Technical report CS11-04-00 Department of Computer Science, University of Cape Town

MLS reconstruction from noisy point sets

Bruce Merry[*] Christoph Held[†]

November 2011

Abstract

For digital preservation of cultural heritage sites in Africa, laser range scanning has been used to produce point clouds. The literature contains extensive work on reconstructing surface models from such point clouds, but often this prior work does not account for artefacts in the data such as vegetation. We have assessed several variations on a specific moving-least-squares (MLS) technique to determine the impact on the quality of the reconstructed surfaces. We found that correct feature size detection and explicit detection of boundaries is important, while a single iteration of almost orthogonal projection is sufficient to give good results.

1 Introduction

The physical process of scanning an object is only the first step in producing a model. The data must be cleaned of artifacts, the individual scans must then be aligned to a common coordinate system, and a surface model must be defined in terms of the point samples [Alexa et al., 2003].

Scanning of architecture and of cultural heritage sites creates a number of challenges that are not a concern when doing scanning under more controlled conditions:

- The sites are often fragile, and so markers cannot be attached to walls.
- There are often unwanted objects on site, such as trees, grass, cars, people, etc. Vegetation is particularly problematic since it has detail that is below the resolution of the scanner, leading to aliasing.
- It is not possible to obtain a completely closed reconstruction, as would be possible with a movable object. Even with complete freedom to place the scanner, the undersides of buildings cannot be scanned, and flat roofs are also challenging to scan.
- It is not practical to obtain uniform sampling density across all surfaces, as today's laser scanning instruments perform a 360° scan with a uniform angular density regardless of the distance to the surface [Rüther et al., 2011].

There are also a few advantages. Unlike machined parts, there are few sharp edges, which can cause difficulties for surface reconstruction algorithms. There are also fewer reflective surfaces, which can lead to artefacts in laser range scans when the laser beam is reflected.

We have developed a system for reconstructing a surface mesh from a set of registered scans, with features designed to address these challenges. Section 2 summarizes previous work in the relevant areas. Section 3 describes our implementation in detail, which is evaluated in Section 4. Section 5 lists our conclusions and makes suggestions for future work.

2 Background

2.1 Definitions

We use the term *surfel* to mean a point together with an oriented normal. A *splat* is an extension of a surfel to include a radius of influence. We use the term *sample* to refer to a sample point retrieved from a physical scanning process, such as laser range scanning. A sample may be augmented by additional information, either physically acquired or computed, to make it a surfel or splat.

We distinguish several sources of error in acquired samples. *Sensor noise* is error caused by limited precision in the scanning technology. *Environmental noise* consists of physically present but unwanted objects, such as people or vegetation. *Registration error* occurs when overlapping scans are not properly aligned, leading to separate sheets of samples.

2.2 Surfaces from point clouds

Curless and Levoy [1996] introduced one of the early volumetric methods, which takes advantage of knowledge of the scanning process to produce more robust results. The samples in one scan are triangulated, and this triangulation is used to estimate a signed distance function

[*]bmerry@cs.uct.ac.za
[†]christoph.held@uct.ac.za

to the surface along the line-of-sight of the scanner. This distance function is discretised to a voxel grid. The distance functions for each scan are merged with a weighted averaging scheme, and finally an isosurface is extracted.

An alternative to defining an implicit surface is to determine explicit triangulations for each scan (guided by the regular sampling grid) and then merge these triangulations together. Turk and Levoy [1994] introduce a *zippering* method that first erodes redundant triangles from the boundaries of overlapping scans, clips boundaries against each other, and then fine-tunes the geometry in the overlapping region using the original distances.

More recently, there have been a number of approaches that generalize beyond range scans, requiring only a collection of points, possibly with oriented normals. Amenta et al. [2001] describe an algorithm based on an approximation to the medial axis. Kazhdan et al. [2006] define an implicit surface function based on the Poisson equation; this algorithm has been extended to support parallel execution [Bolitho et al., 2009] and GPU-based acceleration [Zhou et al., 2011]. A feature of the Poisson methods is that they produce water-tight manifolds. This can be an advantage since it automatically fills in small gaps without the need for a separate hole-filling pass, but it is also a limitation as large gaps in scanning coverage are filled with inaccurate, extrapolated data.

A large class of algorithms are based on moving least-squares (MLS) methods. As these are the basis for our own work, they will be covered in more detail in the following sections.

2.3 Moving least squares surfaces

Levin [2003] and Alexa et al. [2001] defined an implicit surface based on moving least-squares (MLS) data interpolation. The MLS approach is based on local approximations to the surface. From an initial point $\mathbf{x}$, a surface patch is defined that approximates the surface in the neighborhood of $\mathbf{x}$. A projection operator P maps $\mathbf{x}$ onto this local approximation. The MLS surface is defined as the set of points $\mathbf{x}$ such that $P(\mathbf{x}) = \mathbf{x}$. There are many variations on this basic technique, which mostly differ in how the local patches are defined.

A common feature of MLS techniques is the use of a weight function that gives large weight to small distances and rapidly approaches zero for larger distances. Some authors use functions with infinite support such as a Gaussian, while others use functions with a similar shape but finite support. The function parameters may also vary per sample. We use w_i to denote the function applicable to sample i; in some cases we also use w_i to mean the value of this function.

Alexa et al. [2001] define the local approximation and the projection as follows:

1. First, a plane is used to define a local domain. The plane H passing through $\mathbf{a}$ with normal $\mathbf{n}$ is chosen such that $\mathbf{x} - \mathbf{a}$ is parallel to $\mathbf{n}$ and such that
$$\frac{\sum \left(\mathbf{n}\cdot(\mathbf{p}_i - \mathbf{a})\right)^2 w_i(\|\mathbf{p}_i - \mathbf{a}\|)}{\sum w_i(\|\mathbf{p}_i - \mathbf{a}\|)} \tag{1}$$
is locally minimized, choosing the local minimum for which $\mathbf{x}$ is as close to H as possible. Note that since the weights depend on $\mathbf{a}$, the projection of $\mathbf{x}$ onto H, rather than $\mathbf{x}$, this is a non-linear optimization which requires an iterative solver.

2. Define a local coordinate system in H by an affine function $h : H \to \mathbb{R}^2$, with $h(\mathbf{a}) = (0, 0)$.

3. Let $\mathbf{p}'_i$ be the projection of $\mathbf{p}_i$ onto H, and let $d_i = \mathbf{n}\cdot(\mathbf{p}_i - \mathbf{p}'_i)$ be the signed distance of $\mathbf{p}_i$ to H.

4. Use weighted least squares to optimize a bivariate polynomial defining a height-field over H. Specifically, let $g : \mathbb{R}^2 \to \mathbb{R}$ be the bivariate polynomial of some fixed degree that minimizes
$$\sum (g(h(\mathbf{p}'_i)) - d_i)^2 w_i(\|\mathbf{p}_i - \mathbf{a}\|). \tag{2}$$
The local surface approximation $f : \mathbb{R}^2 \to \mathbb{R}^3$ is then given by
$$f(\mathbf{u}) = h^{-1}(\mathbf{u}) + g(\mathbf{u})\mathbf{n}. \tag{3}$$

5. The projection of $\mathbf{x}$ is defined as $P(\mathbf{x}) = f(h(\mathbf{a}))$.

Alexa et al. [2003] originally claimed that this is a true projection procedure i.e., $P(\mathbf{x}) = P(P(\mathbf{x}))$, but Amenta and Kil [2004b] show that this does not hold. Alexa et al. [2004] describe several iterative procedures for projecting points onto the MLS surface.

