Multiresolution Point-set Surfaces

François Duranleau* Philippe Beaudoin Pierre Poulin

LIGUM, Dép. d'informatique et de recherche opérationnelle
Université de Montréal

ABSTRACT

Multiresolution representations of 3D surfaces make it possible to concentrate the efforts of a modification at the appropriate level of detail. This paper introduces a multiresolution representation for point-set surfaces. At each level, the point set is smoothed and downsampled, and the geometric details are encoded along the smoothed surface normal. The resulting structure is only slightly larger than the original point set and allows to reconstruct it precisely. We demonstrate how it can be used for surface deformation and for frequency band scaling.

Keywords: Multiresolution surface representations, point-sampled geometry, shape modeling, deformation.

Index Terms: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

1 INTRODUCTION

Representing 3D surfaces as point sets has gained popularity in many applications. This popularity can be explained by the simplicity, flexibility, and efficiency of such a representation. In fact, numerous techniques have already been proposed to adapt point-set surfaces for efficient rendering, modeling, and animation [20, 13].

A well-studied alternative approach to 3D surface representations lies in mesh-based structures, for which multiresolution techniques have already been proposed. Compared to static fine-grained meshes, these multiresolution representations offer the key benefit to select level of detail for editing the shape, allowing for detail-preserving surface deformations, and speeding up detail-preserving global manipulations.

Although we expect that similar benefits could be gained by using a multiresolution representation of point sets, it is difficult to lift results from mesh-related research and apply them to point-set surfaces. This is mainly because multiresolution meshes rely heavily on the connectivity information available within the structure. However, the fact that this information is not needed is one of the desirable characteristics of point sets. Multiresolution point sets must therefore be built solely upon positions and take into account that point positions can change relative to one another during a manipulation process.

This paper proposes a multiresolution representation of point-set surfaces that does not require the introduction of a global connectivity information. We focus on reconstructing the original point set, not a resampled version of it. Our representation works well even for point sets that contain fine shapes, such as thin fingers on an otherwise large character or small bumps on a surface. Finally, as opposed to previous work in the field [38, 48], the coarsest levels are downsampled point sets such that the size of the final structure is slightly larger than the original point set (roughly 133% the original size). To demonstrate the benefits of such a representation, we explore various manipulations that are made more efficient by the proposed representation.

*e-mail: { duranlef, beaudoin, poulin } @iro.umontreal.ca

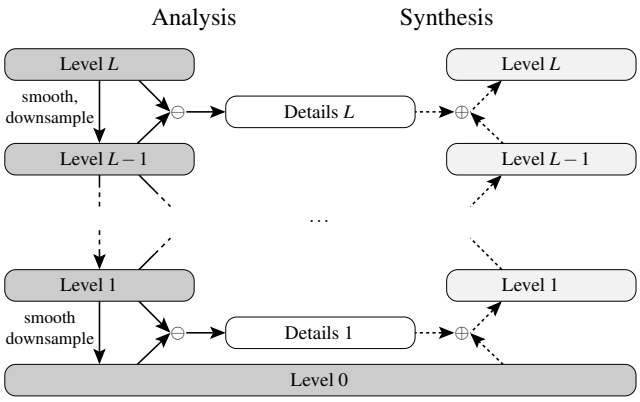


Figure 1: Multiresolution as a series of analysis-synthesis steps.

1.1 Overview

In its broadest sense, multiresolution surfaces can be seen as the combination of an analysis and a synthesis process. The goal of a single analysis step is to generate a coarser surface while extracting and storing the lost details relative to this new surface. Multiple analysis steps can be applied until a coarse enough approximation, the lowest level surface, is obtained. Starting from the lowest level we can iteratively synthesize finer surfaces by reapplying the details. A well-chosen representation for extracted details should make it possible to modify a lower level and see this modification propagate up towards the detailed surfaces.

The multiresolution point sets introduced in this paper follow the overview schematized in Figure 1. Let P^l be the point set at the resolution level l. To generate a coarser level, i.e., a point set P^{l-1}, we first smooth the point set and then downsample it to reduce the number of points. This downsampling does not introduce sampling artifacts because high frequencies are previously removed by the smoothing step.

Each point of P^l is then stored relative to the surface of P^{l-1} first by projecting it on that surface. This projected point is reformulated relative to the lower level point set P^{l-1}, similarly to the method of Boubekeur et al. [10], so that a modification to this level correctly propagates up. Finally, for each point of P^l, we store its displacement over the coarser surface in normal direction plus the information required for the reformulation of its projection.

From a given level $l-1$, synthesis of a point at level l proceeds by first reconstructing the associated point on the coarse surface, computing the normal and applying the displacement value along this direction.

This paper is organized as follows. We begin in Section 2 by reviewing previous work in the fields of mesh-based multiresolution surfaces and multiresolution point-based representations. We then go through the details of the proposed surface analysis scheme (Section 4), followed by the synthesis scheme (Section 5). Results are then presented and discussed in Section 6. Finally, we conclude the paper and suggest directions for future research.

2 Previous Work

2.1 Mesh-based Multiresolution Surfaces

Multiresolution techniques have been largely used in the domain of subdivision surfaces due to the intrinsic hierarchical property of the subdivision procedure [11, 30, 49]. Two types of multiresolution techniques are commonly used to encode the difference from one level to another. One type subsamples the detail vertices and only records a detail per odd vertex, which is similar to the wavelet transform. The other type encodes the detail vectors over all the vertices. Depending on the goal of the multiresolution representation construction, i.e., a coarser approximation or a smoother approximation of the original geometry, two different techniques are presented by Kobbelt et al. [21] for constructing the multiresolution hierarchy from arbitrary triangle meshes. Hubeli and Gross [19] have generalized these ideas to non-manifold meshes.

