Surface Reconstruction, One Triangle at a Time

Daniel Freedman*

Abstract

A new algorithm is presented for surface reconstruction from unorganized points. Unlike many existing methods, this algorithm does not select a subcomplex of the Delaunay Triangulation of the points. Instead, it uses an incremental technique, adding one triangle of the surface at a time. As a result, the algorithm does not require the surface's embedding space to be $\mathbb{R}^3$; the dimension of the embedding space may vary arbitrarily without substantially affecting the complexity of the algorithm. A wider variety of surfaces may therefore be reconstructed, including the class of non-orientable surfaces, such as the Klein Bottle. The algorithm is of an experimental character; while it is motivated from geometric and topological intuition, no proofs of homeomorphism are given. Instead, its efficacy is demonstrated from the point of view of correct experimental reconstruction of surfaces of varying genus.

1 Introduction

This paper treats the problem of surface reconstruction from unorganized points. A critical distinction from previous surface reconstruction algorithms is that the current method does not require the surface to be embedded in $\mathbb{R}^3$. Rather, the surface may be embedded in an arbitrary Hilbert space. As a result, we may argue that the current algorithm is more "topologically intrinsic," as it depends only on the manifold itself, and not the space in which the manifold lives. Practically speaking, the algorithm can reconstruct many surfaces which cannot be reconstructed using earlier algorithms; the most interesting of these are perhaps the non-orientable surfaces, such as the Klein Bottle and the real projective plane.

The algorithm is able to function in a high-dimensional embedding space because it is *incremental*. At each iteration, the surface is grown by adding a single triangle. A major issue related to such an incremental algorithm is the following: how can we can avoid choosing triangles that lead to surface self-intersections? We deal with this issue by performing all computations *locally*, in a plane which approximates nearby tangent spaces of the manifold. Within this plane, we can guarantee that a new triangle exists, which can be added without leading to self-intersections. Note that the emphasis of this paper is *experimental*; instead of proving the "correctness of a surface," the algorithms are motivated from the point of view of geometric and topological intuition. The proof, in this case, is in the pudding, and the results show that the algorithm works on a variety of surfaces, including those of codimension > 1.

*Rensselaer Polytechnic Institute, Department of Computer Science, Troy, NY, USA, 12180, freedman@cs.rpi.edu. Research supported in part by NSF grant IIS-0133144.

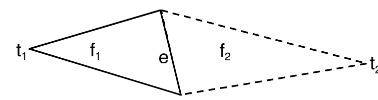

Figure 1: Adding the second coface. The notation in this figure will be used throughout the paper.

2 Related Work

The following survey of the literature on surface reconstruction is not exhaustive, due to space constraints. One of the earliest papers in the field was that of Boissonat [6], which presented a method of "sculpting" Delaunay Triangulations. The algorithm of Hoppe *et al.* [15] constructed a function whose zero level-set was the surface. Curless and Levoy [8] made improvements to this algorithm by taking advantage of the type of data that is produced by laser range scanners. The Ball-Pivoting Algorithm of Bernardini *et al* [5] is very fast in practice, but requires relative uniformity of sampling.

The crust of Amenta and Bern [2] provided the first algorithm which both allowed for highly non-uniform sampling and which provided theoretical guarantees. This algorithm was simplified in [3], as was the proof of homeomorphism. Finally, certain guarantees of the "watertightness" of a surface are given by the power-crust algorithm [4].

Dey and others have used a related notion, the "co-cone," in order to develop several new algorithms. Some of these extensions include: algorithms with better complexity [13]; algorithms which are implemented using data structures designed to handle large amounts of data [11]; and algorithms which allow for the detection of undersampling [9]. A more recent paper [10] uses the co-cone idea in order to determine the topological dimension of various objects, and presents a reconstruction scheme based on the computed dimension.

Other approaches include those of Adamy *et al* [1], Gopi *et al* [14], Boissonat and Cazals [7], and Edelsbrunner [12].

