Moving Least Squares Multiresolution Surface Approximation

Boris Mederos  Luiz Velho  Luiz Henrique de Figueiredo

April 28, 2003

Abstract

We describe a new method for surface reconstruction based on unorganized point clouds without normals. We also present a new method to refine the initial triangulation. The output of the method is a refined triangular mesh with points in the moving least squares surface of the original point cloud.

Keywords: multi-resolution; surface reconstruction; \( k \)-nearest neighbors; moving least squares.

1 Introduction

We consider the problem of surface reconstruction and refinement from scattered data points without normals. Several algorithms are known for this important problem [4–13], including a number of recent algorithms with theoretical guarantees [4–7]. Those algorithms use a 3D Delaunay triangulation of the original point cloud to compute a triangular surface mesh. Computing the Delaunay triangulation can be slow and susceptible to numerical errors.

Gopi, Krishnan, and Silva [12] proposed an algorithm based on Differential Geometry that projects the neighborhood of each sample point on a tangent plane, computes the 2D Delaunay triangulation of this projected neighborhood, and lifts it to 3D.

Hoppe et al. [10] estimate a tangent plane at each sample point using its \( k \)-nearest neighbors and use the distance to the plane of the nearest sample as an estimative of the signed distance function to the surface. The zero set of this distance function is extracted using the marching cubes algorithm [11].
Other algorithms [8, 9, 13] use a greedy approach. The ball pivoting algorithm [9] rolls a ball over the sample points; it requires the normal to the surface at each sample point, but it can create artifacts.

Boissonnat [8] starts by finding an initial edge pq in the triangulated reconstruction; he then computes a tangent plane around the edge, projects the k-nearest neighbors of both vertices onto this plane and determines the point r that maximizes the angle \( \angle \overline{pq} \) (where the bar represents projected points on the tangent plane). This point r determines a surface triangle prq. The process is repeated for each border edge, resulting in a triangulated surface. Boyer [13] proposes an incremental algorithm over the 3D Delaunay triangulation of the samples, and relies on regular interpolants.

Our algorithm uses a different approach: it computes a set of representative points near of the original point cloud and triangulate these representative points using a new incremental algorithm based on k-nearest neighbors. This is somewhat similar to what is done in [8, 9, 13]: For each border edge, the algorithm uses an angle criteria to select a point to make a triangle with the edge. The initial triangulation is refined using a new method based on moving least square operators [1–3]. Our algorithm does not need Delaunay triangulations and it can handle surfaces with borders.

Section 2 describes in detail the four steps of our method: clustering, reduction, triangulation, and refinement. These steps are illustrated above. Section 4 discusses several examples of the method in action.

## 2 Our Method

Starting from a point cloud \( Q = \{ q_i \} \subseteq \mathbb{R}^3 \) that samples an unknown \( C^1 \) manifold \( S \) contained in \( \mathbb{R}^3 \), our method produces a mesh with points near the moving least-squares surface of \( Q \). The method has four steps:

1. **Clustering**: Split the point cloud into a set of clusters.
2. **Reduction**: Compute for each cluster a representative point onto the moving least squares surface of its k-nearest neighbors.
3. **Triangulation**: Build a triangulated surface over the set of projected representative points.
4. **Refinement**: Refine the initial mesh into a finer mesh that is adapted to the geometry of the unknown surface \( S \).

We now describe each of these steps in detail.

### 2.1 Clustering

The goal of this first step is to partition the original set of points \( Q \) into a finite set of clusters, such that the curvature of the original surface \( S \) varies
a little within each cluster. Since $S$ is not known, its curvature must be estimated from the sample points $Q$.

We use a hierarchical clustering method based on a BSP tree [1]. Each node in this tree contains a subset $P = \{p_1, \ldots, p_n\}$ of the original point cloud $Q$. We use the covariance matrix $C$ of $P$ to decide whether or not to subdivide the node:

$$C = \begin{bmatrix}
p_1 - \bar{p} \\
p_2 - \bar{p} \\
\vdots \\
p_n - \bar{p}
\end{bmatrix}^T \begin{bmatrix}
p_1 - \bar{p} \\
p_2 - \bar{p} \\
\vdots \\
p_n - \bar{p}
\end{bmatrix}, \quad \bar{p} = \frac{1}{n} \sum_{i=1}^{n} p_i.$$

Note that $C$ is a $3 \times 3$ matrix.