Other techniques are based on displacement over a smooth subdivision surface [24] or on a sequence of displacements in normal directions of successive levels [18].

Multiresolution techniques contain a wide class of applications, e.g., polyhedral compression, continuous level-of-detail control, compression of a function defined on surfaces, multiresolution editing of surfaces and surface optimization [30], as well as signal processing [17].

2.2 Point-based Multiresolution Representations

The notion of multiresolution in point-based geometry has often been used to actually describe hierarchical structures, mostly for rendering purposes, e.g., [39, 41, 8, 32, 34, 33, 44]. In our case, we use the term multiresolution in the same sense as in multiresolution meshes to describe surfaces that can be edited at different levels of scale.

Among these hierarchical structures, the bounding sphere hierarchy of QSplat [41] (and all follow-up work) comes a little closer to our notion, as it encodes points in a sphere relative to its center, but it is not locally independent of rotation or scale and so does not allow for manipulation at a coarse level.

Progressive point-set surfaces [12, 42] are closer in spirit to our notion of multiresolution, but they still lack the possibility to edit the surface at lower resolution levels, mostly due to either a dependence on implicit k-neighborhoods during synthesis [12], or detail encoding that is not independent of rotation and scaling [42].

The fan-based representation of Linsen and Prautzsch [27], which is also a progressive technique, allows for a multiresolution representation in our sense. They propose to store, for each point, a one-ring neighborhood with a minimal relative angle criterion. When reducing the point set, they basically replace a point with the difference between itself and its neighborhood's centroid. Synthesis simply reverses the operation.

Multiscale point-set surface representations [38, 48] also correspond to our notion of multiresolution surfaces. These techniques represent a surface by a series of smoother and smoother point sets, although each level contains as many points as the original set. This mimics the mesh-smoothing-based decomposition of Kobbelt et al. [21]. Our technique extends this representation by allowing the number of points to also be reduced.

3 Preliminaries

In this section, we present some concepts and notations that will be used throughout the paper.

Moving-least-squares (MLS) surfaces have been introduced by Levin [25, 26] and popularized in computer graphics by Alexa et al. [6]. Since then, many variants have been proposed [5, 7, 22, 40, 3, 2, 46, 16, 28, 29, 4]. Our multiresolution representation does not depend on a particular variant, as long as an orthogonal (or quasi-orthogonal) projection operator to bring a point on the surface exists. In our implementation, we use the variant of Alexa and Adamson [5], which we briefly describe in this section.

Given a finite point set $P \subset \mathbb{R}^3$ and a point $\mathbf{x} \in \mathbb{R}^3$, we note as $\mathscr{N}_P(\mathbf{x}) \subset P$ the set of the k nearest neighbors of $\mathbf{x}$ in P. Let $\mathbf{p}_i \in P$ denote the ith point of P. Then, following Alexa and Adamson [5], a normal estimation at $\mathbf{x}$ over P, noted as a vector valued function $\mathbf{n}_P(\mathbf{x})$, can be computed as the direction of smallest weighted covariance of P in $\mathbf{x}$, i.e., the eigenvector of smallest eigenvalue of

$$\sum_i (\mathbf{x}-\mathbf{p}_i)(\mathbf{x}-\mathbf{p}_i)^{\mathsf{T}} \phi(\|\mathbf{x}-\mathbf{p}_i\|) \tag{1}$$

where ϕ is a positive monotonous decreasing function (typically a Gaussian). Alternately, if for each $\mathbf{p}_i \in P$ a normal $\mathbf{n}_i$ is provided, we can define $\mathbf{n}_P(\mathbf{x})$ as the weighted average of all normals, that is

$$\mathbf{n}_P(\mathbf{x}) = \frac{\sum_i \phi(\|\mathbf{x}-\mathbf{p}_i\|)\mathbf{n}_i}{\|\sum_i \phi(\|\mathbf{x}-\mathbf{p}_i\|)\mathbf{n}_i\|}. \tag{2}$$

If the given normals are reliable, this is more robust than using covariance analysis as it is less sensitive to point distribution.

The point set P defines an implicit surface $\mathscr{S}_P$ as follows:

$$\mathscr{S}_P = \left\{ \mathbf{x} \in \mathbb{R}^3 \;\middle|\; \mathbf{n}_P(\mathbf{x})^{\mathsf{T}} \big(\mathbf{x} - \mathbf{a}_P(\mathbf{x})\big) = 0 \right\}$$

where

$$\mathbf{a}_P(\mathbf{x}) = \frac{\sum_i \phi(\|\mathbf{x}-\mathbf{p}_i\|)\mathbf{p}_i}{\sum_i \phi(\|\mathbf{x}-\mathbf{p}_i\|)} \tag{3}$$

is the weighted centroid around $\mathbf{x}$.

We use adaptive weighting as described by Pauly et al. [37], that is, $\phi(d) = \mathrm{e}^{-d^2/h_{\mathbf{x}}^2}$, where $h_{\mathbf{x}}$ is one third the distance between $\mathbf{x}$ and the farthest point in $\mathscr{N}_P(\mathbf{x})$. Thus the summation index i in Equations (1)-(3) is taken among the indices of each point of $\mathscr{N}_P(\mathbf{x})$.

