A Fast and Efficient Projection-Based Approach for Surface Reconstruction

Category: Research Paper (Number: papers_1364)

Abstract

We present a fast, memory efficient, linear time algorithm that generates a manifold triangular mesh S passing through a set of unorganized points $P \subset \mathcal{R}^3$. Nothing is assumed about the geometry, topology or presence of boundaries in the data set except that P is sampled from a real manifold surface. The speed of our algorithm is derived from a projection-based approach we use to determine the incident faces on a point. Our algorithm has successfully reconstructed the surfaces of unorganized point clouds of sizes varying from 10,000 to 100,000 in about 3–30 seconds on a 250 MHz, R10000 SGI Onyx2. Our technique is especially suitable for height fields like terrain and range scan data even in the presence of noise. We have successfully generated meshes for scan data of size 900,000 points in less than 40 seconds.

CR Categories: 1.3.5 [Computer Graphics]: Computational Geometry and Object Modeling.

Keywords: Geometric Modeling, Surface Reconstruction and Fitting, Three-Dimesional Shape Recovery, Range Data Analysis.

1 Introduction

The problem of surface reconstruction from unorganized point clouds has been, and continues to be, an important topic of research. The problem can be loosely stated as follows: *Given a set of points P which are sampled from a surface in $\mathcal{R}^3$, construct a surface S so that the points of P lie on S.* A variation of this *interpolatory* definition is when S *approximates* the set of points P.

There are a wide range of applications for which surface reconstruction is important. For example, scanning complex 3D shapes like objects, rooms and landscapes with tactile, optical or ultrasonic sensors are a rich source of data for a number of analysis and exploratory problems. Surface representations are a natural choice because of their applicability in rendering applications and surface-based visualizations (like information-coded textures on surfaces). The challenge for surface reconstruction algorithms is to find methods which cover a wide variety of shapes. We briefly discuss some of the issues involved in surface reconstruction.

We assume in this paper that the inputs to the surface reconstruction algorithm are sampled from an actual surface (or groups of surfaces). A proper reconstruction of these surfaces is possible only if they are "sufficiently" sampled. However, sufficiency conditions like sampling theorems are fairly difficult to formulate and as a result, most of the existing reconstruction algorithms ignore this aspect of the problem. Exceptions include the works of [3, 7, 2].

If the surface is improperly sampled, the reconstruction algorithm can produce artifacts. A common artifact is the presence of spurious surface boundaries in the model. Manual intervention or additional information about the sampled surface (for instance, that the surface is manifold without boundaries) are possible ways to eliminate these artifacts. The other extreme in the sampling problem is that the surface is sampled unnecessarily dense. This case occurs when a uniformly sampled model with a few fine details can cause too many data points in areas of low curvature variation.

Sometimes the data might contain some additional information on the structure of the underlying surface. Laser scanners that generate samples uniformly on a sphere (or a cylinder, depending on its degrees of freedom) are typical examples. In this case, it is known that adjacent data points have a very high probability of being adjacent to each other in the final mesh. We refer to these data sets as *organized point clouds*. This information can be exploited by some algorithms, including ours, to give quick results.

Another issue in surface reconstruction is the presence of noise and outliers in the original data. The mode of data acquisition has a direct impact on this. For example, range scan data can be very noisy when the surface is not oriented transverse to the scanning beam. Noisy data introduce high frequency artifacts in the reconstructed surface (like micro-facets) and this is a cause of concern for many algorithms.

The choice of underlying mathematical and data structural representation of the derived surface is also important. The most common choice are triangular or polygonal mesh representations. Triangular meshes also allow us to express the topological properties of the surface, and it is the most popular model representation for visualization and rendering applications.

Finally, the recent thrust in research to build augmented reality and telepresence applications has introduced an interesting variation of the surface reconstruction problem. Consider an application where multiple cameras or camera-projector pairs are used to extract the geometry of dynamic scenes at interactive rates [23]. In this scenario, the surface reconstruction algorithm should be able to handle extremely large data sets (order of many millions of points) and provide a suitable surface representation without significant latency. One of the main motivations for this work is to develop an approach that handles both *organized* and *unorganized* point clouds very efficiently in time and memory requirements.

1.1 Main Contributions

In this paper, we present a fast and efficient projection-based algorithm for surface reconstruction from unorganized point clouds. Our algorithm incrementally develops an interpolatory surface using the surface oriented properties of the given data points. The main contributions of this paper include:

- **Linear time performance:** Each iteration of our algorithm advances the reconstructed surface boundary by choosing one point on it and completes all the faces incident on it in constant time. The constant of proportionality is very small.
- **Speed:** We have tested our algorithm on a number of data sets ranging from 10,000 to 100,000 unorganized points. It takes about 3–30 seconds to reconstruct the mesh. We have also tested our algorithm on an organized point cloud of size 6.5 million. After simplifying this data to around 900,000 points, it took us about 40 seconds to generate the mesh on a 250 MHz, R10000 SGI Onyx2 with 16 GB of main memory.
- **Memory efficiency:** Our algorithm has minimal memory overhead because it goes through a single pass of all data points to generate the mesh. We do not maintain the computed triangles in our data structure because our method does not revisit them. Only the input data has to be stored.