3 Incremental Reconstruction

In this section, we describe the incremental algorithm for surface reconstruction, which may be applied to any surface without boundary which is embedded in a Hilbert Space of dimension D. The algorithm is given in pseudocode in Figure 2; smaller algorithms, which are called by the main al-

```
RECONSTRUCT({x_i}_{i=1}^n)
  (e, f_1, t_1, K) = INITIALIZE({x_i}_{i=1}^n)
  do
    b = barycentre(e)
    (Γ, L) = FILTER({x_i}_{i=1}^n, t_1, b, e, K)
    {u_i}_{i∈Γ} = PROJECT({x_i}_{i=1}^n, e, t_1, b, Γ)
    d_min = ∞
    for all t_2 ∈ Γ
      f_2^pot = e ∪ {t_2}
      if ||x_{t_2} − b|| < d_min
        if INTERSECT(f_2^pot, L, {u_j}_{j∈Γ}) = False
          d_min = ||x_{t_2} − b||
          f_2 = f_2^pot
    K = K ∪ {σ : σ ⊂ f_2}
    draw (e, f_1, t_1) from K with #cofaces(e) = 1
  until there is no e
  return K
```

Figure 2: Incremental reconstruction.

gorithm, are specified in Figures 3, 4, and 5. The algorithm grows the surface by adding one simplex at a time. At each iteration, the algorithm chooses a "dangling edge" e from the current simplicial complex K: an edge which has only a single coface, f_1. Since the surface is assumed to be without boundary, every edge must have exactly two cofaces; the goal is then to find a second coface f_2 for this dangling edge. See Figure 1, which shows the notation to be used throughout the paper. When the new simplex f_2 is added, the dangling edge will no longer be dangling; however, the new simplex may both create new dangling edges, as well as destroy other existing dangling edges. The algorithm terminates when there are no more dangling edges; at this point, we have produced a 2-manifold without boundary.

The INITIALIZE routine given in Figure 2 generates the initial triangle f_1 (as well as other quantities). Due to space constraints, the INITIALIZE algorithm is not detailed here; however, one may imagine a variety of possible algorithms.

3.1 Filtering

```
FILTER({x_i}_{i=1}^n, t_1, b, e, K)
  ρ = (x_{e^1} − b) / ||x_{e^1} − b||
  ξ_1 = (x_{t_1} − b) − ⟨x_{t_1} − b, ρ⟩ρ
  Γ = ∅
  for i = 1 to n
    ξ_2 = (x_i − b) − ⟨x_i − b, ρ⟩ρ
    if angle(ξ_1, ξ_2) ≥ π − ε
      Γ = Γ ∪ {i}
  L = {σ ∈ K : vert{σ} ⊂ Γ}
  return (Γ, L)
```

Figure 3: Filtering of the simplicial complex K.

Before the second coface f_2 is added, we have a simplicial complex K. We would like to ensure that after adding f_2, we still maintain a simplicial complex; that is f_2 can only intersect the other simplices of K is the "natural" ways allowed for a simplicial complex. The only simplices in K which might intersect any potential second coface f_2 are those in the neighbourhood of the dangling edge e. As a result, we filter both the set of samples $\{1, \ldots, n\}$ and the simplicial complex K, to get a subset of samples of Γ and a subcomplex L which are in the neighbourhood of e.

We cannot straightforwardly use a distance criterion to detect the neighbourhood of e, as the points may be quite non-uniformly sampled. Instead, we use a tangent space based method. The idea is to approximate the tangent space of points near the dangling edge e by the plane P_1 running through e's first coface f_1. (Consult Figure 1 for notation.) Now, consider the vertex i; we may combine this vertex with the dangling edge e, to form a second plane P_2. The idea is that we consider i to lie in the neighbourhood of the dangling edge if the two planes P_1 and P_2 are almost coincident, that is, there is an angle between them of nearly π. The intuition behind this filter is straightforward: if the original surface M is smooth and well-sampled, there should be several points i around any dangling edge e such that the two planes P_1 and P_2 form an angle of greater than $\pi - \epsilon$. The size of ϵ reflects the fact that we are not infinitely well-sampled; we have found $\epsilon = \pi/4$ to be an excellent choice in practice, for the surfaces tested in this paper.