An import aspect of MLS surfaces is the existence of a projection operator $\Psi_P(\mathbf{x})$ that takes a point $\mathbf{x}$ close to $\mathscr{S}_P$ and projects it onto $\mathscr{S}_P$. It can be implemented as an iterative process that repeatedly projects $\mathbf{x}$ onto the plane passing by $\mathbf{a}_P(\mathbf{x})$ perpendicular to $\mathbf{n}_P(\mathbf{x})$. An "almost" orthogonal variant always projects $\mathbf{x}$ of the first iteration.

4 Analysis

Analysis, or decomposition, consists of decomposing a point set P^l at a resolution level l into a smoother, coarser point set and corresponding geometric details. It can be formulated as follows:

$$P^{l-1} = \Lambda\Big(\Phi\big(P^l\big)\Big) \tag{4}$$

$$D^l = P^l \ominus P^{l-1} \tag{5}$$

where Φ is a smoothing operator, Λ is a downsampling operator, and $\ominus$ extracts the higher frequency detail values. Details essentially comprise displacement values of each point of P^l over $\mathscr{S}_{P^{l-1}}$ plus some partial topological information to locate their projection on $\mathscr{S}_{P^{l-1}}$. Note that in our representation, $|P^l| = |D^l|$. The full analysis for L levels repeats this decomposition for $l = L, L-1, \ldots, 1$, where P^L is the most detailed point set.

4.1 Generating the Coarser Level

The smoother, downsampled coarser point set P^{l-1} is computed in two stages: a smoothing operator Φ first reduces the high frequencies in the point set P^l, then a downsampling operator Λ computes a reduced point set representative of the smoother point set.

As Pauly et al. [38] and Zhang et al. [48], we construct a multiresolution MLS surface. Thus, a natural choice for Φ is the MLS projection operator Ψ_{P^l}. However, in their case, the number of points is constant for all levels. So, in order to reduce computation times, they fix the size of k-neighborhoods for all levels and

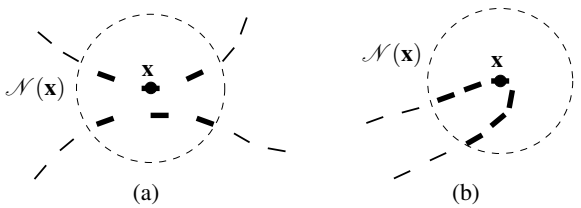


Figure 2: Typical 6-neighborhoods around a point $\mathbf{x}$, that are problematic for computing $\mathbf{n}(\mathbf{x})$ or for downsampling. Points with normals are represented as short segments.

they downsample P^l by an increasing factor, which enlarges the smoothing kernel. In our case, the number of points decreases, thus basically, no downsampling is required. However, to obtain a significant smoothing between levels, we would still have to choose a large value for k. Consequently, we also downsample P^l prior to smoothing, but by a constant factor for all levels. This leads to a trivial choice for Λ, that is, the same downsampling operator, which ensures that the downsampling rate is coherent with the smoothing operation. It is better to downsample the smoothed point set instead of taking the downsampled point set used for smoothing as it reduces sampling artifacts [38].

It is important to note that the core of our method (detail extraction in Section 4.2) does not depend on a particular smoothing or downsampling technique, and we could use a variety of such techniques for smoothing, e.g., [36, 23, 48, 38, 45, 31] or downsampling, e.g., [37, 12, 27, 9, 42, 47, 43]. However, since we rely on MLS projection and estimation of normal directions during detail extraction, some care must be taken when downsampling. Figure 2 illustrates the problem: for any given k-neighborhood around a point $\mathbf{x}$, the neighbors should fall on a local patch of surface around $\mathbf{x}$. This means that, with no explicit topology, point sets cannot be uniformly downsampled without risking collapses. For instance, in Figure 2(a), two opposing pieces of a surface could collapse into a line.

In our current implementation, the downsampling operator Λ uses a hierarchical clustering method [37], which has the advantage of offering an intuitive control on the downsampling rate by specifying a target cluster size. However, some modifications were necessary to take into account the condition on k-neighborhoods. Pauly et al. [37] suggest an adaptive clustering technique based on the notion of surface variation, which is based on covariance analysis on clusters' content. This is useful only if the clusters have a significant size. In our case, we typically downsample by a factor of 4 or 5, which can be roughly obtained by setting similar cluster sizes. At this scale, covariance analysis is unreliable. Also, the k-neighborhood condition should be fulfilled in the resulting point set, so analysing the content of clusters does not really help.

We thus implemented an additional refinement step. Clustering is first run as usual. Let C be the set of computed clusters, $\bar{\mathbf{c}}_i$ be the centroid of the ith cluster, noted C_i, and $\check{\mathbf{n}}_i$ its normal, computed as the normalized average of all normals associated to each point in C_i. Let also $\bar{C} = \{\bar{\mathbf{c}}_i\}_{i=0}^{|C|-1}$, i.e., the set of all cluster centroids. Then for each $\bar{\mathbf{c}}_i$, we check if $\mathscr{N}_{\bar{C}}(\bar{\mathbf{c}}_i)$ satisfies the condition given in the next paragraph. If the condition is not satisfied, we tag C_i to be split, unless $|C_i| = 1$. After all clusters are checked, we split all tagged clusters, following the hierarchical clustering partitioning [37]. We repeat this process until no clusters need to be split, and the final $\bar{C}$ corresponds to the resulting point set.