- **Robustness:** In the special case of terrain data or data from common center-of-projection scanning devices, our algorithm can tolerate high noise levels. The error introduced by noise has to be bounded, however.

2 Previous Work

The problem of surface reconstruction has received significant attention from researchers in computational geometry and computer graphics. In this section, we give a brief survey of existing reconstruction algorithms. We use a classification scheme by Mencl et. al. [21] to categorize the various methods. The main classes of reconstruction algorithms are based on *spatial subdivision*, *distance functions*, *surface warping* and *incremental surface growing*.

The common theme in spatial subdivision techniques is that a bounding volume around the input data set is subdivided into disjoint cells. The goal of these algorithms is to find cells related to the shape of the point set. The cell selection scheme can be surface-based or volume-based.

The surface-based scheme proceeds by decomposing the space into cells, finding the cells that are traversed by the surface and finding the surface from the selected cells. The approaches of [17, 11, 4, 3] fall under this category. The differences in their methods lie in the cell selection strategy. Hoppe et. al. [17, 18, 16] use a signed distance function of the surface from any point to determine the selected cells. Bajaj and Bernardini [4, 5, 6] construct an approximate surface using α-solids to determine the signed distance function. Edelsbrunner and Mucke [22, 11] introduce the notion of α-shapes, a parameterized construction that associates a polyhedral shape with a set of points. The choice of α has to be determined experimentally. More recently, Guo et. al. [13] use visibility algorithms and Teichmann et. al. [25] use density scaling and anisotropic shaping to improve the results of reconstruction using α-shapes. For the two-dimensional case, Attali [3] introduces *normalized meshes* to give bounds on the sampling density within which the topology of the original curve is preserved.

The volume-based scheme decomposes the space into cells, removes those cells that are not in the volume bounded by the sampled surface and creates the surface from the selected cells. Most algorithms in this category are based on Delaunay triangulation of the input points. The earliest of these approaches is Boissonat's [9] "Delaunay sculpting" algorithm that successively removes tetrahedra based on their circumspheres. Veltkamp [27] uses a parameter called γ-indicator to determine the sequence of tetrahedra to be removed. The advantage of this algorithm is that the γ-indicator value adapts to variable point density. However, both the approaches of Boissonat and Veltkamp cannot handle objects with holes and surface boundaries. Amenta et. al. [2, 1] use a Voronoi filtering approach based on three-dimensional Voronoi diagram and Delaunay triangulation to construct the *crust* of the sample points. They provide theoretical guarantees on the topology of their reconstructed mesh given "good" sampling.

The distance function of a surface gives the shortest distance from any point to the surface. The surface passes through the zeroes of this function. This approach leads to approximating instead of interpolatory surfaces [17, 16, 8, 10]. Hoppe et. al. [17, 16] use a Reimannian graph to compute consistent normal throughout the surface to determine the signed distance function. The approach of Curless and Levoy [10] is fine-tuned for laser range data. Their algorithm is well suited for handling very large data sets.

Warping-based reconstruction methods deform an initial surface to give a good approximation of the input point set. This method is particularly suited if a rough approximation of the desired shape is already known. Terzopoulos et. al. [26] use *deformable superquadrics* to fit the input data points. Szeliski et. al. [24] model the interaction between *oriented particles* to construct their surface.

The basic idea behind incremental surface construction is to build-up the surface using surface-oriented properties of the input data points. Boissonnat's surface contouring algorithm [9] starts with an edge and iteratively attaches further triangles at boundary edges of the emerging surface using a projection-based approach. This algorithm is similar in vein to our approach. A crucial difference between our methods is that Boissonnat's algorithm is edge-based, while ours is vertex-based. Further, his algorithm can only generate manifolds without boundaries. The approach of Mencl and Muller [19, 20] is to start with a global wireframe of the surface generated using Euclidean minimum spanning tree construction, and to fill it iteratively to complete the surface.

3 Algorithm Overview

Our algorithm starts at a data point, and finds all its incident triangles. Then each of its adjacent vertices in the boundary of the triangulation is processed in a breadth-first fashion to find their other incident triangles. Thus the boundary of the completed triangulation propagates on the surface of the point cloud till it processes all the data points. In the rest of the paper, we refer to the point being processed as the *reference point*, R.