Since $C$ is symmetric, positive semi-definite, its three eigenvalues are real and we can order them: $\lambda_1 \leq \lambda_2 \leq \lambda_3$. These eigenvalues measure the variation of the points in $P$ along the directions of their corresponding eigenvectors $v_1, v_2, v_3$.

The eigenvectors $v_2$ and $v_3$ define the directions of highest variation and define a regression plane $\Pi$ for $P$. The eigenvector $v_1$ is normal to $\Pi$ and its eigenvalue $\lambda_1$ measures the variation of the points in $P$ with respect to the plane $\Pi$. So, small values of $\lambda_1$ mean that all points in $P$ are approximately on $\Pi$. Hence, the ratio

$$\sigma = \frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3}$$

is a good measure of the flatness of the point set $P$ and can be used as an estimate for the curvature of $S$ around $P$.

We use this flatness measure to define subdivision criteria for the BSP tree. A node is divided in two when both conditions below are satisfied:

1. The ratio $\sigma$ is larger than a user-defined tolerance $\sigma_{\max}$;
2. The number $n$ of points in $P$ is larger than a user-defined value $n_{\min}$.

The nodes are divided by classifying the points of $P$ with respect to the regression plane $\Pi$. The points of $P$ that are in the same side of $\Pi$ will
form a new node. The leaves of the BSP tree will be the clusters. See Figure 2.

2.2 Reduction

We find a representative point for each cluster using a new method based on the moving least squares theory \[1-3\]. The moving least square surface of a set of points \(R\) near the surface \(S\) is the set of fixed points of a projection operator \(\Psi(R, \cdot)\) defined in \[1-3\]:

\[
MLS(R) = \{y \in \mathbb{R}^3 : \Psi(R, y) = y\}.
\]

Given a point cloud \(Q\) that samples the unknown surface \(S\) and a point \(r \in \mathbb{R}^3\) we compute the operator \(\Psi(K, r)\) as an approximation of the projection operator \(\Psi(Q, r)\). Where \(K\) is the set of \(k\) nearest neighbors of \(r\) in \(Q\), see for details \[3\].

To determine the representative point of each cluster we computed the point \(c\) (the centroid of the set of points \(P_0\) in the cluster). The point \(c\) is not necessarily near of \(S\). In order to approximate \(c\) to the surface \(S\) we compute the point \(r = c + t_{\text{min}}n_c\). Where \(n_c\) is the normal direction of this point, calculated as the eigenvector associated to the minimum eigenvalue of the matrix \(M = \{m_{ij}\}\). The matrix \(M\) is a \(3 \times 3\) weighted covariance matrix over the set \(K\) of \(k\) nears neighbors of \(c\) in \(Q\), the \(m_{ij}\) are defined by:

\[
m_{ij} = \sum_{p_i \in K} (p_{li} - c_i)(p_{lj} - c_j)\exp(-\frac{\|p_i - c\|^2}{h^2}).
\]

Where \(h\) is a parameter reflecting the spacing between neiboring points, see \[1,2\] for more details. We use \(h = 3 \cdot r\) with \(r\) the minimum distance of \(c\) to its near neighbor in \(Q\).
Using the direction $n_c$ we calculate $t_{\text{min}}$ as the minimum of the functional below, this functional can be interpreted as the distance of $c + t_{\text{min}}n_c$ to the original surface

$$\sum_{p_i \in K} d(p_i, c + tn_c)^2.$$  

The point $r = c + t_{\text{min}}n_c$ will be projected in the moving least squares surface MLS($Q$) of the initial set of sampled points. The final point $r_{\text{proj}} = \Psi(Q, r)$ will be the representative of the cluster. See Figures 3.

### 2.3 Triangulation

The next step is to build a triangulated surface over the set of representative points. We do this using an incremental algorithm that introduces new features in the basic framework of previous incremental algorithms [8,9,13].

The main idea behind the algorithm is to determine incrementally the restricted Delaunay triangles, because these triangles form a piecewise linear manifold homeomorphic to the original surface [4].

