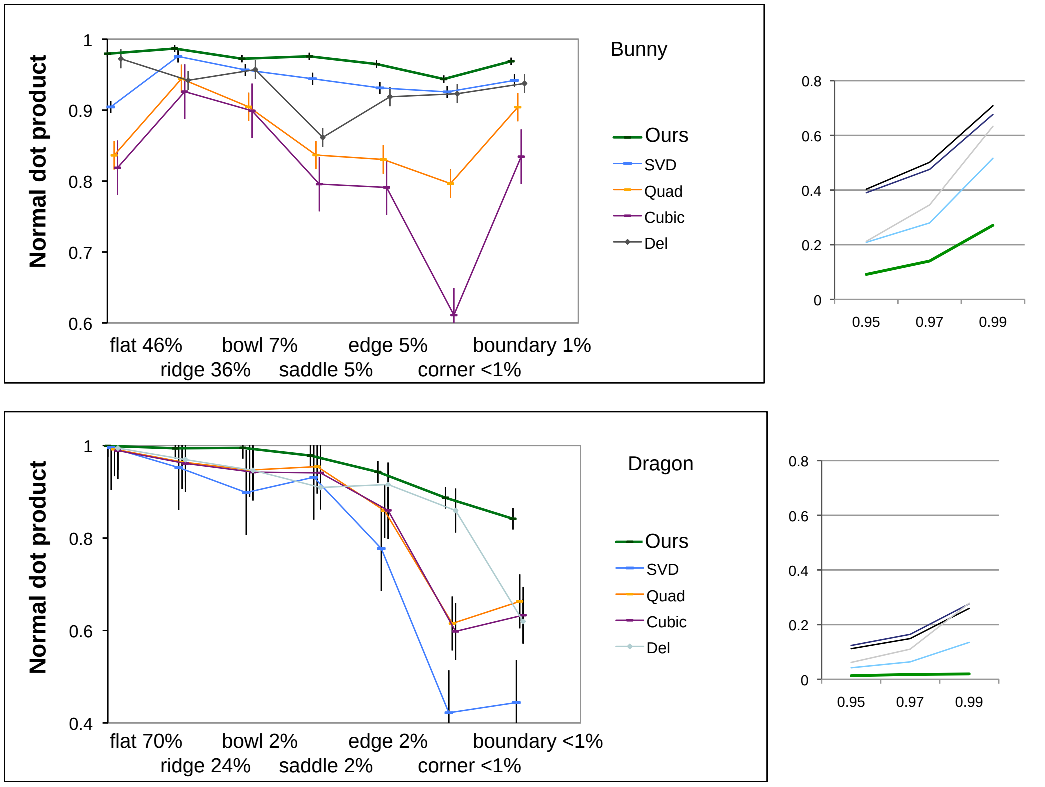


Figure 8: Comparison to SVD, quadratic, cubic, and a Delaunay approach on real data sets (Top – low-res bunny, Bottom – marching cubes dragon). Normals are grouped by shape classification (from left to right: Flat, ridge, bowl, saddle, edge, corner, boundary). Standard error bars are shown slightly separated for clarity. On the right are plots of the percentage of points for which the dot product is below 0.95, 0.97, and 0.99.