A question remains: how to compute the angle between the planes P_1 and P_2? This angle is computed as the angle between two vectors ξ_1 and ξ_2; ξ_i lies in plane P_i, and is perpendicular to the edge e. See Figure 3 for precise details.

Once the points have been filtered, it is a simple matter to filter the simplices. If K is the original simplicial complex, and Γ is the filtered set of vertices, then $L = \{\sigma \in K : \text{vert}\{\sigma\} \subset \Gamma\}$ is the filtered subcomplex. In other words, we retain exactly those simplices in which all of the vertices have passed the angle test. This begs the question: are there any simplices which are not in the neighbourhood of the dangling edge which are retained? The answer is yes. Such a simplex comes about if it lies, approximately, in the tangent space of the first coface f_1 of the dangling edge e. Note, however, that regardless of the location of such a simplex, it is distorted very little in the process of projection, to be discussed in the next section. As a result, there are two possibilities: (a) the simplex is in the neighbourhood of the dangling edge, in which case it is is crucial to the incremental triangulation procedure; (b) the simplex is not in the neighbourhood of the dangling edge, and is not distorted, in which case it does not affect the incremental triangulation procedure.

3.2 Projection

The ultimate goal is to choose a second coface f_2 for the dangling edge e. In order to do so, we would like to be able to perform "incremental triangulation" in the plane, which

PROJECT($\{x_i\}_{i=1}^n, e, t_1, b, \Gamma$)
  T = orthonormal basis for $\{x_{e^2} - x_{t_1}, x_{e^1} - x_{t_1}\}$
  for $i \in \Gamma$
    $u_i = \frac{\|x_i - b\|}{\|T^T(x_i - b)\|} T^T(x_i - b)$
  return $\{u_i\}_{i \in \Gamma}$

Figure 4: Projection onto the plane.

will be discussed in the next section. However, incremental triangulation is only possible if we situate the neighbourhood of e, as embodied in the filtered subcomplex L, in the plane. Thus, we project all points in Γ and simplices in L, retaining the same abstract simplicial (sub)complex, but obtaining a new geometric one.

Let T be an orthonormal basis for the plane running through the first coface f_1 of the dangling edge; T is a $D \times 2$ matrix. In this, case a straightforward orthogonal projection of a point x onto this plane is specified by $\tilde{u} = T^T(x - b)$, where b is the barycentre of the dangling edge. This particular choice makes the barycentre b the origin of the plane; this is an arbitrary, but sensible, choice given the crucial role of the dangling edge. Because of the filtering, all points lie near the plane onto which we are projecting; thus, the projection is close to an isometry. However, whatever local distortion is introduced may be lessened if we modify the projection slightly. In particular, we ensure that the projection of a given sample is the same distance away from the origin of the projected space (which corresponds to the projection of b) as the sample is from b in the embedding space. Thus, we have that $u = \frac{\|x-b\|}{\|T^T(x-b)\|} T^T(x - b)$. (Note that we need not worry about the case $T^T(x - b) = 0$. The reason is that we have filtered all points for which such a relation is possible.)

3.3 Incremental Triangulation

INTERSECT($f_2^{pot}, L, \{u_j\}_{j \in \Gamma}$)
  test = False
  for all $\sigma \in L$ and while test = False
    solve the following linear program:

    $\max_{\alpha,\beta} \theta = \sum_{i \in f_2^{pot} - \sigma} \alpha_i$

    subject to: $\sum_{i \in f_2^{pot}} \alpha_i u_i = \sum_{j \in \sigma} \beta_j u_j$
      $\sum_{i \in f_2^{pot}} \alpha_i = 1, \quad \sum_{j \in \sigma} \beta_j = 1$
      $\alpha_i \geq 0, \quad \beta_j \geq 0$

    if there is a feasible solution with $\theta^* > 0$
      test = True
  return test

Figure 5: Determining whether or not the face f_2^{pot} intersects the subcomplex L.