The algorithm computes a sequence of triangulated surfaces with border. At each step, it chooses a border edge, finds a new triangle associated to this edge, and updates the current surface. The algorithm maintains a half-edge data structure $H$ that represents the current surface and a list $L$ of half-edges that represents the current boundary. Here is a summary of the main steps:

1. **Build a 3d-tree on the set of representatives.**
2. **Get an initial triangle and initialize $H$ and $L$ with it.**
3. **repeat**
   - **Remove a half edge $uv$ from $L$.**
   - $usv \leftarrow$ the triangle adjacent to $uv$ in $H$.
   - $P \leftarrow$ set of $k$-nearest neighbors of $u$.
   - $q \leftarrow$ GET_POINT($usv$, $u$, $P$)
   - Insert $uqv$ into $H$ and its border edges into $L$.
4. **until** $L$ is empty.
The initial surface contains a single triangle. This triangle is inserted into $H$ and all its edges are inserted into $L$. To find the initial triangle, we select a point $p \in Q$ and determine its nearest neighbor $q \in Q$. These two points define the first border edge: $pq$. Among the $k$-nearest neighbors of $p$, we determine the point $r$ that maximizes the angle $\angle prq$. The initial triangle $prq$ is in the Delaunay triangulation of the $k$-nearest neighbors of $p$ (see section 3, Lemma 3).

Once the initial surface has been found, we continue by removing edges from $L$ until it is empty. For each edge $uv$, we select a point $q$ among the $k$-nearest neighbors of $u$ to create a new triangle $uqv$, which is added to $H$ and $L$. (We use a 3d-tree built at the start of the algorithm to identify $k$-nearest neighbors.)

The number of edges of $uqv$ inserted in $L$ varies: it is zero when the triangle $uqv$ fills a hole in $H$; it is one when $uq$ or $vq$ are consecutive edges of $uv$ in the border of the current surface; otherwise, it is two. When $q$ is already in $H$, we have to join two connected components of the border or to split one component in two new connected components. We determine the point $q$ using the following algorithm:

**GET\_POINT** ($usv$, $u$, $P$):

1. $d_u \leftarrow$ distance from $u$ to its nearest neighbor.
2. $d_v \leftarrow$ distance from $v$ to its nearest neighbor.
3. $d_{min} \leftarrow \min\{d_u, d_v\}$.
4. repeat
   1. $C_\theta \leftarrow$ set of points $t$ such that:
      1. the dihedral angle between the triangles $usv$ and $utv$ is in $[\pi - \theta, \pi + \theta]$  
      2. Triangle $utv$ does not violate the topology of $H$.
      3. $\max\{d(u, t), d(v, t)\} < d_{min}$.
   2. if $C_\theta \neq \emptyset$ then
      1. Determine the set $C'_\theta$ of points $q \in C_\theta$ with maximum angle $\angle uqv$.
   3. else
      1. $\theta \leftarrow \theta + \varepsilon_2$.
   4. until $C'_\theta \neq \emptyset$ or $\theta > \frac{\pi}{2} + \varepsilon_2$

Here, $\theta$, $\varepsilon_1$, and $\varepsilon_2$ are user-defined tolerances.

Condition (i) in this algorithm is based on the theorem below, which we prove in section 3:

**Theorem 1** Let $F$ be a $\beta, r$-sampling, ($r < \frac{1}{4}$) of a surface $S$. Then the angle between two adjacent restricted Delaunay triangles sharing an edge is at least $\pi - 2\left(\frac{2r}{1-4r} + \arcsin\left(\frac{3r}{1-r}\right)\right)$.

This theorem show that the dihedral angle between two adjacent restricted Delaunay triangles converges to $\pi$ as the sampling density increases.
(that is, as $r \to 0$). We start with an initial region $[\pi - \theta, \pi + \theta]$ with small $\theta$ to determine the set of point $C_\theta$ in this region that satisfy criteria (ii) and (iii). If $C_\theta$ is not empty, we find the point $q \in C_\theta$ with maximum angle $\angle u q v$; otherwise, we increase $\theta$ by $\varepsilon_2$ and repeat the process until we find a point $q$ or $\theta$ gets too large.

Condition (ii) is used to eliminate the points $t$ that are interior to $H$ and the points on the boundary of $H$ that would violate the topology if they were selected. When the point $t$ is on the boundary of $H$, we first fit a plane $\Pi$ to the set of points adjacent to $t$ in $H$ (the star of $t$) and project this start onto $\Pi$, resulting in a set of 2d triangles $T$. If these triangles do not form a triangulation, that is, is two edges intersect, then we have a topology violation.