There are a few assumptions we make about the data set. The sampling of the data is *locally uniform*, which means that the distance ratio of the farthest and closest neighbor of a point in the final triangulation is less than a constant value. The second assumption is that the model is *locally smooth*. The quantitative measure of this assumption is that all the angles between the normals of the faces incident on a vertex in the original surface are within 90° of each other. The final assumption is to distinguish points from two close layers of the model. The closest distance between a point P in one layer and another layer is at least μm, where μ is a constant and m is the shortest distance between P and another point in its layer.

Our algorithm can be broadly divided into three stages: *bucketing*, *point pruning*, and finally the *triangulation step*.

Bucketing: In this stage, the data structure is initialized with the input data. Our data structure is a depth pixel array similar to the *dexel* structure [15]. We maintain a 2D pixel array into which all data points are orthographically projected. The points mapped on to the same pixel are sorted by their depth (z) values.

Point Pruning: We first apply a *distance criterion* to prune down our search for candidate adjacent points in the spatial proximity of R. It is applied in two stages. Our algorithm takes an axis-aligned box of appropriate dimensions centered at R and returns all the data points inside it. By using our data structure, this search is limited to the pixels around the pixel where R is projected. The second stage of pruning uses an Euclidean metric, which further rejects the points that lie outside a *sphere of influence* centered at R. The choice of the box dimensions and the radius of the sphere are described in the next section. The points chosen by this second level (C_R) are called the *candidate points* of R.

Visibility Criterion: Next, we estimate the tangent plane at R, and project R, C_R, and the mesh boundary in their vicinity on this tangent plane. Points in C_R that are occluded from R in the projection are removed.

Angle Criterion: This is an optional step, which tries to remove "skinny" triangles at R, to improve the quality of triangulation.

Triangulation: Finally, the remaining points in C_R are then connected in order around R to complete the triangulation (see video).

4 Surface Reconstruction

In this section, we describe our approach to surface reconstruction in detail. The output of our algorithm is an interpolatory, non-self-intersecting triangular mesh of the given point cloud.

The implicit function theorem [12] of smooth surfaces forms the basis of our approach. Without loss of generality, it states: *"Given an implicit surface $S \equiv f(x, y, z) = 0$, and a point P on it, such* *that the tangent plane to S at P is parallel to the (x, y) plane, then S in the neighborhood of P can be considered as a height function $f(x, y, h(x, y)) = 0$, a local parameterization on its tangent plane".*

Our algorithm is a greedy method and works with two parameters: μ, which quantifies our definition of *locally uniform sampling*, and α, which gives a lower bound on the angle between consecutive neighbors of a point on a boundary of the surface. In order to improve the quality of triangulation, we can optionally specify a minimum angle parameter, β. It is not necessary for the completion of our algorithm, though.

Terminology: We categorize the data points at any given stage of our algorithm as *free*, *fringe*, *boundary* and *completed* points. The *free* points are those which have no incident triangles. The *completed* points have all their incident triangles determined. Points that lie along the current surface boundary are either *fringe* or *boundary* points. *Boundary* points are those points which have been chosen as a reference point but have some missing triangles due to the maximum allowable angle parameter α. *Fringe* points have not yet been chosen as a reference point.

We maintain two invariants during our algorithm's execution:

Invariant 1: No *free*, *fringe* or *boundary* point can be in the interior of a triangle (because of our distance criterion).

Invariant 2: At the end of each iteration, the point chosen as the reference point becomes a *completed* or a *boundary* point. This is used later to prove claims about occluded points (for visibility criterion).

Our algorithm starts with the bucketing step by orthographically projecting the data points onto the *dexel* data structure. The following steps are used to choose the right set of points to be connected to the reference point R.

4.1 Point Pruning

Pruning by Distance Criterion: Points far away from the reference point R are not adjacent to it. We eliminate them by applying the distance criterion in two stages. Initially, we employ the cheaper L_∞ metric to narrow down our search. It is performed by constructing an axis-aligned box of suitable dimension around R and choosing all the *free*, *fringe* and *boundary* points inside the box. The use of our data structure (and bucketing) makes this step a constant time operation.

The dimension of the box is derived from μ as follows. In a general case, R (a *fringe* point) already has a few incident triangles. Let m be the minimum distance from R to its existing adjacent vertices. From our definition of *locally uniform sampling*, the farthest neighbor of R can be at most μm away. This gives the lower bound, and an estimate on the dimension of the box. When R has no incident triangles (for example, at the very beginning), we find the closest point to R using the *dexel* array representation and find m.

We call a sphere of radius μm centered at R as the *sphere of influence* (S_R) around R. The second stage of pruning uses a stricter L_2 metric and returns all points inside S_R. These points are the *candidate points* (C_R) of R. We would like to make an observation about the candidate point set. The radius of S_R is dependent on m, which changes from one vertex to another. Therefore, it is possible that a vertex p might be in the *sphere of influence* of R, but not vice-versa. But this asymmetry does not affect the topology of the reconstructed mesh.