We may now turn the task of determining the second co-

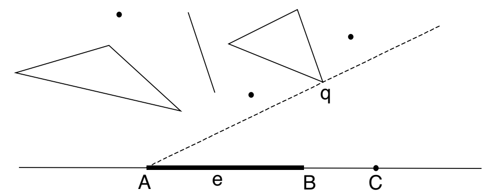

Figure 7: An illustration of the proof that there is always a second coface f_2; see text. The ray is the dotted line.

face f_2 of the dangling edge e. In particular, we must find a vertex t_2 in the filtered set of vertices Γ; f_2 is then the 2-simplex whose vertices are the vertices of e combined with t_2. Now, we may form the abstract collection of subsets $\hat{L} = L \cup \{\sigma : \sigma \subset f_2\}$; furthermore, we may form the collection of subsets in the plane $\hat{G}$ by mapping each vertex i in $\hat{L}$ to its corresponding projected point u_i. The key property for any f_2 to satisfy is that $\hat{G}$ must be a geometric simplicial complex. In other words, f_2 must only intersect the existing simplices in the neighbourhood of the dangling edge e in the natural ways which leave the entire ensemble a simplicial complex.

Our first task must be to determine whether such a coface f_2 always exists. Fortunately, this is the case, as can easily be demonstrated.

Theorem: If L contains at least one vertex which does not lie on the line through e, then there exists a second coface f_2, which, when added to L, leaves L a simplicial complex.

Proof: Label the two vertices of the edge A and B, as shown in Figure 7. Take a ray whose endpoint is A, and which lies along the edge AB. Rotate this ray upwards into the halfplane which does not contain f_1, as shown in Figure 7; note that based on the filtering procedure, there will be no vertices lying in the halfplane containing f_1. Stop rotating the ray when it first intersects a simplex. Note that it is possible that the ray initially intersects some vertices, when the angle of rotation is 0, such as point C in Figure 7; however, we only look for intersections once the angle of rotation is greater than 0. There are two possibilities. (1) The ray intersects a vertex. This vertex may either be isolated, or it may belong to an edge or triangle. In any of these cases, it is easy to see that we may use this vertex as t_2, the third vertex of the second coface f_2 of the dangling edge e. To see this, assume to the contrary, that this vertex will not give us a proper f_2. In this case, we must have that another simplex intersects the interior of f_2. However, this is impossible, as the interior of f_2 lies below the ray; and we know that there are no simplices below the ray, by construction. This yields the desired contradiction. (2) The ray intersects an edge. The edge may be isolated, or it may be a face of a triangle. In either case, we may use the vertex of the edge which is closer to the dangling edge e as t_2. The argument is precisely the same as in the first case. In the case that the ray intersects more than one simplex simultaneously, we may simply take the vertex which is closest to the dangling edge as t_2, and the previous