Condition (iii) is used to determine border edges, and is based on the following theorem proved in section 3:

**Theorem 2** Let $F$ be a $\beta, r$-sampling ($r < \frac{1}{3}$) of a surface $S$. Let $T_1 = uvt$ and $T_2 = uvq$ be restricted Delaunay triangles sharing the edge $uv$. Then, the length of the longest edge of $T_2$ is at most $\frac{2\beta}{1-3r}d$, where $d = \min\{d_u, d_v\}$ and $d_u$ (respectively, $d_v$) is the distance of $u$ (respectively, $v$) to its nearest neighbor.

This theorem shows that there is little variation between the length of the edges of two triangles adjacent to a interior edge. We observed that the length of edges of the adjacent triangles to a boundary edge are very different.

In practice, we have no way to estimate $\beta$ and $r$, and so we use a predefined value $\varepsilon_1$ as the constant $\frac{2\beta}{1-3r}$ and a point $t$ is eliminated if $\frac{\max\{d(u,t),d(v,t)\}}{d_{\min}} > \varepsilon_1$. If $C_\theta$ is empty, the edge $uv$ is a candidate border edge. If $C_\theta$ is empty for all $\theta$ tested, $uv$ is a border edge.

If $C_\theta$ is not empty, we compute the subset $C'_{\theta}$ of points $q$ with maximum angle $\angle u q v$. We justify this criterion as follows: In $\mathbb{R}^2$, given a Delaunay triangle $uvs$, its adjacent Delaunay triangle $uvq$ is the one with maximum angle $\angle u q v$. Although this is a criterion for $\mathbb{R}^2$, we use it for surfaces because the surface normal varies little in the surface neighborhood of a point $u$ containing the vertex of the restricted Delaunay triangulation (see [4–6]); in other words, we can consider this neighborhood as flat.

This triangulation algorithm is fast. Figure 4 shows the triangulated surfaces over the set of representatives shown in Figure 3.

### 2.4 Refinement

The goal of this step is to refine the initial coarse triangulation into a finer triangulation adapted to the geometry of the unknown surface $S$ from which the point cloud $Q$ was sampled.
We propose the following method to refine an edge $uv$: We compute the middle point of the edge. If this point is too far from the original point cloud $Q$, it is projected onto the moving least squared surface and a new edge is added to each adjacent triangle of the edge $uv$, dividing each triangle into two new triangles. We repeat this process for each edge in the original triangulation. The details are discussed below.

More precisely, for each edge $uv$ we compute its midpoint $m = \frac{u + v}{2}$ and its normal

$$n_m = \frac{n_u + n_v}{\|n_u + n_v\|},$$

where $n_u$ and $n_v$ are the normals of $u$ and $v$ respectively, computed using the local triangulation (star) of each vertex.

Using the set $K$ of $k$-nearest neighbors of $m$ in the original points cloud, we minimize the following functional with respect to $t$:

$$\alpha = \min_t \sum_{p \in K} \|p - m + t \cdot n_m\|^2 = -\frac{1}{K} \sum_{p \in K} (p - m) \cdot n_m$$

As in the reduction step, this is interpreted as finding the point on the line through $m$ in the direction of $n_m$ that is closest to the original surface $S$.

We use $\alpha$ as a measure of the need of refinement. If $\alpha$ is smaller than some predefined value $\varepsilon$, we do not refine the edge; otherwise, we project the point $m + \alpha \cdot n_m$ onto the moving least squares surface, producing the new

Figure 4: Triangulations
point $\bar{m} = \Psi(Q, m + \alpha \cdot n_m)$ and replace the triangles $a_{uv}$ and $b_{uv}$ adjacent to $uv$ with the new triangles $u\tilde{m}a$, $v\tilde{m}a$ and $u\tilde{m}b$, $v\tilde{m}b$, respectively. If the ratio between the length of the minimum edge and the maximum edge of the triangles $a_{uv}$ or $b_{uv}$ is too small, we do not divide the triangles.

This procedure is applied to all the old edges while the scalar $t$ is larger than $\varepsilon$. In the resulting mesh, we flip the diagonals of each quadrilateral formed by the pairs of adjacent triangles if the length of one diagonal (the common edge) is larger than the length of the other diagonal (the segment joining the opposite vertex).

The final result is a triangulated surface whose vertices are on the moving least squares approximation surface of the original points cloud. Figure 5 shows two refinement of the initial triangulations shown in Figure 4.

3 Theory

In this section we will prove the Theorems 1 and 2 proposed in 2.3 (Triangulation) to justify the reconstruction algorithm. Also we prove the lemma 3 referenced in the same section 2.3. First we will give an introduction to the theory necessary to prove the theorems.