To test the condition, we verify that $\mathscr{N}_{\bar{C}}(\bar{\mathbf{c}}_i)$ satisfies the following:

$$\min_{\bar{\mathbf{c}}_j \in \mathscr{N}_{\bar{C}}(\bar{\mathbf{c}}_i)} \check{\mathbf{n}}_j{}^{\mathsf{T}} \check{\mathbf{n}}_i > \tau \quad \text{or} \quad \frac{\lambda_0}{\lambda_1} < \nu$$

where τ is the tolerance on normal deviation in a k-neighborhood, $\lambda_0 < \lambda_1$ are the two smallest eigenvalues of the weighted covariance matrix of Equation (1), and ν is essentially a threshold on the neighborhood's "sharpness". This is akin to the surface variation of Pauly et al. [37], but tends to better catch sampling problems in thin creases.

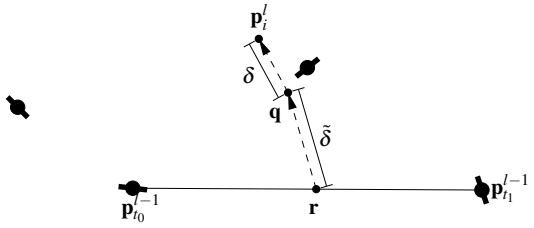


Figure 3: Detail extraction. A point $\mathbf{p}_i^l \in P^l$ is projected on $\mathscr{S}_{P^{l-1}}$ (short segments represent points in P^{l-1}), giving point $\mathbf{q}$ and resulting in the geometric detail value δ. Then $\mathbf{q}$ is reformulated on a base triangle (illustrated here as the segment $(\mathbf{p}_{t_1}^{l-1} - \mathbf{p}_{t_0}^{l-1})$) as $\mathbf{r} + \tilde{\delta}\mathbf{n}(\mathbf{r})$.

4.2 Detail Extraction

Once the coarser level is generated (Section 4.1), we have both point sets P^l and P^{l-1}. The corresponding details are noted as $D^l = P^l \ominus P^{l-1}$, which roughly represents the frequency band of geometric details [48, 38].

Our main problem is to coherently compute and store D^l with the points produced by the upsampling operator during synthesis (Section 5), that is, there must be a one-to-one matching correspondence between them, whether or not P^{l-1} has been deformed. This is not a problem with meshes because the topological information is explicit. Point sets have an implicit, or geometry-dependent, topology. Thus, if P^{l-1} is modified, neighborhoods change, and any upsampling operation based on implicit neighborhoods [6, 12, 14, 15, 31] has no guarantee to reproduce a point set P^l such that $|P^l| = |D^l|$, with a clear correspondence between detail elements in D^l and points in P^l. Therefore, we have to store some form of topological information together with the details.

Similarly to Linsen and Prautzsch [27], we could handle this problem by explicitly storing the full k-neighborhood in P^{l-1} of a given point in P^l, and the corresponding detail value would be the difference vector between the point and the neighborhood's centroid. This requires k indices and three scalars for each element of D^l. However, as we will see, we can do better based on an intrinsic reformulation similar to that of Boubekeur et al. [10].

Detail extraction basically consists of two steps. First, geometric detail information is computed by taking the *difference* between $\mathscr{S}_{P^l}$ and $\mathscr{S}_{P^{l-1}}$. This difference can be expressed as displacement of each point of P^l over $\mathscr{S}_{P^{l-1}}$, which is computed using $\Psi_{P^{l-1}}$. Let Q be the point set resulting in the projection of all points of P^l on $\mathscr{S}_{P^{l-1}}$. The second step consists in reformulating each point of Q intrinsically in P^{l-1} to be able not only to reconstruct Q during synthesis, but also a deformed version in accordance to a deformation of P^{l-1}.

More specifically, for each point $\mathbf{p}_i^l \in P^l$, we compute its corresponding detail information $\mathbf{d}_i^l$ using the following steps:

1. Compute the projection $\mathbf{q}$ of $\mathbf{p}_i^l$ on $\mathscr{S}_{P^{l-1}}$. The geometric detail value δ is the signed distance (according to the surface's orientation) between them.

2. Find a set of three points $\{\mathbf{p}_{t_0}^{l-1}, \mathbf{p}_{t_1}^{l-1}, \mathbf{p}_{t_2}^{l-1}\} \subset P^{l-1}$ forming a triangle $\mathscr{T}$ that *encloses* $\mathbf{q}$.

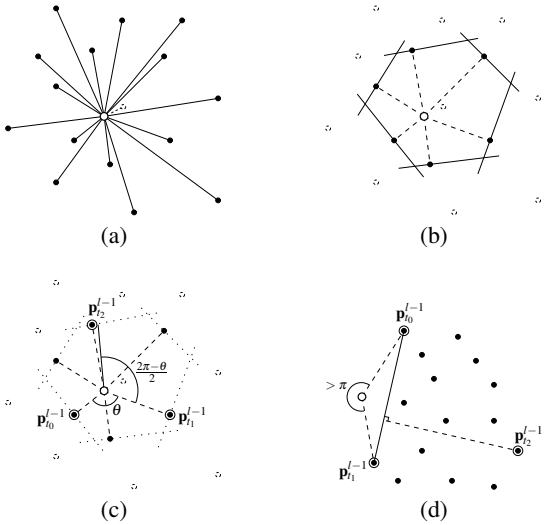


Figure 4: Base triangle selection around $\mathbf{q}$ (hollow dot in the center, or left in (d)). (a) Initial candidates without one point considered too close. (b) Conversion into a BSP-neighborhood. (c) Selection of $\{\mathbf{p}_{t_0}^{l-1}, \mathbf{p}_{t_1}^{l-1}, \mathbf{p}_{t_2}^{l-1}\}$. (d) Boundary case.