Choice of Projection Plane: Typically, given a triangulated mesh, the triangulation around a point implicitly defines an ordering of its adjacent vertices on a projection plane. In order to complete the triangulation around a reference point R, our algorithm finds this ordering directly by projecting a selected neighborhood around R on a plane. The choice of the projection plane is an important issue, and dictates the robustness of our approach. According to the implicit function theorem, the best projection plane would be the tangent plane at R.

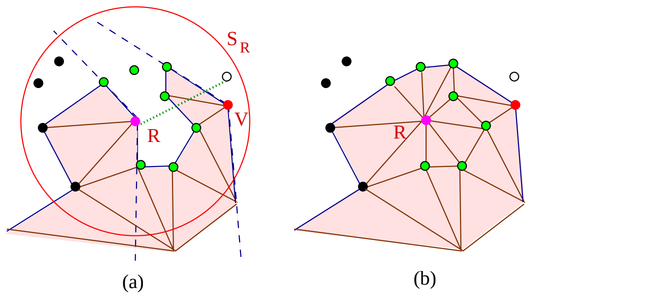


Figure 1: (a) Visibility test around R. The black points are behind R's boundary edges, the white points are occluded by other edges, and the point V is eliminated as R is behind its boundary edges. (b) Completed mesh at R

In the general case, R is a *fringe* point and the estimate of its tangent plane can be made on the basis of the triangles already incident on R. This is our choice for the projection plane (P_R). The ordering of the *candidate* points (C_R) around R in this plane will be incorrect if and only if there is a triangle incident on R with its normal deviating by more than 90° from the projection plane normal. This condition does not arise, however, because of our assumption about the *surface smoothness*. When R does not have any incident triangles, we fit an oriented bounding box around C_R to get an estimate of the orientation of the surface, and hence the projection plane.

Pruning by Visibility: The next stage of pruning is based on visibility in the plane P_R. It eliminates the points which potentially form a self-intersecting mesh. We define the *boundary edges* of a point as the set of edges incident on that point that lie on the current surface boundary. We project R, C_R, and their *boundary edges* on the plane P_R (see Color Plate 2 - middle row and video). If the line of sight from R to a candidate vertex is obstructed by any edge, then that point is an occluded point. The existence of visibility between these points in the plane is a sufficient but not a necessary condition for the visibility between them in the object space. In the limit, when the local surface approaches the tangent plane in a densely sampled point cloud, it becomes a necessary condition as well. We take a conservative approach and prune all the points in C_R which are occluded from R on P_R.

Points occluded from R are determined as follows. Initially, we order the points in C_R by angle around R in the plane P_R.

1. All the points between consecutive *boundary edges* of R (shown by the dotted-line wedge at R in Figure 1(a)) are removed as they cannot be visible from R. They are said to be in the *invisible* region of R. The black points in the figure are examples.

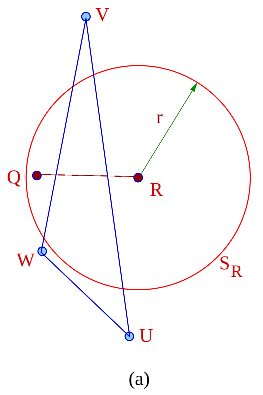
2. Similarly, points are removed which have R in their invisible region (for example, point V in the same figure). We denote the set of points from C_R remaining after this step as C_R^v.

3. Finally, we eliminate points that are occluded from R because of an existing edge in the mesh (for example, the white point in Figure 1(a)).

A straightforward approach of checking all possible occluding edges is very expensive. We state the following theorem which limits our search to very few edges.

Theorem: Only the *boundary edges* of the points in the set C_R can be possible occluding edges between R and C_R^v.

Proof: We sketch only a brief, informal proof due to space limitations. From Invariant 1, it is easy to show that if an internal edge is occluding, there must be at least one boundary edge which is also occluding. This eliminates all the internal edges from contention. Figure 2(a) shows an example where a boundary edge (say UV) occludes the point Q from R, but its endpoints are not in C_R. Since



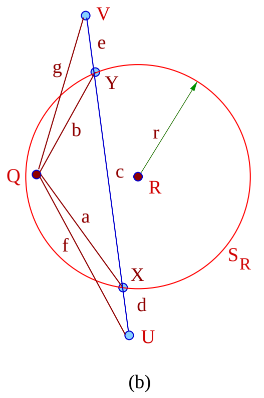


Figure 2: (a) Determining occluding edges (b) Angle XQY is obtuse, so $a < c$. Further, $f < d+a < d+c < d+c+e$. Therefore, $|QU| < |UV|$.

one of U, V or W (let us assume U) must have already been chosen as a reference point, it is a *completed* or *boundary* point by Invariant 2. Further, classical geometry tells us that $|QU| < |UV|$ (see Figure 2(b)). From our distance criterion and Invariant 2, we claim that vertex Q must be adjacent to U. The maximum allowable angle (α) criterion does not apply here if U is a *boundary* point. This implies that R lies in the invisible region of Q, and hence cannot belong to C^v_R as it will be eliminated by condition 2 above. Therefore, UV cannot be an occluding edge.