Amenta et al. [4–6] gives a definition of $lfs(x)$ (local feature size of $x$) as the distance of $x$ to the medial axis of the surface $S$ and also define the concept of $r$-sample:

**Definition 1** (Amenta et al. [4].) The set $F \subset S$ is a $r$-sample if for each $p$
in $F$ then $B(p, r \cdot lfs(p)) \cap F \neq \emptyset$.

We will use a different sampling criterium, see [14]

**Definition 2 (Dey et al. [14].)** The set $F \subset S$ is a $\beta, r$-sample $\beta > 1$ if the following two condition are valid:

i) For all $x \in S B(x, r \cdot lfs(x)) \cap F \neq \emptyset$.

ii) For all $p \in F B(p, r \cdot lfs(p)) \cap F = \{p\}$.

The Voronoi diagram of a set of samples $F \subset S$ induce a decomposition in the surface named the restricted Voronoi diagram. Given a Voronoi cell $V_p$ the restricted voronoi cell associated to $p$ is $V_{p,S} = V_p \cap S$. Its dual, the restricted Delaunay triangulation, is obtained in the following way: $pq$ is an edge if $V_{p,S} \cap V_{q,S} \neq \emptyset$, $pqr$ is a triangle if $V_{p,S} \cap V_{q,S} \cap V_{r,S} \neq \emptyset$. Amenta in [4] prove that the polyhedron defined by the restricted Delaunay diagram is homeomorphic to the original surface under sufficiently dense $r$-sampling ($r \leq 0.1$).

The next two lemmas were proved in [4]. The first, state a bound in the angle between the normals of points sufficiently nears. The second, bounds the angle between the normal to a restricted Delaunay triangle and the surface normal at the vertex with angle at least $\frac{\pi}{3}$.

**Lemma 1 (Amenta et al. [4].)** For any two point $p$ and $q$ on $F$ with $d(p, q) < r \cdot \min\{lfs(p), lfs(q)\}$, for $r < \frac{1}{3}$, the angle between the normal $n_p$ and $n_q$ is less than $\frac{r}{1-3r}$.

**Lemma 2 (Amenta et al. [4].)** If $T$ is a restricted Delaunay triangle and $s$ is a vertex of $T$ with angle at least $\frac{\pi}{3}$ then the angle between the normal at $s$ in $S$ and the normal of $T$ is at most $\arcsin(\sqrt{\frac{3r}{1-r}})$.

We will use the lemmas above to prove the **Theorem 1** proposed in section 2.3 to bound the angle between two adjacent restricted Delaunay triangles.

**Theorem 1** Given a $\beta, r$-sample $F$ ($r < \frac{1}{4}$) of a surface $S$, the angle between two adjacent restricted Delaunay triangles sharing an edge is at least $\pi - 2(\frac{2r}{1-4r} + \arcsin(\frac{\sqrt{3r}}{1-r}))$.

**Proof.** Let be $T_1 = p_1p_2p_3$ and $T_2 = p_1p_2q_3$ two restricted Delaunay triangles sharing an edge $p_1p_2$ and $s = V_{p_1,S} \cap V_{p_2,S} \cap V_{p_3,S}$. Because the nearest points to $s$ are the $p_i$ ($i = 1, 2, 3$) we have $d(s, p_i) < r \cdot lfs(s)$.

Using that $lfs(s) < d(s, p_i) + lfs(p_i)$ and $d(s, p_i) < r \cdot lfs(s)$ we obtain $lfs(s) < \frac{1}{1-r}lfs(p_i)$, from this inequalities we get $d(s, p_i) < \frac{r}{1-r} \cdot \min\{lfs(s), lfs(p_i)\}$. Using **Lemma 1** we have that the angle between the
normals \( n_s \) and \( n_{p_i} \) is less than \( \frac{r}{1-4r} \), hence, the angle between the two normals \( n_{p_i} \) and \( n_{p_j} \) is less than \( \frac{2r}{1-4r} \).

We assume without lost of generality that \( p_3 \) is a vertex with angle at least \( \frac{\pi}{4} \). Applying Lemma 2 we obtain that the angle between the normal of the surface at \( p_3 \) and the normal of the triangle \( T_1 \) is less than \( \arcsin\left(\frac{\sqrt{3}}{1-r}\right) \).