Amenta and Kil [2004b] give a different definition of a point set surface in terms of *extremal surfaces*. The surface is defined in terms of a normal field $n : \mathbb{R}^3 \to \mathbb{P}^2$ and an energy function $e : \mathbb{R}^3 \times \mathbb{P}^2 \to \mathbb{R}$ ($\mathbb{P}^2$ being the space of unoriented directions). The extremal surface is defined as
$$S = \{\mathbf{x} | \mathbf{x} \in \operatorname*{arglocalmin}_{\mathbf{y}\in\ell_{\mathbf{x},n(\mathbf{x})}} e(\mathbf{y}, n(\mathbf{x}))\}, \tag{4}$$
where $\ell_{\mathbf{x},\mathbf{n}}$ is the line parallel to the direction $\mathbf{n}$ passing through $\mathbf{x}$. They show that if e is defined as in Equation (1) and
$$n(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{n}} e(\mathbf{x}, \mathbf{n}) \tag{5}$$
then this is equivalent to the previous definition but without the polynomial fitting step.

This approach is more generic, since the normal field n may be computed in other ways, such as by interpolating normals associated with the samples. The same authors also give an alternative to Equation (5) which makes projection more robust [Amenta and Kil, 2004a].

In addition to giving each sample a different radius of influence, its weight function can also be scaled based on its quality. Cuccuru et al. [2009] define the quality to be proportional to the sampling density (higher being better). Fiorin et al. [2008] use a combination of the distance from surface boundaries and the sampling density to define a quality metric.

MLS surfaces have also been extended using robust statistics to handle sharp edges and outliers [Fleishman et al., 2005, Öztireli et al., 2009] and to extract isosurfaces from large data sets [Cuccuru et al., 2009, Fiorin et al., 2007].

2.4 Algebraic point set surfaces

Guennebaud and Gross [2007] introduce Algebraic Point Set Surfaces (APSS). As with other MLS methods, the surface is defined in terms of a projection procedure. The difference is that the surface is locally approximated by a sphere rather than a plane or polynomial patch. Given a point $\mathbf{x}$ in space near the surface, the projection $P(\mathbf{x})$ is computed as follows:

1. The splats in the local neighborhood of $\mathbf{x}$ are identified i.e., those for which $\mathbf{x}$ falls within the sphere of influence of the splat.
2. The splats are weighted based on their distance from $\mathbf{x}$, using the weight functions w_i.
3. A sphere is fitted to the splats, using weighted least-squares to match the positions and normals. The sphere is represented in an algebraic (implicit) form, which simplifies the fitting procedure and robustly decays to a plane.
4. $\mathbf{x}$ is projected onto the sphere.

The MLS surface is defined as the set of points for which $P(\mathbf{x}) = \mathbf{x}$. Note that P is not a true projection: if $\mathbf{x}$ does not lie on the MLS surface, then $P(\mathbf{x})$ will usually not either.

The sphere is described by a vector $\mathbf{u}$ of 5 parameters, defining the implicit function

$$S_{\mathbf{u}}(\mathbf{x}) = \begin{pmatrix}\mathbf{x}^T & \mathbf{x}^T\mathbf{x} & 1\end{pmatrix}\mathbf{u}. \tag{6}$$

The surface of the sphere is the level set $S_{\mathbf{u}}(\mathbf{x}) = 0$. The gradient of the implicit function is

$$\nabla S_{\mathbf{u}}(\mathbf{x}) = \begin{pmatrix}I_3 & 2\mathbf{x}\end{pmatrix}\mathbf{u}. \tag{7}$$

For each surfel with position $\mathbf{p}_i$ and normal $\mathbf{n}_i$, we have two constraints:

$$S_{\mathbf{u}}(\mathbf{p}_i) = 0 \tag{8}$$

$$\nabla S_{\mathbf{u}}(\mathbf{p}_i) = \mathbf{n}_i. \tag{9}$$

The normal constraints are essential, because the position constraints are trivially satisfied by $\mathbf{u} = \mathbf{0}$. Guennebaud et al. [2008] solve for the normal constraints first to determine four of the coefficients, and then use the position constraints only to determine u_4 (which effectively determines the radius of the sphere). They give the following explicit formulae:

$$u_3 = \frac{(\sum w_i)(\sum w_i\mathbf{p}_i^T\mathbf{n}_i) - (\sum w_i\mathbf{p}_i)^T(\sum w_i\mathbf{n}_i)}{(\sum w_i)(\sum w_i\mathbf{p}_i^T\mathbf{p}_i) - (\sum w_i\mathbf{p}_i)^T(\sum w_i\mathbf{p}_i)} \tag{10}$$

$$\begin{pmatrix}u_0\\u_1\\u_2\end{pmatrix} = \frac{\sum w_i\mathbf{n}_i - 2u_3\sum w_i\mathbf{p}_i}{\sum w_i} \tag{11}$$

$$u_4 = \frac{-\begin{pmatrix}u_1 & u_2 & u_3\end{pmatrix}\sum w_i\mathbf{p}_i - u_3\sum w_i\mathbf{p}_i^T\mathbf{p}_i}{\sum w_i}. \tag{12}$$

2.5 Isosurface extraction

Isosurface extraction is the process of creating an explicit surface representation (usually a triangular or polygonal mesh) by sampling an implicit surface. Most current schemes derive from Marching Cubes [Lorensen and Cline, 1987]. The Marching Cubes algorithm samples the implicit function on a cubic lattice, and independently triangulates each cube. Each vertex of the cube is classified as either inside or outside the surface, leading to $2^8 = 256$ topological cases (although this number can be greatly reduced by symmetries). The vertices of the triangulation are determined by interpolation along the edges of the cube.

One of the major shortcomings of the Marching Cubes algorithm is that there are two possible ways to connect vertices on a face if one pair of opposite corners lies inside and the other pair lies outside the surface. Without extra processing, it is possible that this will be done inconsistently for the two cubes adjoining a face, leading to topological errors [Nielson and Hamann, 1991].

The Marching Tetrahedra (or Marching Tetrahedrons) algorithm addresses the ambiguity by using tetrahedral rather than cubic cells. Several variations of the scheme exist, depending on how space is divided into tetrahedra. The original scheme [Payne and Toga, 1990] starts with the same cubic lattice as Marching Cubes, but subdivides each cube into five tetrahedra, one larger than the others. Cubes can also be subdivided into six identical tetrahedra, as shown in Figure 3. Chan and Purisima [1998] introduce an alternative tessellation based on a body-centred cubic lattice, which has more regularly-shaped tetrahedra.