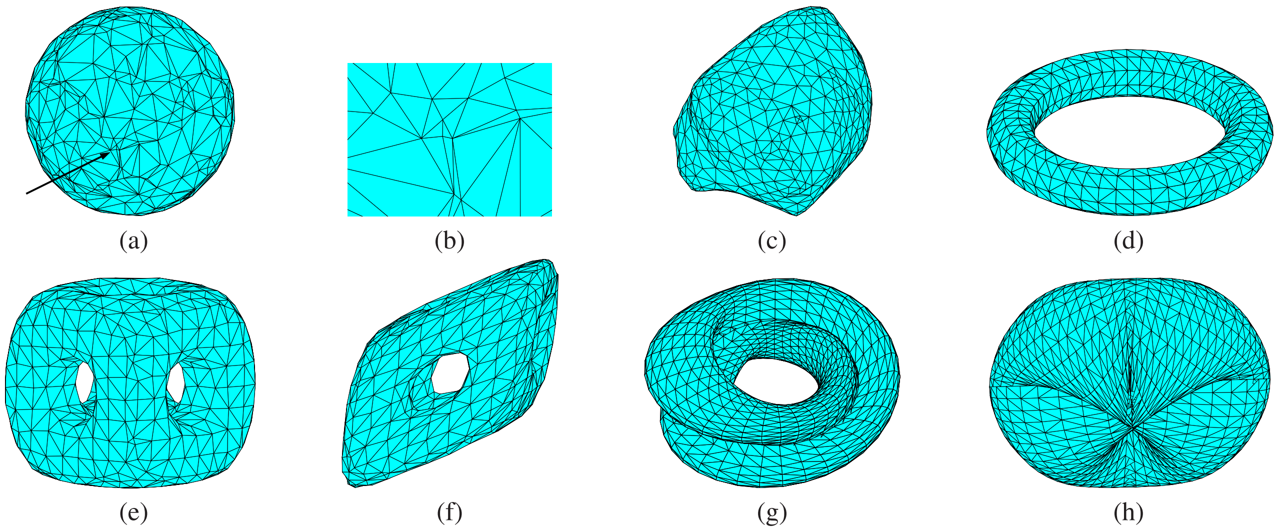

Figure 6: Results on various surfaces; see text for more details.

arguments carry through. □

Contained in this proof is in fact an algorithm. However, we do not use this algorithm, as the triangle f_2 picked out by this algorithm may not be "geometrically nice." For example, it could be a very long, skinny triangle which is almost parallel to the dangling edge. Thus, we find all possible triangles which are valid, and choose the one which is best. There are a variety of criteria we could use to designate the best triangle, but we choose a natural one: the proximity of third vertex t_2 to the barycentre b of the dangling edge.

Thus, the problem reduces to finding an algorithm for determining whether two simplices, σ (any simplex in L) and f_2^{pot} (a potential second coface) intersect in a "problematic" way. This algorithm can be designed as a linear program, shown in Figure 5. Here, the variables are $\alpha = \{\alpha_i\}_{i \in f_2^{pot}}$ and $\beta = \{\beta_j\}_{j \in \sigma}$; these are the barycentric coordinates of f_2^{pot} and σ, respectively. The three constraints ensure that the two simplices intersect. The objective function will be positive if the two simplices intersect, but do not intersect in a common face; if they intersect in a common face it will be 0. As a result, we simply test if the linear program has a feasible solution with a positive value for the objective function ($\theta^* > 0$). If this is the case, then the simplices intersect in a problematic way, and f_2^{pot} is not a valid second coface.

3.4 Complexity

Let F be the number of faces of the surface, E the number of edges, and n the number of sample points. Then the complexity of the algorithm can be analyzed directly from Figure 2. (a) The main loop runs F times. (b) The main loop of FILTER is $O(n)$, and the computation of L is $O(n+E+F)$. (c) PROJECT is $O(|\Gamma|)$. (d) The scheme to determine simplex-simplex intersection is $O(1)$, as the dimension of the linear program is a constant, unrelated to n, E, or F. As a result, INTERSECT is $O(|L|)$, and the inner loop ("for all $t_2 \in \Gamma$") is $O(|\Gamma||L|)$. (e) Drawing a new dangling edge from K takes $O(|K|) = O(n + E + F)$. (f) While not described explicitly, the procedure to find the first simplex is $O(n)$. Thus, the total complexity is $O(F(n + E + F + |\Gamma||L|))$.

We can simplify this expression, based in part on the fact that K is a manifold. This implies that there are exactly two cofaces for each edge; since each triangle has three edges, we must have that the number of edges is $E = 3F/2$. Furthermore, we have that the Euler Characteristic is given by $\chi = n-E+F = n-F/2$; assuming that the genus is a constant independent of the number of samples n, then so is χ (which depends linearly on the genus), and thus $F = O(n)$. This allows us to reduce the complexity to $O(n^2 + n|\Gamma||L|)$. It is possible that $|\Gamma| = |L| = O(n)$; this can happen if there is a very large flat area of the manifold. In this case, the complexity is $O(n^3)$. In more typical cases, however, $|\Gamma| = |L| = O(1)$; this occurs when the neighbourhoods are much more localized. In this case, the complexity is $O(n^2)$.

The key is that the exponent of n is independent of D, the dimension of the embedding space. This comes about because we do not use a partition of the entire space (such as a Voronoi Diagram) to reconstruct the manifold; the incremental algorithm works the same way in any space.