Combining this inequalities we get:

\[
\angle(n_{p_i}, T_1) < \angle(n_{p_i}, n_{p_3}) + \angle(n_{p_3}, T_1) < \frac{2r}{1-4r} + \arcsin\left(\frac{\sqrt{3}}{1-r}\right)
\]

In particular \( \angle(n_{p_1}, T_1) < \frac{2r}{1-4r} + \arcsin\left(\frac{\sqrt{3}}{1-r}\right) \) and thinking in the same way we get \( \angle(n_{p_1}, T_2) < \frac{2r}{1-4r} + \arcsin\left(\frac{\sqrt{3}}{1-r}\right) \). We conclude that the angle between \( T_1 \) and \( T_2 \) is at least \( \pi - 2\left(\frac{2r}{1-4r} + \arcsin\left(\frac{\sqrt{3}}{1-r}\right)\right) \).

It is important to note that in the prove of the theorem above we do not use the parameter \( \beta \), this theorem is valid in the more general case of a \( r \)-sampling.

In the section 2.3 we propose also the Theorem 2 that define the criterion used to determine border edges.

**Theorem 2** Let \( F \subset S \) be a \( \beta, r \)-sampling \( (r < \frac{1}{4}) \). Let \( T_1 = p_1p_2p_3 \) and \( T_2 = p_1p_2p_4 \) be restricted Delaunay triangles sharing the edge \( p_1p_2 \). Then, the length of the longest edge of \( T_2 \) is at most \( \frac{2\beta}{\sqrt{3}r}d \), where \( d = \min\{d_{p_1}, d_{p_2}\} \) and \( d_{p_1} \) (respectively, \( d_{p_2} \)) is the distance of \( p_1 \) (respectively, \( p_2 \)) to its nearest neighbor.

**Proof.** The distance \( d(p_i, p_j) \) between the points \( p_i \) and \( p_j \) in the triangles \( T_k, k = 1, 2 \) is less than \( \frac{2r}{1-4r}lfs(p_i) \). Because \( d(p_i, p_j) + lfs(p_j) > lfs(p_i) \) and \( d(p_i, p_j) < \frac{2r}{1-4r}lfs(p_i) \) we get \( lfs(p_i) > lfs(p_j) - \frac{2r}{1-4r}lfs(p_j) \), hence \( lfs(p_i) > \frac{1-3r}{1-4r}lfs(p_j) \).

Using the last equation and \( d(s, p_i) > \frac{\beta}{2}lfs(p_i) \), where \( s \) is the nearest neighbor of \( p_i \) \((i = 1, 2)\) we will have the following lower bound for the distance of \( s \) to \( p_i \): \( d(s, s_i) > \frac{r(1-3r)}{\beta(1-r)}lfs(p_j), \ j = 1, 2 \).

The edges \( p_ip_4, i = 1, 2 \) are inside the Voronoi ball with center at \( s = V_{p_1,S} \cap V_{p_2,S} \cap V_{p_4,S} \) and radius \( d(s, p_i) \). The diameter of this Voronoi ball is less than \( \frac{2r}{1-4r}lfs(p_j) \), \( j = 1, 2 \), hence an upper bound of \( d(p_i, p_4) \) is \( \frac{2r}{1-4r}lfs(p_j) \).

We can conclude that \( d = \min\{d_{p_1}, d_{p_2}\} \) is bigger than \( \frac{r(1-3r)}{\beta(1-r)}lfs(p_j) \) and the length of the longest edge of \( T_2 \) is less than \( \frac{2r}{1-4r}lfs(p_j) \), therefore, the ratio between the upper bound and the lower bound is \( \frac{2\beta}{\sqrt{3}r} \). \( \square \)
The next lemma shows that the initial triangle calculated by the algorithm is in the Delaunay triangulation of the set of \( k \)-nearest neighbors of the initial vertex.

**Lemma 3** Let be \( p, q, r \in F \), \( q \) the nearest neighbors of \( p \) in \( F \) and \( r \) the point in the set of \( k \)-nearest neighbors of \( p \) maximizing the angle \( \angle prq \), then the triangle \( T = pqr \) is in the Delaunay triangulation of the \( k \)-nearest neighbors of \( p \).

**Proof.** Let \( s \) be the circumcenter of the triangle \( T \), we will prove that the ball \( B \) of center \( s \) and radio \( d(s, p) \) is empty of the \( k \)-nearest neighbors of \( p \). Denote by \( P \) the plane containing the triangle \( T \) and \( H \) the plane orthogonal to \( P \) through the edge \( pq \).