A weakness of the Marching Cubes/Tetrahedra algorithms is that the sampling rate is uniform, rather than adapted to the detail level of the extracted isosurface. The body of literate on adaptive methods is too large to fully describe here, so we will just list some examples. Treece et al. [1999] adapt the Marching Tetrahra algorithm to cluster vertices that are close together, while maintaining the topology. This removes poorly conditioned triangles, but the sampling rate is still essentially uniform. Kazhdan et al. [2007] start with an octree rather than a uniform lattice, and triangulate each octree cell in a way that preserves topology where cells of different sizes meet.

3 Implementation

3.1 Normal estimation

A rough approximation to the normals at the sample points is done from the raw scan data. The sampling grid is triangulated by connecting nearest neighbors, and discontinuities are handled by rejecting triangles with edges longer than a threshold (this is based on the implementation in Scanalyze Curless and Levoy [1996]). The sample normals are then estimated as the average of the face normals of the incident triangles. This seems to produce acceptable results, and has the advantage that normals are always correctly oriented towards the scanner. Nevertheless, it may be interesting to compare this to a more general and robust method such as that of Guennebaud and Gross [2007].

3.2 Sample spacing estimation

A key input in many surface reconstruction techniques is an estimate of the sampling density. Each sample should influence its local neighborhood, but not have unexpected effects on unrelated parts of the surface.

Some authors have used a single estimate of sample spacing for an entire model, but this is not suitable for our range scans. Both the distance and orientation of the surface relative to the scanner have a large effect on the sampling density. We compute a local sample spacing for each scanned sample, using the same formula as Meshlab [Cignoni et al., 2008]. Let the distance to the Kth-nearest neighbor be D. The local sample spacing is set to

$$r_i = \frac{2D}{\sqrt{K}}. \tag{13}$$

K should be chosen to be small enough to make the nearest neighbor search efficient but large enough to smooth out noise; we have chosen $K = 16$.

While this provides adequate results in well-sampled regions, outliers are problematic. Because they are far away from the rest of the data, r_i becomes excessively large. Since r_i determines the region of influence of sample i, this causes outliers to have much greater influence than other points, when ideally they should have no influence at all. Figure 7a shows an example of an artefact where such a sample has created a second sheet behind a wall. In Figure 7b we have clamped the sample spacing to 20mm, which corrects the artefact.

However, this approach has two disadvantages. Firstly, it requires human intervention to specify the parameter appropriately, which may require expert knowledge. Secondly, the parameter is global, and there may not be a single value that is both large enough to avoid distorting correct results and small enough to eliminate the artefacts.

We have modelled only isotropic sampling, but when scanning a plane that is angled relative to the scanner, the sampling will be anisotropic. Adamson and Alexa [2006] show how this can be incorporated into a MLS surface definition. While we have not implemented this method, it is independent of our other design choices and so it could easily be added.

3.3 Sphere fitting

We base our surface definition on Algebraic Point Set Surfaces. Initial experiments with the implementation in Meshlab [Cignoni et al., 2008] suggested that it gave reasonable results, and the Meshlab implementation of Robust Implicit Moving Least Squares [Öztireli et al., 2009] did not produce noticeably better results, even in areas with large amounts of environmental noise.

We found the equations in Section 2.4 to be numerically unstable, even with double-precision arithmetic. The denominator in Equation (10) may be much smaller than the two terms in the subtraction, yielding an inaccurate result. This occurs because the coordinates in $\mathbf{p}_i$ can be many orders of magnitude larger than the local sample spacing.

To avoid this problem, we instead describe the sphere relative to a local coordinate system, which is a translation of the world coordinate system. If $\hat{\mathbf{o}}$ is the origin of this coordinate system (in world coordinates), then we use an implicit function of the form

$$\hat{S}_{\mathbf{u},\hat{\mathbf{o}}}(\mathbf{x}) = S_{\mathbf{u}}(\mathbf{x} - \hat{\mathbf{o}}). \tag{14}$$

Given $\hat{\mathbf{o}}$, we can compute $\mathbf{u}$ by first translating all the samples to the local coordinate system and then solving for $S_{\mathbf{u}}$ as before.

This still leaves the choice of $\hat{\mathbf{o}}$. The choice has no algebraic effect and only determines the numeric stability. An attractive choice is to use the weighted mean of the samples, $\frac{\sum w_i \mathbf{p}_i}{\sum w_i}$. This would cause the right-hand terms of both the numerator and denominator of Equation (10) to vanish, entirely eliminating this source of numeric instability. The disadvantage is that the weighted mean cannot be determined until after all the samples have been collected, requiring a second pass over the samples to do the translation. Instead, we take $\hat{\mathbf{o}}$ to be the position of one of the samples. This maintains the desirable property that all the samples are near to $\hat{\mathbf{o}}$, while allowing all the samples to be processed in a single pass.

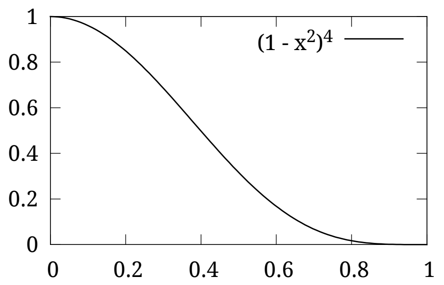


Figure 1: The weight function ϕ used to give more influence to nearby samples

3.4 Weight function

In the sphere-fitting process, each surfel is given a weight. We initially followed Guennebaud and Gross [2007] in using the function

$$w_i(\mathbf{x}) = \phi\left(\frac{\|\mathbf{p}_i - \mathbf{x}\|}{r_i \cdot h}\right) \tag{15}$$

$$\phi(d) = \begin{cases} (1-d^2)^4 & \text{if } d < 1 \\ 0 & \text{otherwise,} \end{cases} \tag{16}$$

where r_i is the local sample spacing of sample i and h is a global tuning factor controlling smoothing. Figure 1 shows ϕ.

This weight function works well if the local sample spacing varies smoothly across the surface, but we found that it causes problems where there is a discontinuity. This typically occurs at the boundary of a high-detail scan, where the neighboring region has much lower detail in the scanning coverage. The samples from the edge of the low-detail region have a large radius of influence, and so their influence spills over into the high-detail region and dominates the effect of the high-detail samples. To adjust for this, we follow Cuccuru et al. [2009] in using the inverse of the sampling density as a quality metric, defining the weight function as

$$w_i(\mathbf{x}) = \frac{1}{r_i}\phi\left(\frac{\|\mathbf{p_i} - \mathbf{x}\|}{r_i \cdot h}\right). \tag{17}$$

This reduces the weight of samples with a large radius of influence relative to those with a smaller radius of influence. Because only the relative values of the weights are important, regions with a homogeneous local sample spacing will be relatively unaffected by this change.

To avoid undersampled regions, we do not attempt to compute a fit to fewer than four surfels (following Guennebaud and Gross [2007]); at least two are required to properly constrain the sphere-fitting problem.

The weight function is C_3 continuous. While that is good from a theoretical point of view (since it avoids discontinuities in the surface), the presence of near-zero values causes additional numerical instabilities when fitting spheres. If all but one of the surfels has a weight extremely close to zero, then those surfels have almost no contribution to the design matrix and so it becomes ill-conditioned. To avoid this, we have reduced the radius of influence of each sample by 1% without adjusting the weight function. This does cause a discontinuity in the weight function at the boundary, but of such small magnitude that it does not cause discontinuities in the extracted isosurface.