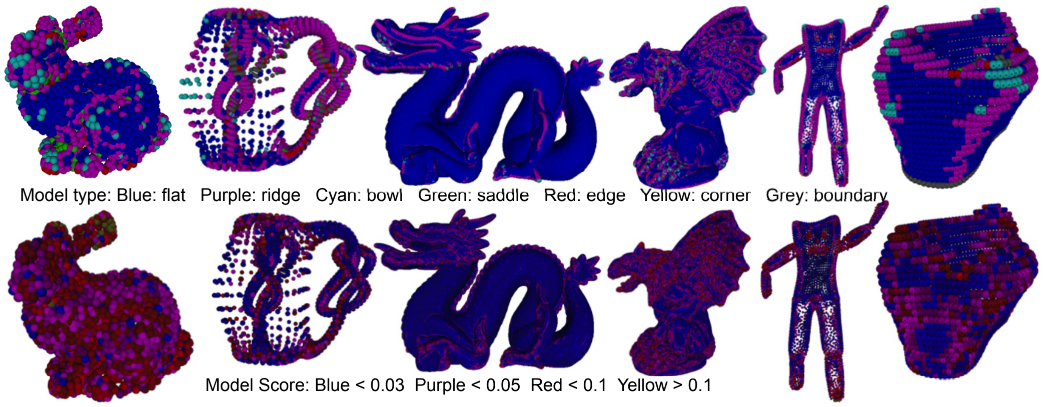

Model type: Blue: flat Purple: ridge Cyan: bowl Green: saddle Red: edge Yellow: corner Grey: boundary
Model Score: Blue < 0.03 Purple < 0.05 Red < 0.1 Yellow > 0.1

Figure 9: Model type and model score for the bunny, 2-holed mug, dragon, gargoyle, figure, and radius data sets.

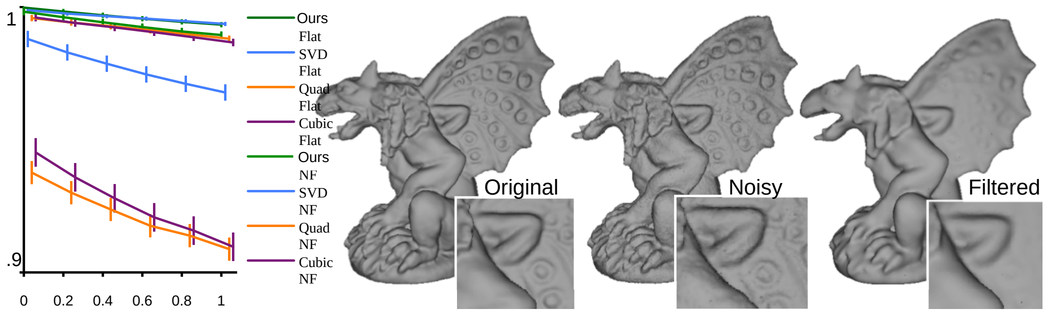


Figure 10: Behaviour under the addition of noise for our approach, SVD, Quad, and Cubic. Shown are average normal error for all data points split into flat (63%) and not flat (37%) sets. Model error as noise increases: 0.035, 0.042, 0.047, 0.051, 0.055, 0.058. Right: Surface reconstruction result for original, maximum noise, and filtered data sets.

Total	Model	Alg. logic	One-ring	Validation	Normal
17.6 σ 5.1	1.5 σ 2.3	3.2 σ 1.2	4.0 σ 1.1	4.3 σ 1.6	4.5 σ 4.0

Table 2: Running time, 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. Includes edge, corner, and boundary models. Platform: Pent-M 1.5GHz, 768 MB Ram.

uneven distribution of samples, such as boundaries or widely separated contours.

The exhaustive search algorithm is not practical; it takes minutes to run per point because it evaluates all $\sum_{i \in 4,k} \binom{k}{i}$ possible one-rings, which is exponential.

7.4. Timings

Running times for the heuristic algorithm, broken down by algorithm stages, are given in Table 2. The timings represent the total amount of time spent in that stage. Extracting the k-nearest neighbors for all data points is is $O(nlogn)$ using a *kd* tree (Arya et al., 1998). The running time per data point depends on the neighborhood size k and the number of candidate one-rings tried. Our algorithm employs a maximum cap on the number of candidates, so the running time is $O(nlogn + nk)$, as is the traditional fitting approach. Practically speaking, our algorithm is roughly a factor of 10 - 20 times slower than single-pass quadratic fitting for the fitting stage if we use all of the models.