3. Find a point $\mathbf{r}$ on $\mathscr{T}$'s supporting plane using a special non-orthogonal projection and compute its barycentric coordinates $(\beta_0, \beta_1, \beta_2)$ in $\mathscr{T}$.

4. Set

$$\mathbf{d}_i^l = (t_0, t_1, t_2, \beta_1, \beta_2, \delta). \tag{6}$$

There is no need to store all three barycentric coordinates because $\beta_0 = 1 - \beta_1 - \beta_2$ (the choice of which two we keep is completely arbitrary). Optionally, a seventh value can be added to the tuple $\mathbf{d}_i^l$: $\tilde{\delta}$, the signed distance between $\mathbf{q}$ and $\mathbf{r}$. Also, instead of storing δ, we can store $\delta/\Delta_{\mathscr{T}}$, where $\Delta_{\mathscr{T}}$ is the area of $\mathscr{T}$. Reasons and explanations for these options are provided in Section 5.

Finally, $D^l = \{\mathbf{d}_i^l\}_{i=0}^{|P^l|-1}$. Figure 3 illustrates the detail information we extract. Steps 1, 2, and 3 are further explained in the following sections. Note that for all operators and functions described in Section 3 and used afterwards, the point set on which they are computed is always P^{l-1}. Thus, we omit the point set in subscript, which is implicitly understood to be P^{l-1}.

4.2.1 Geometric Detail Value

The fundamental information of geometric detail is the displacement of points of P^l over $\mathscr{S}_{P^{l-1}}$. For a point $\mathbf{p}_i^l \in P^l$, the geometric detail value δ is computed as follows:

$$\delta = \mathbf{n}(\mathbf{q})^\mathsf{T} \left(\mathbf{p}_i^l - \mathbf{q}\right)$$

where $\mathbf{q} = \Psi(\mathbf{p}_i^l)$. Ideally, for δ to really represent a displacement value, Ψ should compute an orthogonal projection. Alexa and Adamson [5] did present such a projection operator. However, in practice, we can rely on their "almost" orthogonal variant briefly described in Section 3. It is faster and, according to our experience, the loss of precision is not significant enough.

4.2.2 Base Triangle Selection

After the computation of δ, we have in hand point $\mathbf{q}$. The next step is to reformulate $\mathbf{q}$ *intrinsically* in P^{l-1}. Boubekeur et al. [10] proposed a method that expresses a point in terms of barycentric coordinates in a triangle formed by three neighbors. The point and the triangle are projected on a locally fitted plane prior to compute the barycentric coordinates. We use a similar idea, however several modifications were required to fit our needs.

The basic idea is to find a set of three points $\{\mathbf{p}_{t_0}^{l-1}, \mathbf{p}_{t_1}^{l-1}, \mathbf{p}_{t_2}^{l-1}\} \subset P^{l-1}$ that forms a triangle $\mathscr{T}$ that *encloses* $\mathbf{q}$, and then to reformulate $\mathbf{q}$ in terms of barycentric coordinates $(\beta_0, \beta_1, \beta_2)$ in $\mathscr{T}$ of its projection on $\mathscr{T}$'s supporting plane, i.e.

$$\mathbf{q} = \tilde{\delta}\mathbf{n}_{\mathscr{T}} + \sum_{j=0}^{2} \beta_j \mathbf{p}_{t_j}^{l-1} \tag{7}$$

where $\mathbf{n}_{\mathscr{T}}$ is $\mathscr{T}$'s normal and $\tilde{\delta}$ is the displacement of $\mathbf{q}$ over $\mathscr{T}$. We say that $\mathscr{T}$ encloses $\mathbf{q}$ if all barycentric coordinates are positive. Note in $\mathbf{d}_i^l$, we actually store the indices t_0, t_1, t_2 of $\mathscr{T}$'s vertices.

Then the next problem is to actually find $\mathscr{T}$'s three vertices. As reported by Boubekeur et al. [10], for improved robustness, $\mathscr{T}$ should be as small and equilateral as possible. For reasons explained later, we also require that $\mathbf{n}_{\mathscr{T}}$ should not be very different than the normal of points in an area of $\mathscr{S}_{P^{l-1}}$ close to $\mathbf{q}$. All these constraints together make the problem ill-defined; we must rely on some heuristics.

For selecting our triangle, we first compute a reference plane $\mathscr{H}$ passing through $\mathbf{q}$ perpendicular to $\mathbf{n}(\mathbf{q})$. Next, to help keep $\mathbf{n}_{\mathscr{T}}$ close to its constraints, we filter out of $\mathscr{N}(\mathbf{q})$ all points for which their projection on $\mathscr{H}$ is closer than $(1+\varepsilon)r$ from $\mathbf{q}$, where r is the distance of the farthest point in $\mathscr{N}(\mathbf{q})$ from $\mathbf{q}$ (Figure 4(a)). The relative tolerance ε can be quite large (we use a value of 0.1). Next, to enforce a smaller size for $\mathscr{T}$, the result is converted into a BSP-neighborhood [35] (Figure 4(b)), and then sorted by increasing angle around $\mathbf{q}$ [27], where the closest point is set to angle 0. Angles are computed from projections on $\mathscr{H}$.