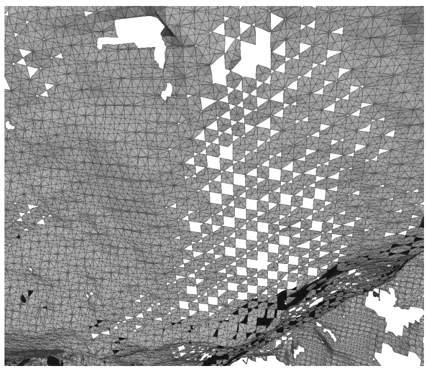


Figure 2: The effect of making h too small relative to the isosurface extraction lattice.

A limitation of this family of weight functions is that a single parameter h controls both the smoothing and the support of the function. A large value for h will thus fill holes and widen the domain of the projection operation, but blur out details; while a small value will preserve small features but leave more holes in the model. Figure 2 shows what happens when h is too small: the domain of the projection operator becomes too thin to be properly sampled by the lattice used in isosurface extraction, causing regular patterns of holes to appear.

3.5 Isosurface extraction

Although a point set surface can be used directly as a surface representation format for storage and rendering [Alexa et al., 2003, Adamson and Alexa, 2003], the implied surface is closely tied to the exact details of the algorithm that define it. A point set surface is unlikely to be displayed the same way by any two rendering implementations, which makes it of limited use as an archival record. In contrast, triangle mesh representations are an explicit representation of the geometry, and so are more suitable for archival and interchange.

To extract a mesh, we first define an implicit function in a domain near the MLS surface which approximates the signed distance to the surface (positive on the outside, negative on the inside). We then extract the isosurface where this implicit function is zero.

We have used the *almost orthogonal projection* proce-

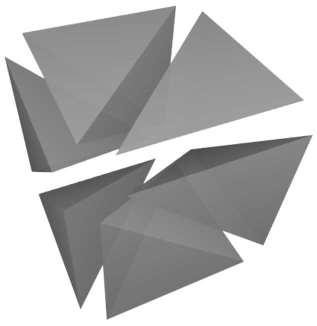


Figure 3: Partition of a cube into six tetrahedra

dure of Alexa et al. [2004] to find a nearby point on the MLS surface: given an initial point $\mathbf{x}$, we set $\mathbf{q}_0 = \mathbf{x}$ and then iteratively refine the projection as follows:

1. Compute the parameters of the algebraic sphere $S_{\mathbf{u}(\mathbf{q}_i)}$.
2. Let $\mathbf{q}_{i+1}$ be the projection of $\mathbf{x}$ onto this sphere.

This is repeated until the process converges to within a user-specified tolerance. We then locally approximate the surface by the tangent plane to the most recently fitted sphere, and compute the signed distance from $\mathbf{x}$ to the surface as $(\mathbf{x} - \mathbf{q}_\infty) \cdot \mathbf{n}$.

In our initial implementation, we used the iterative projection procedure a fixed number of times. We found that features such as grass that could not properly be resolved by the scanner would cause the projection procedure to become chaotic and fail to converge. In our current implementation, if projection fails to converge within a user-specified number of iterations, the starting point is treated as being outside the domain of the signed distance function.

We used a Marching Tetrahedra algorithm in which each cube in a cube lattice is split into six tetrahedra, as shown in Figure 3. This avoids the topological ambiguities in Marching Cubes. It also uses more interpolations per cube than Marching Cubes, without requiring additional samples of the implicit function. We felt this to be an advantage, since the implicit function is very expensive to evaluate.

In a standard isosurface extraction, the implicit function always returns either a positive or negative distance, and so the resulting surface is a manifold with no holes. Since we wanted to produce holes in our model where there was no scanner coverage, we needed to adapt the algorithm. We allowed the implicit function to take on a special “undefined” value, indicating that the sample fell outside the domain of the function. This integrates quite naturally with the finite support used for the samples: when sampling far away from the surface, the sample point will not be influenced by enough samples to produce a fitted sphere, and so the function does not have a defined value anyway. When a vertex of a tetrahedron takes on an undefined value, it is skipped and so produces no triangles.

In fact, most samples in a cubic lattice produce the undefined value, because the function only has support near the scanned surface. We found that a significant amount of time was spent on processing these samples, since each one required a walk through a spatial data structure to check for intersections with the spheres of influence. To reduce this time, we perform a forward pass over the spheres of influence to produce a list of samples that are near the surface and hence worth evaluating.

Initially this was stored in a Standard Template Library set container, but we found that this took an excessive amount of storage. We changed this to a simple bitmap of potentially useful cells, into which the bounding cubes of the spheres were rasterised. Although this bitmap is effective due to the very low cost per sample (one bit), it will scale poorly: both storage and iteration costs are $O(N^3)$ for an $N \times N \times N$ sampling grid. Further investigation is required into alternative spatial data structures such as octrees, k-d trees, lists of ranges along scan lines, etc.

The implicit function is expensive to compute, and so it is useful to evaluate it just once per lattice vertex and re-use the result across multiple incident cells. In the original work on Marching Cubes, the authors achieve this by storing the values for two slices at a time, and exploit the linear iteration order to avoid any unnecessary computations. While this would also work for our implementation, we wished to allow for greater flexibility. For example, if the cells of interest are held in an octree, then a depth-first walk of the octree would be a more natural iteration order. Greater flexibility would also simplify parallelisation, where the iteration order would be non-deterministic.

To facilitate this flexibility, we instead used a direct-mapped cache. The cache size is chosen to be the smallest power of two that will hold at least one slice of data. Hash collisions occasionally cause data to be evicted which is still needed later, but we found that this increased the number of function evaluations by less than 1% in most cases and only 7% in the worst case.

For further mesh processing, it is desirable to have a connected mesh rather than a “triangle soup” in which each triangle is independent. When processing all cells in a linear order it is relatively straightforward to associate each generated vertex with a canonical cell and to generate its index only when processing that cell. When skipping known-empty cells this becomes more complicated, and parallel processing would make it even more difficult. To avoid these problems, we process each cell independently and emit a triangle soup, and use a post-process to identify and merge shared vertices.

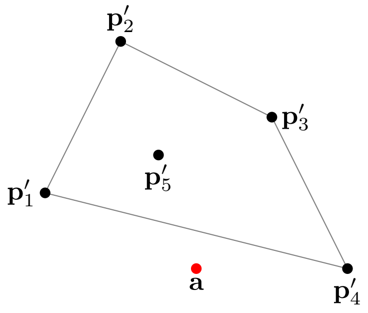


Figure 4: Boundary detection heuristic. The samples $\mathbf{p}_1$ to $\mathbf{p}_5$ form the local neighborhood. Their projections $\mathbf{p}'_1$ to $\mathbf{p}'_5$ onto the reference plane are shown, along with the projection $\mathbf{a}$ of the initial point. In this case, $\mathbf{a}$ falls outside the convex hull so it is deemed to lie beyond the boundary of the surface.

3.6 Boundary handling

While movable objects can be scanned from all directions, possibly leaving only a few small holes, this is not possible for a building. At the very least the underside of a building cannot be scanned, and it is not always practical to scan the roof. Thus, we expect our models to have large boundaries.