3.5 Results

Results of running the algorithm on surfaces of varying genus are shown in Figure 6. (a) demonstrates reconstruction of a sphere; (b) shows the highly non-uniform sampling of this sphere. (c) shows the reconstruction of a surface which is a topological sphere; in this case, the sampling non-uniformity arises from the fact that the marching cubes algorithm was used to extract the surface from an implicit function representation. (Marching cubes is known to generate "ugly triangles.") (d) and (e) illustrate correct results when running on surfaces with non-spherical topology; (d) is a torus, while (e) has genus 5.

Thus far, we have seen surfaces which are "pedestrian" by the standards of surface reconstruction algorithms. Despite the reasonably complex topology of the last example, we would expect any of a variety of the algorithms discussed in Section 2 to be able to handle these surfaces. We would not, however, expect these algorithms to work on the surfaces shown in Figures (f), (g), or (h); all of these surfaces are of codimension > 1. (f) shows the same surface as in (e), but which has been embedded in $\mathbb{R}^{40}$ through an isometry. The surface looks sheared, due to the fact that it has been sliced along three dimensions; projection along these dimensions does not preserve angles, despite the fact that the overall transformation is an isometry. (g) and (h) show surfaces which cannot be embedded in $\mathbb{R}^3$, as they are non-orientable. Both are examples of Klein Bottles; (g) shows the classic "figure-8" Klein Bottle, while (h) is a "flat" (zero Gaussian curvature) Klein Bottle. Both have been embedded in $\mathbb{R}^5$, and are shown along three particular dimensions.

References

[1] U. Adamy, J. Giesen, and M. John. New techniques for topologically correct surface reconstruction. In *Proc. of the 11th Ann. IEEE Visualization Conference*, pages 373–380, 2000.

[2] N. Amenta and M. Bern. Surface reconstruction by voronoi filtering. *Disc. and Comp. Geom.*, 22:481–504, 1999.

[3] N. Amenta, S. Choi, T. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. In *Proc. 16th ACM Sympos. Comput. Geom.*, pages 213–222, 2000.

[4] N. Amenta, S. Choi, and R. Kolluri. The power crust, union of balls, and the medial axis transform. *Computational Geometry – Theory and Applications*, 19:127–153, 2001.

[5] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Trans. Vis.*, 5(4):349–359, 1999.

[6] J. Boissonnat. Geometric structures for three-dimensional shape reconstruction. *ACM Trans. Graph.*, 3:266–286, 1984.

[7] J. Boissonnat and F. Cazals. Smooth surface reconstruction via natural neighbour interpolation of distance functions. *Comp. Geom. – Theory and Appl.*, 22(1):185–203, 2002.

[8] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proc. SIGGRAPH '96*, pages 303–312, 1996.

[9] T. Dey and J. Giesen. Detecting undersampling in surface reconstruction. In *Proc. 17th ACM Sympos. Comput. Geom.*, pages 257–263, 2001.

[10] T. Dey, J. Giesen, S. Goswami, and W. Zhao. Shape dimension and approximation from samples. In *Proc. 13th ACM-SIAM Sympos. Discrete Algorithms*, pages 772–780, 2002.

[11] T. Dey, J. Giesen, and J. Hudson. Delaunay based shape reconstruction from large data. In *Proc. IEEE Symp. Parallel & Large Data Vis. Graph.*, pages 19–27, 2001.

[12] H. Edelsbrunner. Surface reconstruction by wrapping finite point sets in space. In B. Aronov, S. Basu, J. Pach, and M. Sharir, editors, *Ricky Pollack and Eli Goodman Festschrift*. Springer-Verlag, 2002. to appear.

[13] S. Funke and E. Ramos. Smooth-surface reconstruction in near-linear time. In *Proc. 13th ACM-SIAM Sympos. Discrete Algorithms*, pages 781–790, 2002.

[14] M. Gopi, S. Krishnan, and C. Silva. Surface reconstruction based on lower dimensional localized delaunay triangulation. *Eurographics*, 19(3):C467–C478, 2000.

[15] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Proc. SIGGRAPH '92*, pages 71–78, 1992.