The rest of the points which are ordered around R can be triangulated as shown in Figure 1(b).

Pruning by Angle Criterion: The triangulation we get from the previous step is a valid one. However, to improve the quality of triangulation, this pruning step removes points that could potentially form triangles with very small angles ("skinny" triangles). This is not a necessary component for the working of our algorithm. Since our algorithm does not introduce additional (*Steiner*) points, it cannot always achieve the desired quality.

We explain the working of this step using an example. In Figure 3, consider the points N_1 and N_2. Let us assume that the angle at R of $\triangle RN_1N_2$ is less than β (our minimum angle parameter). One of these points can be removed to improve the triangulation. The choice of the removable vertex is not arbitrary. For example, if N_2 is rejected, it gets trapped inside the triangle (in the projection plane) formed by R, N_1, and any one of N_3, N_4, or N_5. This violates Invariant 1.

The following algorithm describes a way to avoid such scenarios and to form a good triangulation whenever possible. Assume that we have to form the triangulation from P to N_5 in Figure 3, where RP and RN_5 are consecutive *boundary edges* of R. We start our processing by ordering the points around P. In our example, this ordering would be $P_s = (N_1, N_2, N_5, N_4, N_3)$, and the ordering around R is $R_s = (N_1, N_2, N_3, N_4, N_5)$. Let $P_s[i]$ ($R_s[i]$) be the i^{th} element in P_s (R_s). Without loss of generality, we assume $R_s[i] = N_i$. The following pseudo-code finds all possible adjacent points to P around R, without trapping any other point inside the triangle.

```
for 1 $\leq i \leq 5$
    Let $N_j$ be the vertex in $P_s[i]$
    Mark $N_j$ as **considered**
    $T(N_j) = \{N_k \mid k < j, N_k$ is not marked **considered**$\}$
    if ($T(N_j) = \phi$)
        then $N_j$ **can be an adjacent point** to $P$ around $R$
        else $N_j$ **cannot be an adjacent point** to $P$ around $R$
```

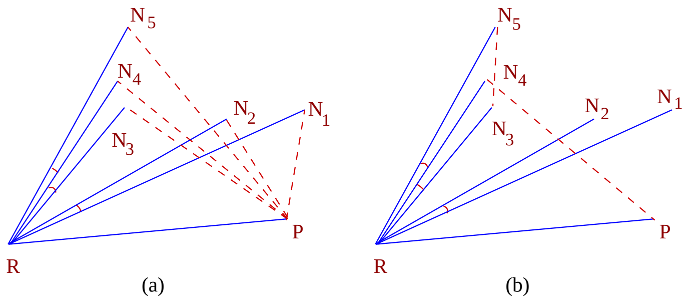


Figure 3: Pruning by Angle Criterion: (a) Ordering around R and P; angles between N_1N_2, N_3N_4, and N_4N_5 are less than β. (b) N_2 trapped in $\triangle RPN_3$, and N_4 trapped inside $\triangle RN_3N_5$

In our example, N_1, N_2, and N_3 are possible adjacent points because when they were chosen, all the points before them in R_s were already marked **considered**. If any of the remaining points are chosen, it will trap some vertex. In fact, the set $T(N_j)$ consists of precisely those vertices that will be trapped if N_j were chosen as the adjacent point to P around R.

From the set of possible adjacent points $\{N_1, N_2, N_3\}$, we can choose any vertex. For the sake of argument, let us choose N_3 as the adjacent point to P and form the triangle RPN_3. Now the same algorithm is applied at N_3, and the points N_4 and N_5 are ordered around it. It can be seen that the point N_4 cannot be removed, as it will get trapped inside the triangle RN_3N_5. Hence, we cannot eliminate the skinny triangle RN_3N_4. It is important to note that even if we had chosen N_1 or N_2 from the original possible adjacent point set, we would have ended up in the same situation.

4.2 Triangulation

The remaining points from C_R after the various pruning steps are the final adjacent points and are connected in order around R to complete the triangulation in the object space. If consecutive adjacent points subtend more than α (maximum allowable angle parameter) at R in the object space, then they are not connected to form a triangle. This maximum angle describes the characteristics of the holes in the model, and R is considered as a *boundary* point. All the *free* points in the adjacent point list are labeled as *fringe* points and are appended in order at the end of the queue. The algorithm chooses the next point from the queue as the new reference point R, and continues with the triangulation around R.