To a certain extent, these boundaries will arise naturally due to the finite support of the weighting function: in regions far from any scanned samples, the implicit function will be undefined. Nevertheless, we found that the surface tended to be extrapolated beyond the actual coverage of the data, and in some cases holes are extruded in unnatural ways: see Figure 9. We used a simple algorithm to detect this situation: after a point has been almost orthogonally projected onto the MLS surface, we take the local samples, namely those whose spheres of influence contain the projected point, and project them onto a plane. Ideally we would use a tangent plane to the MLS surface, but finding it is an expensive procedure [Alexa et al., 2004], so we approximate it by taking the tangent plane to the algebraic sphere fitted to the local samples in the final iteration of projection. If the projected point falls within the convex hull of the projected samples, it is considered to be internal to the surface; otherwise it is considered to have been extrapolated. See Figure 4 for an example of the latter.

This is very similar to the *angle criterion* used by Bendels et al. [2006] and others, with a critical angle of π. As the authors have described, the method suffers from artefacts and in particular is not robust to outliers that fall outside the real boundary. Their work includes several other criteria for boundary detection which are more robust but tend to produce a less sharp boundary; adapting these techniques is future work.

The boundary detection defines a two-valued function over the MLS surface: points inside have the value 1 and points outside have the value -1. To combine this with the isosurface extraction, we take the initial triangles that are produced and clip them as follows:

1. The boundary function is evaluated at the three vertices.

2. The triangle is clipped to the halfspace $f(\mathbf{x}) > 0$ where f is the linear interpolation of the boundary function between the vertices.

The clipping procedure yields either 0, 1 or 2 triangles.

The binary nature of the function tends to cause some aliasing. While aliasing can be a major issue in polygonization algorithms, in this case the aliasing is only in the shape of the hole boundary and does not affect the positions or normals of the remaining geometry.

To amortise the cost of evaluating the boundary function, it is computed once for each vertex introduced into a cell, and then cached so that it is not recomputed for neighboring cells.

3.7 Isolated components

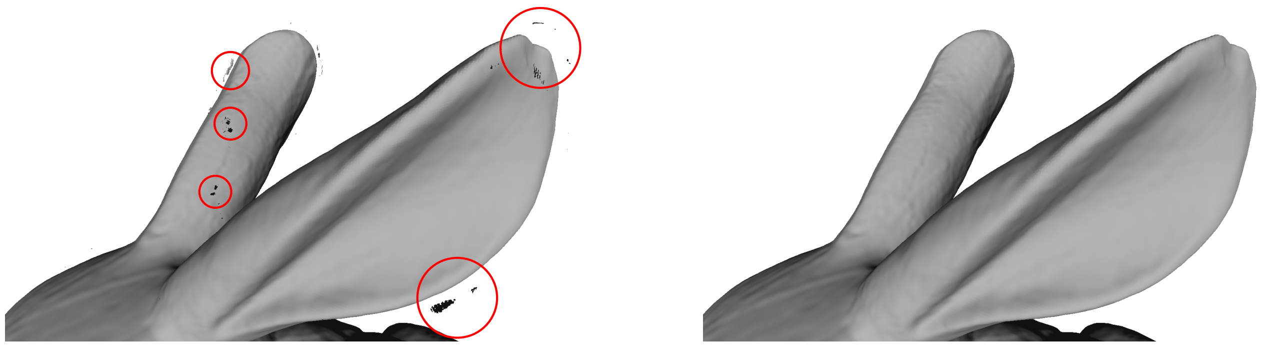
Amenta and Kil [2004a] show that the MLS projection procedure is unreliable when started too far from the surface, and may lead to unwanted zeros. Figure 5a shows an example of this. We follow Meshlab in post-processing the result to delete any disconnected components whose size (measured in vertices) falls below a threshold. This is usually successful, as shown in Figure 5b, but if the threshold is not correctly chosen it can lead to the accidental removal of useful data if a scanning campaign scans disconnected regions. We have also found problems if the lattice used for isosurface extraction is too coarse, leading to these unwanted regions remaining connected to the real surface and thus not being culled.

4 Results

For our testing we used four data sets. The *Gereza* (Figures 6, 7) *Cave* (8 top, 9) and *Namora2* (8 bottom) scans are from African heritage sites, while *Bunny* (5) is the well-known Stanford clay bunny.

Except where otherwise noted, distances are reported in millimetres and times in seconds. In table headings, *Grid* refers to the spacing between the lattice points used by the Marching Tetrahedra algorithm and h is the global smoothing parameter in Equation (17).

Figure 6 compares our implementation to a number of existing packages. Geomagic [Edelsbrunner et al., 1998] is a commercial package based on Delaunay tetrahedralization of the data points. PlyMC [Callieri et al., 2003] is



(a) Before removal (b) After removal

Figure 5: Removal of isolated components

a free implementation of the volumetric method of Curless and Levoy [1996]. RIMLS is the Meshlab implementation of Robust Implicit Moving Least Squares [Öztireli et al., 2009]. OctreeMerge is an out-of-core implementation of a Moving Least Squares technique based on the extremal surface definition [Fiorin et al., 2007], and Poisson is an out-of-core implementation of the Poisson equation method [Bolitho et al., 2007].

The Poisson method gives noticably different results to the others, because it produces a water-tight mesh without holes. While filling small holes is often desirable, it also fills in very large holes (such as the top of the wall, which has not been scanned), giving results that may diverge significantly from the physical object.

4.1 Sample spacing

Table 1 shows the parameters we have used for each data set. Because the bunny data set contained fairly clean data, we did not need to clamp sample spacing estimates. All the other models benefitted from it. Figure 7 shows an example where limiting the sample spacing is important.

4.2 Projection

Section 3.5 describes how we used a bitmap of potentially useful cells to quickly skip the bulk of cells during isosurface extraction. Table 2 shows the effectiveness of this approach. Note that while this did eliminate at least 90% of the cells in most cases, we still found that nearly half of all signed distance queries were rejected because there were fewer than four samples in the neighborhood.

In Section 3.5 we described our iterative scheme to determine the distance of an arbitrary point from the MLS surface. We found that a single iteration is usually sufficient to give accurate results, with differences being visible mainly where the input samples are sparse or suffer from environment noise. This confirms previous findings that increasing the number of iterations makes little visual difference [Cuccuru et al., 2009]. Figure 8 shows the impact of using extra iterations. Except where otherwise noted, all results in this report use only one iteration.