8. Applications

We demonstrate using local neighborhoods for two applications, surface reconstruction and smoothing. Figure 11 shows several genus zero, manifold, water-tight mesh reconstructions from point clouds and figure 10 shows the result of applying simple isometric smoothing on the point sets before reconstruction. Many existing surface reconstruction algorithms perform better with good normal and noise or outlier estimates. We have used our normal estimation in a moving least squares reconstruction (Li et al., 2010b) and surface reconstruction by using Polymender (Ju, 2004).

8.1. Surface reconstruction via sphere embedding

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.

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.

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 10 shows the gargoyle model with Gaussian noise added in then filtered out (20 iterations of filtering).
Polymender reconstruction: The one-ring can be treated as a collection of triangles and sent to Polymender (Ju, 2004). All Polymender reconstructions (see Figure 11) were run with a depth of 8 and a 70% volume fill.

8.2. Feature size

We can vary k in order to get features at different scales. Figure 12 shows an example of this for the dragon and the fan-disk models. Feature size is an absolute measure (see section 5) based on how fast the surface curves away from the tangent plane.

9. Conclusion

We have presented a point-set analysis approach that uses a combination of local surface models and one-rings to produce robust surface normal estimation. The algorithm produces a local shape classification (including boundaries and sharp edges) and a measure of feature size. The algorithm

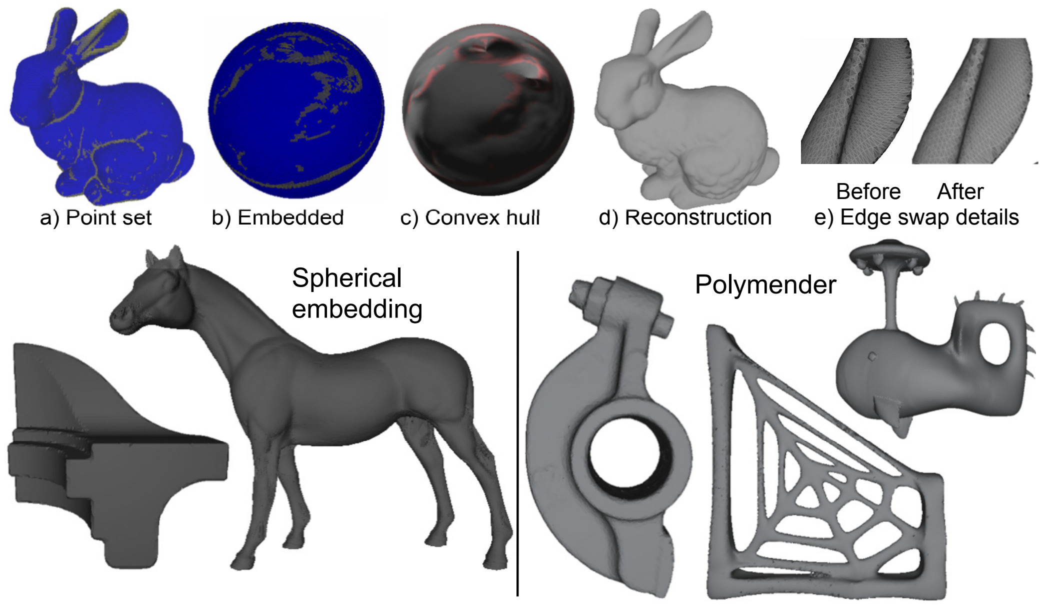


Figure 11: Top: Reconstructing a surface. From left to right: a) The initial point set, colored by feature size. b) The points embedded on the sphere. c) Tessellating the sphere using a convex hull algorithm. d) Moving the mesh vertices back to their original positions and performing edge swaps. e) Ear before and after edge swaps. Bottom, left: Horse and fan-disk model reconstructed using spherical embedding. Bottom, right: Rocker arm, fish, and spiderweb reconstructed using Polymender.