5 Robustness of our algorithm

We avoid most of the robustness problems faced by purely geometric methods (like noise and degenerate situations) by our partially combinatorial approach. In our algorithm, we face robustness problems in the projection plane evaluation. For example, sharp curvature variations in the model might lead to incorrect estimates of the projection plane. An important feature of our algorithm is that it can detect such degenerate cases, and handle them specially.

To test the robustness of our approach to perturbations in the estimated tangent plane at R, we used one of just three projection planes – (X, Y), (Y, Z), and (Z, X), whichever was close to the actual estimate. We were able to triangulate many models including the bunny model satisfactorily. The execution time with this approach is much less than the times listed in the Table 1 because we do not need to explicitly transform the vicinity of R to its tangent plane. But the disadvantage of this approach is that it has a few favorable orientations of the model in the coordinate frame, and different orientations gave different results.

5.1 Handling Noisy Data

Ability to handle noise in the input data is an important consideration for any surface reconstruction algorithm that is applied to real-world data. Statistical methods like median filtering have been applied before to remove outliers and high frequency noise and fit non-interpolatory surfaces. But these methods can sometimes be slow, and might not even generate surfaces that are intuitively correct for some data sets.

Typically, devices used for data acquisition generate sample data in some order. In our algorithm, we can make use of this order and the characteristics of the device to handle noise. We have applied our method on a massive data set of a room (Color Plate 2 - top row) acquired by a laser range scanner. This is a common Center-Of-Projection (COP) device which returns a spherical depth map ((θ, ϕ) map) of the environment around itself. The output of such a device is called a *first-seen surface*, as it samples only the surface that it first sees. It has no information about occluded regions from its COP. The sampling density of this device is also extremely high (one sample per tenth of a degree in both θ and ϕ). When adjacent samples are less than an inch apart, the noise in the samples is nearly two inches. If we apply our original algorithm to this noisy data set, the selected projection plane would be completely altered by innumerable micro-facets formed in the vicinity of a point.

We make use of the fact that the data point is on the *first-seen surface* in the spherical coordinate system to solve this problem. In such a coordinate system, this surface can be considered as a monotonic surface with respect to a unique projection plane similar to height field or terrain data. In this case, the (θ, ϕ) plane can be considered as the projection plane for all the points. Since the perturbations in the data set due to noise are always orthogonal to the projection plane, our algorithm is not affected by it.

The underlying two dimensional *dexel array* is considered as the (θ, ϕ) projection plane with only one data point at each *dexel*. The neighbors of a point in the final triangulation can only be from its adjacent *dexels*. Hence the first step of pruning (by L_∞ metric) chooses only the points from the adjacent *dexels* of R. The radius of S_R is set to a slightly higher value than the noise in the system. As all the points in C_R are visible from R, visibility or angle checks can be skipped. If we choose to remove these tests, it takes less than seven seconds to reconstruct a data set of size around 900,000 points.

In practice, we fix the dimensions of the *dexel* array. We retain one representative point if multiple points get mapped onto the same *dexel*. This thins the high sampling density, and forms a level of simplification. The dimensions of the *dexel* array controls the amount of simplification and the run time of the algorithm. Since all the processing time in our algorithm is dependent on the number of *candidate points*, bounding this number is a major source of speed-up in handling terrain data.

The image in Color Plate 2 (top row) shows the textured reconstructed room model. The texture is created from the intensity values returned by the laser device. The image on the right shows the micro-facets in the floor of the room, in spite of point simplification.

6 Application of Our Approach

The strength of our algorithm lies in its speed and memory requirements. In telepresence applications [23], where multiple cameras are used to extract the geometry of a dynamic scene at interactive rates, the surface reconstruction algorithm should cope up with the multi-million point data input, simplify it and provide a suitable surface representation. Our algorithm is well-suited for these kinds of applications, and can be used to simplify, reconstruct the surface, merge the surfaces extracted from different cameras, and stitch the geometry to create one single geometric representation of the scene. The following approach can be used to solve this problem.

Multiple cameras are located at known locations in the environment to be sampled. The depth extraction can be done using any vision algorithm such as a stereo-based approach. Each pixel in the image captured by every camera has a depth value associated with it. As the image with depth from each camera can be considered as a height field, our algorithm can be used to triangulate the point cloud from each camera independently. The merging of geometry from different cameras is done as follows. The reconstructed mesh of a camera, say C_1, can be transformed to another camera's (say C_2) viewpoint, as we know the exact location and orientation of each camera. The transformed mesh is projected as a *depth mask* on the *dexel array* of C_2. The *first seen surface* property ensures that there can be no point seen from C_2 with a higher depth than its corresponding depth mask value. Points seen by both C_1 and C_2 (equal depth values) are safely eliminated. The triangulation of the rest of the points would give the mesh representation of the geometry visible from C_2 only. The stitching of these meshes is done by considering only their *boundary* points. The resulting surface can be texture mapped with the color values returned by the camera images, to generate a realistic model with fewer data points.