4.3 Boundary handling

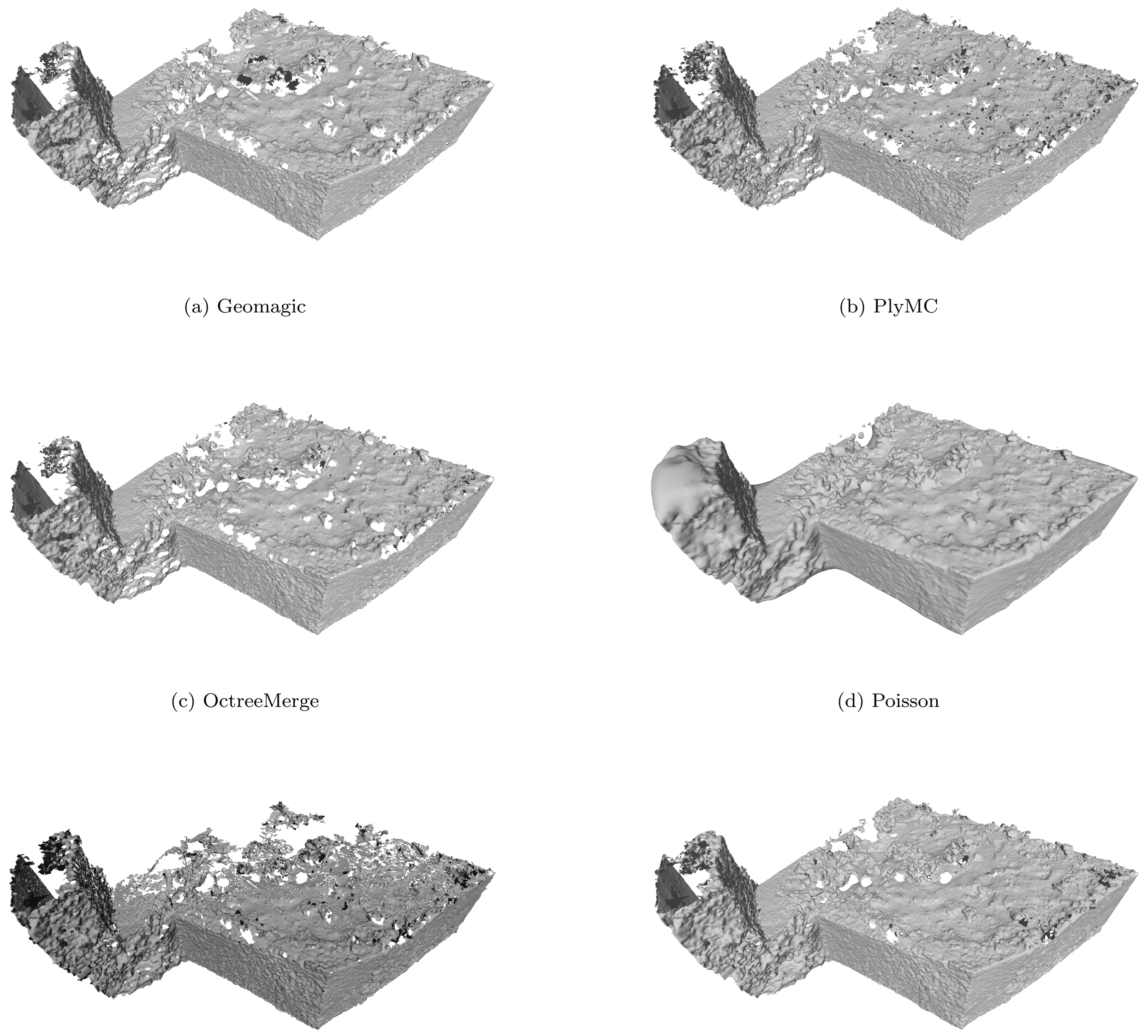
Figure 9 shows an example where explicit boundary handling prevents artefacts. Without the explicit boundary handling, boundaries are extrapolated beyond the support of the data.

Table 3 shows the number of boundary queries across all the models. Note that unlike the number of signed distance queries (Table 2), the number of boundary queries scales only slowly with the smoothing factor h. This is because the boundary queries are only performed on vertices that lie on the MLS surface, rather than for all lattice points in the domain. Increasing h does cause boundaries to be extrapolated further in the initial MLS surface, which explains both the slight increase in the number of queries and the decrease in pass rate as h increases.

The bunny model has only a few small holes, so it is unsurprising that it has the highest pass rate. The other models all have open borders and hence lower pass rates.

4.4 Numerical stability

Single-precision (32-bit) floating-point values take only half the space of double-precision values; and depending on the hardware, single-precision operations can be significantly faster than double-precision ones. To determine whether our implementation is sufficiently numerically stable to use single precision, we used C++ templates to allow all the computations to be run with either single- or double-precision, and compared the results.



(a) Geomagic (b) PlyMC

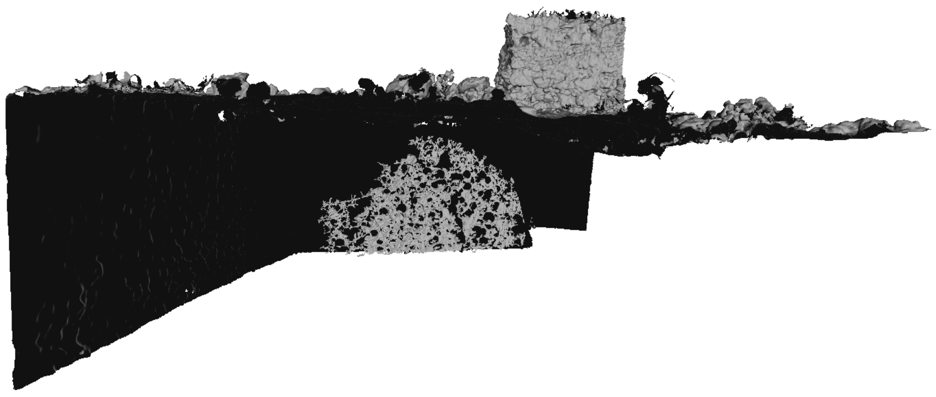
(c) OctreeMerge (d) Poisson

(e) RIMLS (f) Our implementation

Figure 6: Gereza model, reconstructed using different software packages

Table 1: Sample spacing estimation parameters. The times reported include all file I/O.

		Sample spacing				
Dataset	**Samples**	**Mean**	**Limit**	**Clamped**	**Time**	**Grid**
Bunny	362 230	0.44	N/A	0.00%	2.5	0.3
Cave	163 158	18.0	50	1.19%	1.1	10.0
Gereza	1 889 889	5.9	20	0.17%	15.8	10.0
Namora2	1 260 338	16.1	50	0.83%	10.0	20.0



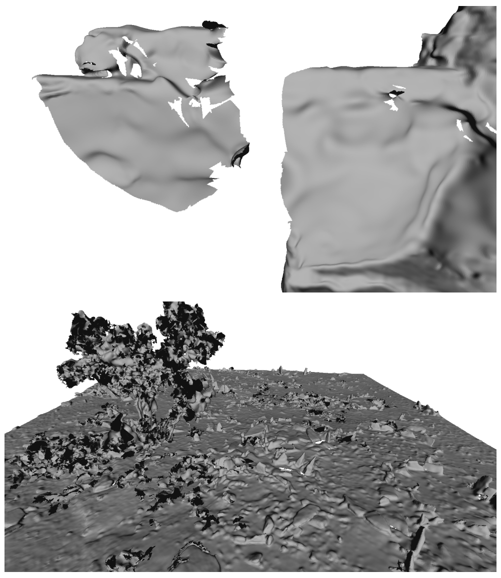
(a) No sample spacing limit. A few isolated samples with poorly estimated sample spacing cause the inside of a wall to form a noisy circular inside face.

(b) The sample spacing is limited, preventing the isolated samples from interfering.