We finally have to select three points among the remaining candidates for which all their relative angle is as close to $\frac{3\pi}{2}$ as possible. However, to avoid an exhaustive search, we arbitrarily fix the closest neighbor as the first vertex. We then scan the set of candidates in increasing order of angle around the center and select the point with the relative angle to the first vertex closest to $\frac{3\pi}{2}$ as the second vertex. To select the last vertex, we continue to scan and choose the point with a relative angle closest to $\frac{2\pi-\theta}{2}$, where θ is the relative angle between the second and first vertices. Figure 4(c) shows an example of the selection procedure's final step. For all examples in Figure 4, to simplify the illustration, all points are understood as being locally coplanar.

For point sets with boundaries, if $\mathbf{q}$ lies close to a boundary, the BSP-neighborhood could be degenerated, that is, it may have less than three neighbors or could lead to sliver triangles. Most importantly, a majority of points on boundaries will fall outside of $\mathscr{T}$ by a significant distance. For greater robustness, we have to resort to a different selection algorithm in these cases, which are detected if the BSP-neighborhood has less than three neighbors or if the maximum relative angle is greater than π.

The general idea of the selection for boundaries is to find a triangle such that only one barycentric coordinate is negative, and the other two are relatively small. We also want to keep the triangle with a good aspect ratio to minimize the impact of stretching. What we do is simply to set the first two vertices of $\mathscr{T}$ as the two neighbors in $\mathscr{N}(\mathbf{q})$ for which their relative angle is maximal (usually greater than π), and set the third as the neighbor in $\mathscr{N}(\mathbf{q})$ that has the maximum orthogonal projection distance on the segment $(\mathbf{p}_{t_1}^{l-1} - \mathbf{p}_{t_0}^{l-1})$. Figure 4(d) shows an example of our triangle selection for a boundary case.

Finally, because the downsampling operator can introduce sampling density discontinuities, false boundaries could be detected. Therefore, before we apply the boundary case selection, we try again the regular case using a neighborhood of all points falling

inside a sphere centered at $\mathbf{q}$. The radius is the average radius of all points' k-neighborhood in P^{l-1}, which can be computed once after P^{l-1} is computed.

4.2.3 Non-orthogonal Projection on Triangle

If we simply reformulate $\mathbf{q}$ as in Equation (7), we can face two main problems during synthesis. First, the direction of $\mathbf{n}_{\mathscr{T}}$ might not be reliable after a deformation of P^{l-1}. Second, neighbors of $\mathbf{p}_i^l$ in P^l might have their projection on $\mathscr{S}_{P^{l-1}}$ be reformulated on the same triangle, thus any deformation of P^{l-1} will result in a piecewise linear deformation of the reconstruction. We could instead find a point $\mathbf{r}$ on the triangle $\mathscr{T}$'s supporting plane, noted $\mathscr{H}_{\mathscr{T}}$, such that $\mathbf{q} = \Psi(\mathbf{r})$, but reversing the projection procedure is far from trivial.

Instead we compute a point $\mathbf{r}$ on $\mathscr{H}_{\mathscr{T}}$ such that the line passing by $\mathbf{q}$ of direction $\mathbf{n}(\mathbf{r})$ intersects $\mathscr{H}_{\mathscr{T}}$ at $\mathbf{r}$, i.e., we compute a point $\mathbf{r}$ satisfying the following set of equations:

$$\mathbf{q} - \Big(\mathbf{n}(\mathbf{r})^{\mathsf{T}}(\mathbf{q}-\mathbf{r})\Big)\mathbf{n}(\mathbf{r}) = \mathbf{r} \tag{8}$$

$$\mathbf{n}_{\mathscr{T}}{}^{\mathsf{T}}(\mathbf{r} - \mathbf{p}_{t_0}^{l-1}) = 0. \tag{9}$$

Note that we can replace $\mathbf{p}_{t_0}^{l-1}$ by any point on $\mathscr{H}_{\mathscr{T}}$ in Equation (9).

We can combine them together by substituting $\mathbf{r}$ in Equation (9) with the left hand side of Equation (8), and then isolate the expression $\mathbf{n}(\mathbf{r})^{\mathsf{T}}(\mathbf{q}-\mathbf{r})$ in the resulting equation and substitute it back in Equation (8), which gives the following equation:

$$\mathbf{q} - \frac{\mathbf{n}_{\mathscr{T}}{}^{\mathsf{T}}(\mathbf{q}-\mathbf{p}_{t_0}^{l-1})}{\mathbf{n}_{\mathscr{T}}{}^{\mathsf{T}}\mathbf{n}(\mathbf{r})}\mathbf{n}(\mathbf{r}) = \mathbf{r}. \tag{10}$$

The left hand side corresponds to the intersection of the line passing by $\mathbf{q}$ in direction $\mathbf{n}(\mathbf{r})$ with the plane $\mathscr{H}_{\mathscr{T}}$. By choosing an appropriate initial guess for $\mathbf{r}$, we can use Equation (10) as an iterative procedure, i.e.

$$\mathbf{r}^{(t+1)} = \mathbf{q} - \frac{\mathbf{n}_{\mathscr{T}}{}^{\mathsf{T}}(\mathbf{q}-\mathbf{p}_{t_0}^{l-1})}{\mathbf{n}_{\mathscr{T}}{}^{\mathsf{T}}\mathbf{n}(\mathbf{r}^{(t)})}\mathbf{n}(\mathbf{r}^{(t)}). \tag{11}$$