We have also shown the versatility of our approach by applying it on completely unorganized point clouds. However, it is specially suited for image-based rendering applications where the data set is in the form of height fields.

Model	No. of points	No. of Triangles	Init. Time (in secs)	Recons. Time (in secs)
Club	16864	33660	0.2758	3.9644
Bunny	34834	69497	0.5961	9.1809
Foot	20021	39919	0.3802	5.2725
Skidoo	37974	75461	0.6680	8.536
Mannequin	12772	25349	0.2405	3.9289
Phone	83034	165730	1.5634	26.597

Table 1: Performance of our algorithm: See Color Plate 1

7 Implementation

This section describes the issues we faced while implementing our algorithm. We store the all the data points in our *dexel* data structure. Each point in the *dexel* array has a *z* value and a pointer to other vertex attributes.

Generating a unique surface representation *irrespective of the orientation* of a model is an important requirement of any surface reconstruction algorithm. The result of our algorithm is based on the order in which vertices are processed. This, in turn, is dependent on the order of insertion of all the new *fringe* points in the breadth-first-process queue. Any metric which is independent of the orientation of the model can be used for this ordering. In our implementation, we insert points into the queue in the same order as we form new triangles.

Our algorithm derives its speed and efficiency from the simplicity of each of its stages. Further, as most of the computation is on the projection plane, a lot of geometric tricks can be used to speed up the process. We have taken care to avoid costly operations like square roots and trigonometric functions. The ordering of vertices on a plane around a point is done by partitioning the points in different quadrants in the local coordinate system and ordering them independently by computing the $sin(\theta)$ function using first principles. We avoid repeated computations as much as possible by striking a trade-off between memory and speed. Once a point is marked as a *completed* point, memory used by it is immediately freed. As we have optimized our function to order points around R, we use the same function to perform some of the visibility pruning steps.

8 Performance and Results

The complexity of our algorithm is input sensitive, *i.e.,* time spent is proportional to the model complexity. This can be seen from the results shown in Table 1. The bunny model, which has fewer points

No.of points	No. of Triangles	Init. Time* (in secs)	Recons. Time 1 (in secs)	Recons. Time 2 (in secs)
143858	267131	82.508	5.998	1.020
883577	1707468	88.554	38.782	6.913

Table 2: Performance of the system for the Room range data set: See Color Plate 2. Reconstruction Time 2: without visibility and angle criteria check. (*: Includes the reading time of the original data set – 6479713 points)

than the skidoo model, takes more time for reconstruction, due to its complexity. Similarly, the mannequin model takes almost the same time as that of the club model, because of high curvature variations and non-uniform sampling in the regions near the nose, eyes and ears. For the same reason, we are able to handle massive data sets of size in the order of millions of points in a few seconds, as we are making use of height field data properties.

Our algorithm is a single pass algorithm, and does not need to revisit the triangles once they are formed. We do not produce any higher dimensional simplices (like tetrahedra [9, 14]) that require their removal to make it a valid manifold. We also do not change the triangulation once they are completed.

Reduced memory requirement is another feature of our algorithm. Our algorithm does not store the triangles formed during reconstruction in the main memory. Only those triangles which are incident on *fringe* and *boundary* points are retained, as they are used for visibility pruning. Hence we are able to handle massive models with millions of data points.

Table 1 shows the time taken by our algorithm on various point clouds. The initialization time in the table includes the time taken to read in the model and initializing the data structure. All the timing measurements in this paper were made on a 250 MHz, R10000 SGI Onyx2 with 16 GB of main memory. Table 2 shows the timing of our algorithm on the laser data. The initialization time includes the time to read in the original model of around 6.5 million points, filling up the data structure and eliminating the points. The two entries in the table show the timings for two different sizes of the *dexel array*: 400×600 and 1000×1500.

8.1 Limitations of Our Approach

Any projection-based approach gives different triangulation for different starting points. Our approach also suffers from the same limitation. But once the seed point is fixed, the triangulation is same for any transformation of the model. The second limitation is also common to most surface reconstruction algorithms – sharp curvature variations. If the faces incident on a vertex do not satisfy our criterion of surface smoothness, then our algorithm might produce incorrect triangulations. For under-sampled and extremely non-uniformed models, our algorithm produces spurious model boundaries, as shown in the Color Plate 2 (bottom row).