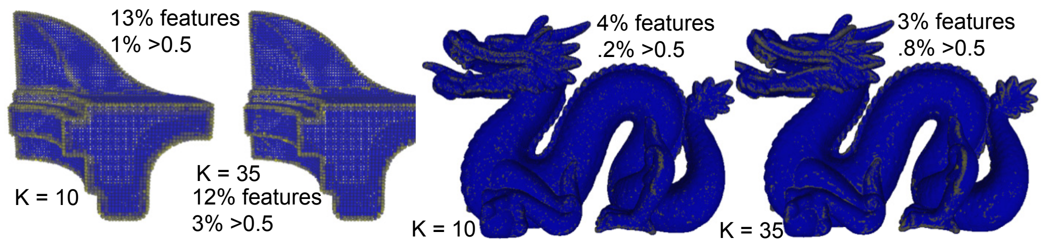


Figure 12: Using the same local neighborhoods, but increasing shape neighborhoods (10, 35), for labeling features. Blue is flat, yellow is a sharp fold.

also identifies places where the sampling is inadequate or too noisy to be sure of the reconstruction. Although specifically design to handle non-uniform sampling cases, it performs better on uniformly sampled data sets than previous algorithms.

Although this 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.

The approach was originally designed to deal with sparse or uneven sampling. In areas of dense sampling the one-rings will tend to have more neighbors (up to 32 in our implementation) but will still be regularly spaced. In particular, if there is a grid pattern to the samples then the one-rings will consist of 9 neighbors. Similarly, hexagonal patterns result in one-rings with 6 neighbors. In practice, we have rarely seen more than 12 neighbors, in part because our heuristic scoring system tends to favor a smaller number of neighbors, provided they are evenly spaced.

An alternative, global approach to constructing one-rings is to create a Delaunay triangulation of the surface. Both our approach and the Delaunay one try to produce reasonable triangles; the major difference is that we get to selectively choose which points to include in the one-ring and this choice does not have to be consistent across the point set. A potentially interesting area of future work would be to do a global optimization that reconciles the one-rings.

Although our approach is more computationally intensive than simple fitting, we believe that the extra cost is warranted in several cases: 1) Where accuracy is important. 2) The data samples are non-uniformly distributed (eg, contour data). 3) Further processing (such as consistent normal orientation) requires good estimates of the validity of the normal. 4) Extraction of sharp features or boundaries. To facilitate the usefulness of this approach we have provided a complete C++ implementation at `https://sourceforge.net/projects/meshprocessing/`.

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.

Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., Wu, A. Y., 1998. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. Journal of the ACM 45 (6), 891–923.
URL `citeseer.ist.psu.edu/arya94optimal.html`

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 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., 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.

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.

Grimm, C., Smart, W., 2008. Local neighborhoods for shape classification and normal estimation. Tech. Rep. CS-2008-15, Computer Science Department Washington University in St. Louis.

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

Klasing, K., Althoff, D., Wollherr, D., Buss, M., 2009. Comparison of surface normal estimation methods for range sensing applications. In: ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation. IEEE Press, Piscataway, NJ, USA, pp. 1977–1982.

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.

Li, B., Schnabel, R., Klein, R., Cheng, Z., Dang, G., Jin, S., 2010a. Robust normal estimation for point clouds with sharp features. Computers & Graphics 34 (2), 94 – 106.

Li, R., Liu, L., Phan, L., Abeysinghe, S., Grimm, C., Ju, T., 2010b. Polygonizing extremal surfaces with manifold guarantees. In: SPM '10: Proceedings of the 14th ACM Symposium on Solid and Physical Modeling. ACM, New York, NY, USA, pp. 189–194.

Meek, D. S., Walton, D. J., Feb. 2000. On surface normal and gaussian curvature approximations given data sampled from a smooth surface. Computer Aided Geometric Design 17, 521–543.

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.

OuYang, D., Feng, H.-Y., 2005. On the normal vector estimation for point cloud data from smooth surfaces. Computer-Aided Design 37 (10), 1071 – 1079.

Park, J. C., Shin, H., Choi, B. K., 2006. Elliptic gabriel graph for finding neighbors in a point set and its application to normal vector estimation. Comput. Aided Des. 38 (6), 619–626.

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.

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