This procedure tends sometimes to overshoot the solution. We can help it by damping the direction $\mathbf{n}(\mathbf{r}^{(t)})$ to the direction bisecting $(\mathbf{q}-\mathbf{r}^{(t)})/\|\mathbf{q}-\mathbf{r}^{(t)}\|$ and $\mathbf{n}(\mathbf{r}^{(t)})$ (or $-\mathbf{n}(\mathbf{r}^{(t)})$ if they are in opposite directions). Let $\bar{\mathbf{n}}(\mathbf{r}^{(t)})$ be that direction. We thus finally have

$$\mathbf{r}^{(t+1)} = \mathbf{q} - \frac{\mathbf{n}_{\mathscr{T}}{}^{\mathsf{T}}(\mathbf{q}-\mathbf{p}_{t_0}^{l-1})}{\mathbf{n}_{\mathscr{T}}{}^{\mathsf{T}}\bar{\mathbf{n}}(\mathbf{r}^{(t)})}\bar{\mathbf{n}}(\mathbf{r}^{(t)}). \tag{12}$$

This procedure will converge if $\mathbf{r}^{(0)}$ is reasonably close to the solution, and if $\mathbf{n}(\mathbf{x})$ is continuous (weighted covariance analysis is continuous, as well as normal weighted average, assuming ϕ is also) and stays reasonably close to $\mathbf{n}_{\mathscr{T}}$, thus our initial constraint on the selection of $\mathscr{T}$. We initialize the iteration with $\mathbf{r}^{(0)}$ as follows:

$$\mathbf{r}^{(0)} = \mathbf{q} - \frac{\mathbf{n}_{\mathscr{T}}{}^{\mathsf{T}}(\mathbf{q}-\mathbf{p}_{t_0}^{l-1})}{\mathbf{n}_{\mathscr{T}}{}^{\mathsf{T}}\mathbf{n}(\mathbf{q})}\mathbf{n}(\mathbf{q}) \tag{13}$$

which is the intersection of the line passing by $\mathbf{q}$ of direction $\mathbf{n}(\mathbf{q})$ and the plane $\mathscr{H}_{\mathscr{T}}$. Figure 5 illustrates an iteration of the procedure.

We stop the iterations when the error on Equation (8) is below a given threshold, that is

$$\|\mathbf{q}-\mathbf{r}^{(t)}\|^2 - \Big(\mathbf{n}(\mathbf{r}^{(t)})^{\mathsf{T}}(\mathbf{q}-\mathbf{r}^{(t)})\Big)^2 < \varepsilon^2 \tag{14}$$

for an error tolerance $\varepsilon > 0$, or if a given maximum number of iterations is reached. Very rarely the procedure has trouble with convergence. We detect convergence problems in two ways:

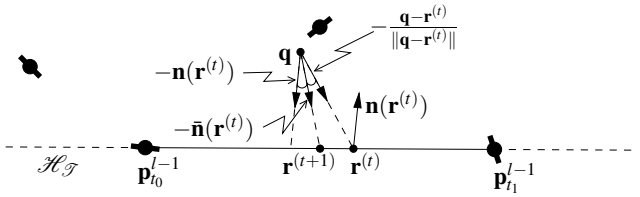


Figure 5: Geometric interpretation of the non-orthogonal iterative projection procedure of $\mathbf{q}$ on $\mathscr{H}_{\mathscr{T}}$. It seeks a point $\mathbf{r}$ on $\mathscr{H}_{\mathscr{T}}$ such $\mathbf{n}(\mathbf{r})$ is parallel to $\mathbf{q}-\mathbf{r}$. The triangle $\mathscr{T}$ is represented as the segment $(\mathbf{p}_{t_1}^{l-1} - \mathbf{p}_{t_0}^{l-1})$, and short thicker segments are points of P^{l-1}.

1. at the end of an iteration, if the error (left hand-side of Inequation (14)) increases after an iteration;
2. the maximum number of iterations has been reached and the error tolerance is still not satisfied.

In Case 1, we further damp $\bar{\mathbf{n}}(\mathbf{r}^{(t)})$ by taking again the bisecting direction between itself and $(\mathbf{q}-\mathbf{r}^{(t)})/\|\mathbf{q}-\mathbf{r}^{(t)}\|$. We repeat this kind of backtracking until the error is smaller than the previous iteration, or when another maximum number of iterations is reached. In the former case, we go on with the main iterative procedure. In the latter case, we do as Case 2. In Case 2, we jitter $\mathbf{r}^{(t)}$ on $\mathscr{H}_{\mathscr{T}}$, reset the iteration counter to zero, and start anew. We allow this jittering for a given maximum number of times.

Note that it is possible that $\mathbf{r}$ falls outside the boundaries of $\mathscr{T}$. However, in all our experiments, it occurred in no more than 2% of all cases, and when it happens, the point is never far outside.

5 Synthesis

Synthesis, or reconstruction, corresponds to the reverse operation of analysis. Starting with a point set P^{l-1}, we want to synthesize point set P^l induced by P^{l-1} and the stored details D^l, i.e.

$$P^l = \Lambda^{-1}(P^{l-1}) \oplus D^l$$

where Λ^{-1} is an upsampling operator that reproduces Q, or a deformed Q according to any deformation of P^{l-1}. The full analysis for L levels repeats this reconstruction for $l = 1, 2, \ldots, L$.