Figure 7: Limiting the sample spacing to prevent isolated samples from causing artefacts.

Table 2: Projection operation

		Cells		Projections	
Dataset	**h**	**Memory**	**Used**	**Total**	**Succeeded**
Bunny	2	13.8 MB	5.3%	7115122	52.4%
	4	14.5 MB	10.6%	13831456	62.0%
Cave	2	35.1 MB	3.3%	13337256	48.3%
	5	39.8 MB	9.5%	33380653	56.7%
Gereza	2	19.5 MB	2.4%	5030025	43.6%
	5	20.8 MB	6.5%	12919956	59.0%
Namora2	4	24.7 MB	6.6%	15012900	53.3%
	6	26.1 MB	10.0%	29243329	57.8%


1 iteration

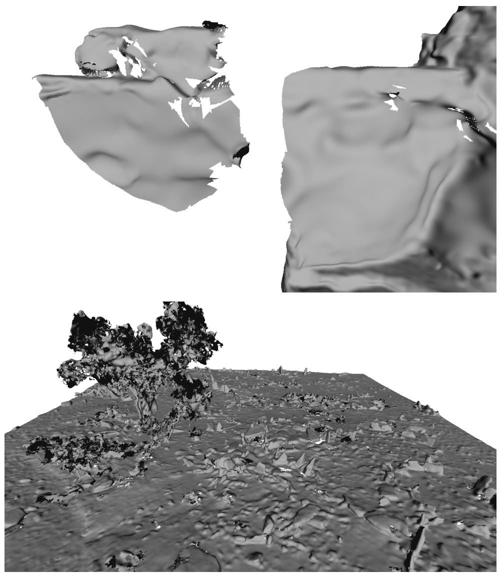
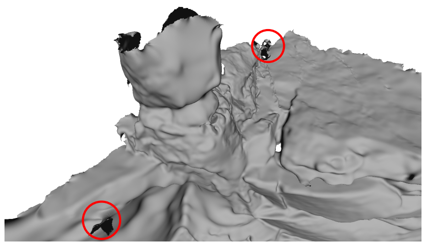

15 iterations

Figure 8: Impact of the number of projection iterations. Using extra iterations marginally reduces some of the artefacts.


(a)

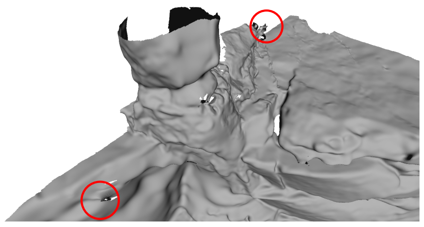

(b)

Figure 9: Explicit boundary removal. (a) No explicit boundary detection is done. The boundaries are extrapolated, causing extra noise (top) and unnatural extrusion of some holes (bottom-left). (b) Boundaries are detected and removed, correcting artefacts but also preventing some hole-filling.

Table 3: Boundary detection. The columns show the total number of boundary detection queries and the fraction of queries for which the query point is inside the boundary.

Dataset	h	Queries	Inside
Bunny	2	3 197 179	98.6%
	4	3 186 845	97.3%
Cave	2	2 481 672	93.1%
	5	3 052 561	86.2%
Gereza	2	3 176 214	92.8%
	5	4 236 312	81.3%
Namora2	4	5 863 472	81.7%
	6	6 033 051	74.9%

Table 4: Distances between meshes produced by single- and double-precision computations.

Dataset	Grid	h	Hausdorff	RMS
Bunny	0.3	4	0.002	0.000
Cave	10.0	5	7.2	0.008
Gereza	10.0	5	6.3	0.006
Namora2	20.0	4	20.6	0.022

Table 4 shows the Hausdorff distance between the single- and double-precision versions, as well as the larger of the forward and reverse root-mean-square (RMS) distances reported by Metro [Cignoni et al., 1998]. As can be seen from the table, the maximum error introduced by the use of lower precision is of a similar size to and usually smaller than the spacing of the lattice used for Marching Tetrahedra, and the average error is three orders of magnitude smaller. We thus feel confident that our implementation is numerically stable at single precision.

4.5 Performance

It should be emphasised that our implementation is a prototype designed to experiment with various algorithmic improvements. It is single-threaded and has not had any low-level optimizations applied. The performance figures are reported so that the relative effects of the various parameters can be assessed, but should not be taken as an absolute indicator of performance.

By comparing the Cave and Gereza data sets in Table 5, it should be clear that running time is relatively unaffected by the number of input samples. Rather, it is mostly affected by the number of cells to process and the smoothing factor h. Apart from increasing the number of cells to process, increasing h also increases the number of samples that contribute to each sphere fit.

The boundary detection significantly increases running time, in some cases by 50%. With such a large number of boundary queries this is not surprising, but it is unfortunate given that the majority of queries are inside the boundary.

5 Conclusions

While there has been extensive work on point set surfaces, few authors have tackled the extremely noisy data caused by the constraints of African heritage sites. We have evaluated a number of existing techniques that aim to improve the quality of MLS surfaces, and found that:

- Sample spacing estimation cannot be done only based on nearest neighborhood information, but specifying an upper bound on the estimated sample spacing greatly improves results.
- Boundaries need to be handled explicitly.
- Using more than one iteration of the almost orthogonal projection operation makes almost no difference in well-sampled regions, while removing some artefacts in problematic regions.

6 Future work

Surface reconstruction is a mature field with a wealth of techniques, many of which we have not explored. For example, we have not discussed normal estimation at all, and it is possible that a refinement of the initial normal estimates, such as described by Guennebaud and Gross [2007], may produce better results. There are also more sophisticated methods for boundary detection and for handling anisotropic sampling that may be valuable.

Estimation of sampling density is not widely discussed in the MLS surface literature, and there is clearly room for improvement, particularly with regard to detecting outliers. Sample spacing could be estimated earlier in the pipeline, exploiting knowledge of the properties of the scanner to give more accurate results. For example, knowing the angular sampling rate of the scanner, the distance from the scanner to the point, and the orientation of the surface relative to the scanner, it would be possible to directly determine the spatial sampling density. The angular density could also be estimated given the position of the scanner.

There is also room for improvement of the weight function. Ideally we would like to find a two-parameter family of functions with one parameter to control smoothness and a separate parameter to control support. That would allow the degree of hole-filling to be controlled independently from smoothness.

Our implementation is in-core, but for a solution to scale to huge amounts of data produced by scanning an entire site it must be able to handle terabytes of data.

Table 5: Performance of our reconstruction. Times exclude the time for sample spacing estimation, which are shown in Table 1.

Dataset	**Samples**	**h**	**Projections**	**Boundary queries**	**Time**	**Time (without boundary handling)**
Bunny	362 230	2	7 115 112	3 197 179	27	20
		4	13 831 456	3 186 845	67	49
Cave	163 158	2	13 337 256	2 481 673	29	24
		5	33 380 653	3 052 561	115	97
Gereza	1 889 889	2	5 030 025	3 176 214	30	21
		5	12 919 956	4 236 312	93	62
Namora2	1 260 338	4	15 012 900	5 863 472	85	56
		6	29 243 329	6 033 051	188	136

Our findings offer some guidance for the design of out-of-core algorithms. Each projection step in projecting onto the MLS surface requires the samples in the local neighborhood to be gathered. When multiple iterations are used, the intermediate steps could be located anywhere in space, making data management difficult; whereas with only a single projection step, the required data can be determined *a priori*. A pre-declared maximum sample spacing is also useful for out-of-core algorithms, as it defines a maximum neighborhood size that needs to be resident in memory.