9 Conclusion

We have presented a new projection-based surface reconstruction algorithm from unorganized point clouds. The key features of our method are speed, memory efficiency and linear time performance. Further, it is a single pass algorithm and can make use of the characteristics of the data acquisition phase to handle noisy data. We have demonstrated the application of our algorithm on various data sets, including a massive, noisy range scan model of a room. We are successfully able to generate valid, non-self-intersecting, orientable manifold surface meshes for point clouds of size a few hundred thousand in a matter of tens of seconds. We believe that such a performance without any manual intervention or restricted applicability is a big win for our algorithm.

References

[1] N. Amenta and M. Bern. Surface reconstruction by voronoi filtering. In *ACM Symposium on Computationl Geometry*, 1998.

[2] N. Amenta, M. Bern, and M. Kamvysselis. A new voronoi-based surface reconstruction algorithm. In *Proceedings of ACM Siggraph*, pages 415–421, 1998.

[3] D. Attali. r-regular shape reconstruction from unorganized points. In *ACM Symposium on Computationl Geometry*, pages 248–253, 1997.

[4] C. Bajaj, F. Bernardini, and G. Xu. Automatic reconstruction of surfaces and scalar fields from 3d scans. In *Proceedings of ACM Siggraph*, pages 109–118, 1995.

[5] C. Bajaj, F. Bernardini, and G. Xu. Reconstructing surfaces and functions on surfaces from unorganized 3d data. *Algorithmica*, 19:243–261, 1997.

[6] F. Bernardini. *Automatic Reconstruction of CAD Models and Properties from Digital Scans*. PhD thesis, Purdue University, Department of Computer Science, 1996.

[7] F. Bernardini and C. Bajaj. Sampling and reconstructing manifolds using alpha-shapes. In *Proc. of Ninth Canadian Conference on Computational Geometry*, pages 193–198, 1997.

[8] E. Bittar, N. Tsingos, and M. P. Gascuel. Automatic reconstruction from unstructured data: Combining a medial axis and implicit surfaces. *Computer Graphics Forum, Proceedings of Eurographics*, 14(3):457–468, 1995.

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

[10] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of ACM Siggraph*, pages 303–312, 1996.

[11] H. Edelsbrunner and E. Mucke. Three dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994.

[12] W. Fulks. *Advanced Calculus: An introduction to analysis*. John Wiley & sons, 1978.

[13] B. Guo, J. Menon, and B. Willette. Surface reconstruction from alpha shapes. *Computer Graphics Forum*, 16(4):177–190, 1997.

[14] B. Heckel, A. C. Uva, and B. Hamann. Clustering-based generation of hierarchical surface models. In C. M. Wittenbrink and A. Varshney, editors, *IEEE Visualization '98: Late Breaking Hot Topics Proceedings*, pages 41–44, 1998.

[15] T. Van Hook. Real-time shaded NC milling display. In *Proceedings of ACM Siggraph*, pages 15–20, 1986.

[16] H. Hoppe. *Surface Reconstruction from Unorganized Point Clouds*. PhD thesis, University of Washington, Department of Computer Science, 1994.

[17] H. Hoppe, T. Derose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized point clouds. In *Proceedings of ACM Siggraph*, pages 71–78, 1992.

[18] H. Hoppe, T. Derose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. In *Proceedings of ACM Siggraph*, pages 21–26, 1993.

[19] R. Mencl. A graph-based approach to surface reconstruction. *Computer Graphics Forum, Proceedings of Eurographics*, 14(3):445–456, 1995.

[20] R. Mencl and H. Muller. Graph-based surface reconstruction using structures in scattered point sets. *Proceedings of CGI '98*, 1998.

[21] R. Mencl and H. Muller. Interpolation and approximation of surfaces from three-dimensional scattered data points. *State of the Art Reports, Eurographics '98*, pages 51–67, 1998.

[22] E. P. Mucke. *Shapes and Implementations in Three-Dimensional Geometry*. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, 1993.

[23] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The office of the future: A unified approach to image-based modeling and spatially immersive displays. In *Proceedings of ACM Siggraph*, pages 179–188, 1998.

[24] R. Szeliski and D. Tonnesen. Surface modeling with oriented particle systems. In *Proceedings of ACM Siggraph*, pages 185–194, 1992.

[25] M. Teichmann and M. Capps. Surface reconstruction with anisotropic density-scaled alpha shapes. In *Proceedings of IEEE Visualization*, pages 67–72, 1998.

[26] D. Terzopoulos, A. Witkin, and M. Kass. Constraints on deformable models: Recovering 3d shape and nonrigid motion. *Artificial Intelligence*, 36:91–123, 1988.

[27] R. C. Veltkamp. Boundaries through scattered points of unknown density. *Graphical Models and Image Processing*, 57(6):441–452, 1995.