The Plane \( H \) divide the ball \( B \) into two disjoint region \( B_1 \) and \( B_2 \). Assume without lost of generality that \( B_1 \) is the smaller region. It is easy to see that \( B_1 \) is empty of \( k \)-nearest neighbors of \( p \). If a \( k \)-nearest neighbors \( f \) of \( p \) is inside \( B_1 \), then the distance of \( f \) to \( p \) is smaller than the distance of \( p \) to \( q \) \((d(f, p) < d(q, p))\), this is a contradiction because \( q \) is the nearest neighbor of \( p \).

The other region \( B_2 \) is empty of samples in the set of \( k \)-nearest neighbors, otherwise, if a \( k \)-nearest neighbors \( f \) of \( p \) is inside the region \( B_2 \), then the angle \( \angle pfq \) is larger than the angle \( \angle prq \). This is false, because the angle \( \angle prq \) is the maximum angle.

\[ \square \]

### 4 Results

Our algorithm has been implemented using CGAL [15] for computing the half edge data structure and the operations in the refinement. The algorithm has been tested on several data sets available on the internet [16, 17] using two refinement steps. The results are shown in Figures 6–10 (and also in the previous figures). Table 1 shows the times (in seconds) taken in each step of the algorithm and the sizes (points and triangles) of the result models.

### 5 Conclusion

We presented a new algorithm for reconstruction and refinement of surface, giving as output a refined triangular mesh with points in the moving least squares surface of the original points cloud.

The new method proposed to computing representatives points produces a simplified sample on the moving least squares surface. The new triangulation algorithm does not need to computed 3D Delaunay triangulations, which in the worst case requires quadratic time and is prone to numerical
Table 1: Results on a 1.8Ghz Pentium 4 with 512Mb of RAM running Linux (s=seconds, p=points, t=triangles)

<table>
<thead>
<tr>
<th>Model</th>
<th>Clustering</th>
<th>Triangulation</th>
<th>Ref 1</th>
<th>Ref 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dragon</td>
<td>62.43 s</td>
<td>297.8 s</td>
<td>27.22 s</td>
<td>58.98 s</td>
</tr>
<tr>
<td></td>
<td>55694 p</td>
<td>109889 t</td>
<td>361455 t</td>
<td>810367 t</td>
</tr>
<tr>
<td>Bunny</td>
<td>3.68 s</td>
<td>22.48 s</td>
<td>2.33 s</td>
<td>7.9 s</td>
</tr>
<tr>
<td></td>
<td>4457 p</td>
<td>8895 t</td>
<td>31623 t</td>
<td>107363 t</td>
</tr>
<tr>
<td>Horse</td>
<td>4.67 s</td>
<td>30.83 s</td>
<td>3.17 s</td>
<td>10.76 s</td>
</tr>
<tr>
<td></td>
<td>6164 p</td>
<td>12046 t</td>
<td>42860 t</td>
<td>144828 t</td>
</tr>
<tr>
<td>Cat</td>
<td>0.77 s</td>
<td>6.02 s</td>
<td>0.61 s</td>
<td>2.06 s</td>
</tr>
<tr>
<td></td>
<td>1244 p</td>
<td>2453 t</td>
<td>29257 t</td>
<td>8535 t</td>
</tr>
<tr>
<td>Cactus</td>
<td>0.21 s</td>
<td>1.58 s</td>
<td>0.2 s</td>
<td>0.68 s</td>
</tr>
<tr>
<td></td>
<td>404 p</td>
<td>782 t</td>
<td>2776 t</td>
<td>9614 t</td>
</tr>
<tr>
<td>Club</td>
<td>1.36 s</td>
<td>5.3 s</td>
<td>1.11 s</td>
<td>3.86 s</td>
</tr>
<tr>
<td></td>
<td>2103 p</td>
<td>4121 t</td>
<td>15037 t</td>
<td>51775 t</td>
</tr>
</tbody>
</table>

errors. The new refinement method is fast and gives a fine triangular mesh adapted to the geometry of the unknown surface.

Acknowledgments. The authors are partially supported by CNPq research grants. The authors are members of Visgraf, the computer graphics laboratory at IMPA, which is sponsored by CNPq, FAPERJ, FINEP, and IBM Brasil.

References


[16] www.cc.gatech.edu/projects/large_models

Figure 8: Horse

Figure 9: Horse Head

Figure 10: Horse Nose