7 Acknowledgements

Funding for this project was received from the Center for High Performance Computing (CHPC). The bunny model is from the Stanford 3D Scanning Repository, while the other models are from the African Cultural Heritage and Landscape Database.

References

Anders Adamson and Marc Alexa. Ray tracing point set surfaces. In *Proceedings of the Shape Modeling International 2003*, pages 272–279. IEEE Computer Society, 2003. ISBN 0-7695-1909-1.

Anders Adamson and Marc Alexa. Anisotropic point set surfaces. In *Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, AFRIGRAPH '06, pages 7–13. ACM, 2006. ISBN 1-59593-288-7.

M. Alexa, S. Rusinkiewicz, Marc Alexa, and Anders Adamson. On normals and projection operators for surfaces defined by point sets. In *Eurographics Symposium on Point-Based Graphics*, pages 149–155, 2004.

Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Point set surfaces. In *Proceedings of the conference on Visualization '01*, pages 21–28, 2001. ISBN 0-7803-7200-X.

Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9 (1):3–15, January 2003. ISSN 1077-2626.

Nina Amenta and Yong J. Kil. The domain of a point set surface. In Markus Gross, Hans-Peter Pfister, Marc Alexa, and Szymon Rusinkiewicz, editors, *SPBG'04 Symposium on Point-Based Graphics*, pages 139–147. Eurographics Association, 2004a. ISBN 3-905673-09-6.

Nina Amenta and Yong Joo Kil. Defining point-set surfaces. *ACM Trans. Graph.*, 23(3):264–270, August 2004b.

Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, SMA '01, pages 249–266. ACM, 2001. ISBN 1-58113-366-9.

Gerhard H. Bendels, Ruwen Schnabel, and Reinhard Klein. Detecting holes in point set surfaces. *Journal of WSCG*, 14, Feb 2006. ISSN 1213-6972.

Matthew Bolitho, Michael Kazhdan, Randal Burns, and Hugues Hoppe. Multilevel streaming for out-of-core surface reconstruction. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 69–78. Eurographics Association, 2007. ISBN 978-3-905673-46-3.

Matthew Bolitho, Michael Kazhdan, Randal Burns, and Hugues Hoppe. Parallel Poisson surface reconstruction. In *International Symposium on Visual Computing*, 2009.

M. Callieri, P. Cignoni, F. Ganovelli, C. Montani, P. Pingi, and R. Scopigno. VCLab's tools for 3D range data processing. In D. Arnold, A. Chalmers, and F. Niccolucci, editors, *4th International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage (VAST)*, 2003.

S.L. Chan and E.O. Purisima. A new tetrahedral tesselation scheme for isosurface generation. *Computers & Graphics*, 22(1):83–90, 1998. ISSN 0097-8493.

P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, 1998. ISSN 1467-8659.

Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an open-source mesh processing tool. In *Sixth Eurographics Italian Chapter Conference*, pages 129–136, 2008.

Gianmauro Cuccuru, Enrico Gobbetti, Fabio Marton, Renato Pajarola, and Ruggero Pintus. Fast low-memory streaming MLS reconstruction of point-sampled surfaces. In *Proceedings of Graphics Interface 2009*, pages 15–22. Canadian Information Processing Society, 2009. ISBN 978-1-56881-470-4.

Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 303–312. ACM, 1996. ISBN 0-89791-746-4.

Herbert Edelsbrunner, Michael A. Facello, Ping Fu, Jiang Qian, and Dmitry V. Nekhayev. Wrapping 3D scanning data. In Richard N. Ellson and Joseph H. Nurre, editors, *Three-Dimensional Image Capture and Applications*, volume 3313 of *SPIE Proceedings*, pages 148–158. SPIE, 1998. ISBN 0-8194-2753-5.

Valentino Fiorin, Paolo Cignoni, and Roberto Scopigno. Out-of-core MLS reconstruction. In *Proceedings of the Ninth IASTED International Conference on Computer Graphics and Imaging*, CGIM '07, pages 27–34, 2007. ISBN 978-0-88986-645-4.

Valentino Fiorin, Paolo Cignoni, and Roberto Scopigno. Practical and robust MLS-based integration of scanned data. In *Sixth Eurographics Italian Chapter Conference*, pages 57–64, 2008.

Shachar Fleishman, Daniel Cohen-Or, and Cláudio T. Silva. Robust moving least-squares fitting with sharp features. *ACM Trans. Graph.*, 24(3):544–552, July 2005.

Gaël Guennebaud and Markus Gross. Algebraic point set surfaces. *ACM Trans. Graph.*, 26(3), July 2007. ISSN 0730-0301.

Gaël Guennebaud, Marcel Germann, and Markus Gross. Dynamic sampling and rendering of algebraic point set surfaces. *Computer Graphics Forum*, 27(2):653–662, 2008. ISSN 1467-8659.

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics Symposium on Geometry Processing*, SGP '06, pages 61–70. Eurographics Association, 2006. ISBN 3-905673-36-3.

Michael Kazhdan, Allison Klein, Ketan Dalal, and Hugues Hoppe. Unconstrained isosurface extraction on arbitrary octrees. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 125–133. Eurographics Association, 2007. ISBN 978-3-905673-46-3.

David Levin. Mesh-independent surface interpolation. In Brunnett, Hamann, and Mueller, editors, *Geometric Modeling for Scientific Visualization*, pages 37–49. Springer-Verlag, 2003.

William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '87, pages 163–169. ACM, 1987. ISBN 0-89791-227-6.

Gregory M. Nielson and Bernd Hamann. The asymptotic decider: resolving the ambiguity in marching cubes. In *Proceedings of the 2nd conference on Visualization '91*, VIS '91, pages 83–91. IEEE Computer Society Press, 1991. ISBN 0-8186-2245-8.

A. Cengiz Öztireli, Gaël Guennebaud, and Markus Gross. Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum*, 28(2):493–501, 2009. ISSN 1467-8659.

B.A. Payne and A.W. Toga. Surface mapping brain function on 3d models. *IEEE Computer Graphics and Applications*, 10(5):33–41, Sep 1990. ISSN 0272-1716.

Heinz Rüther, Christoph Held, Roshan Bhurtha, Ralph Schröder, and Stephen Wessels. Challenges in heritage documentation with terrestrial laser scanning. In *Proceedings of the 1st AfricaGEO Conference*, 2011.

G.M. Treece, R.W. Prager, and A.H. Gee. Regularised marching tetrahedra: improved iso-surface extraction. *Computers & Graphics*, 23(4):583–598, 1999. ISSN 0097-8493.

Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, SIGGRAPH '94, pages 311–318. ACM, 1994. ISBN 0-89791-667-0.

Kun Zhou, Minmin Gong, Xin Huang, and Baining Guo. Data-parallel octrees for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 17:669–681, 2011. ISSN 1077-2626.