Because the upsampling operator Λ^{-1} uses information in D^l, and because all synthesized points can be computed independently, we describe the synthesis procedure in terms of processing each $\mathbf{d}_i^l \in D^l$ instead of globally upsampling and then applying displacement. For $\mathbf{d}_i^l \in D^l$, we compute the corresponding point $\mathbf{p}_i^l$ as follows:

1. Let $(t_0, t_1, t_2, \beta_1, \beta_2, \delta) = \mathbf{d}_i^l$ (Equation (6)).
2. Compute $\mathbf{r} = (1-\beta_1-\beta_2)\mathbf{p}_{t_0}^{l-1} + \beta_1\mathbf{p}_{t_1}^{l-1} + \beta_2\mathbf{p}_{t_2}^{l-1}$.
3. Compute $\mathbf{q}$ by intersecting $\mathscr{S}_{P^{l-1}}$ with the line passing by $\mathbf{r}$ in direction $\mathbf{n}(\mathbf{r})$.
4. Set $\mathbf{p}_i^l = \mathbf{q} + \delta\mathbf{n}(\mathbf{q})$.

Finally, $P^l = \{\mathbf{p}_i^l\}_{i=0}^{|D^l|-1}$, and normals for each synthesized points $\mathbf{p}_i^l$ are computed with $\mathbf{n}_{P^l}(\mathbf{p}_i^l)$, but using covariance analysis (Equation (1)). Weighted average of normals can not be used because they are not known yet.

The intersection with $\mathscr{S}_{P^{l-1}}$ in Step 3 is computed using the intersection procedure of Adamson and Alexa [1]. However, by construction of the base triangle $\mathscr{T}$ formed by points of indices t_0, t_1, t_2 in P^{l-1}, we know that $\mathbf{r}$ already lies close to $\mathscr{S}_{P^{l-1}}$, and we know

(a) Level 0 (b) Level 1 (c) Level 2 (d) Level 3 (e) Original (f) Scaling 2-2-1

Figure 6: Synthesized levels, original surface, and detail scaling for the "Armadillo" point set.

that $\mathbf{n}(\mathbf{r})$ is a direction that, from $\mathbf{r}$, should intersect $\mathscr{S}_{P^{l-1}}$. Therefore, there is no need to construct and use any additional spatial data structure. The simplified procedure can be formulated with the following iterative scheme:

$$\mathbf{q}^{(t+1)} = \mathbf{r} - \frac{\mathbf{n}(\mathbf{q}^{(t)})^{\mathsf{T}}\left(\mathbf{q}^{(t)} - \mathbf{a}(\mathbf{q}^{(t)})\right)}{\mathbf{n}(\mathbf{q}^{(t)})^{\mathsf{T}}\mathbf{n}(\mathbf{r})}\mathbf{n}(\mathbf{r}) \tag{15}$$

where $\mathbf{q}^{(0)} = \mathbf{r}$. Given an error tolerance $\varepsilon > 0$, the iterations stop when $\left|\mathbf{n}(\mathbf{q}^{(t)})^{\mathsf{T}}\left(\mathbf{q}^{(t)} - \mathbf{a}(\mathbf{q}^{(t)})\right)\right| < \varepsilon$.

We can get a faster synthesis if we accept to store an extra scalar per detail tuple. After the non-orthogonal projection on $\mathscr{T}$ (Section 4.2.3), we can also keep the value $\tilde{\delta} = \mathbf{n}(\mathbf{r})^{\mathsf{T}}(\mathbf{q}-\mathbf{r})$, also shown in Figure 3. Then, during synthesis, Step 3 is replaced by

3. Compute $\mathbf{q} = \mathbf{r} + \tilde{\delta}\mathbf{n}(\mathbf{r})$.

Note however that this trick is not exactly equivalent to the previous approach if P^{l-1} has been deformed, especially if the curvature changes. Nevertheless, the results are still satisfactory.

If we want geometric details to scale accordingly with local stretching during synthesis, then, as Boubekeur et al. [10], instead of storing δ as the sixth component of $\mathbf{d}_i^l$ in Equation (6), we store $\delta/\Delta_{\mathscr{T}}$, where $\Delta_{\mathscr{T}}$ denotes the area of $\mathscr{T}$. Then, during synthesis, Step 4 is replaced by

4. Set $\mathbf{p}_i^l = \mathbf{q} + \frac{\delta}{\Delta_{\mathscr{T}}}\Delta_{\mathscr{T}'}\mathbf{n}(\mathbf{q})$, where $\mathscr{T}'$ is the triangle of vertices $\{\mathbf{p}_{i_0}^{l-1}, \mathbf{p}_{i_1}^{l-1}, \mathbf{p}_{i_2}^{l-1}\}$ after a possible deformation of P^{l-1}.

Note that we cannot apply this scaling $\tilde{\delta}$ because it is not computed relative to $\mathscr{S}_{P^{l-1}}$, but from a triangle that roughly locally approximates the surface, and because the triangles selected for two neighbors of P^l can be sufficiently different to have non coherent $\tilde{\delta}$ values while having coherent displacement values over $\mathscr{S}_{P^{l-1}}$.

6 Results and Discussion

We have successfully used our multiresolution representation on a number of different point-set surfaces. Figures 6, 7(a), and 9 show various surfaces synthesized at various levels, and compares the highest level with the original point set. Figure 8 shows similar results, but using smaller points to make the effect of the downsampling process clearly visible. This figure clearly shows that, as described in Section 4.1, more points are maintained in potentially problematic regions such as near the chin and the area between the arm and the body. Figure 9 shows that our representation maintains the shape of boundaries even when a deformation is applied. All the results shown here use the synthesis optimization described at the end of Section 5.

These figures show that synthesized results are very close to the original. In fact, the root mean square errors of the reconstructed points are only 9.5×10^{-5}, 2.64×10^{-4}, 4.90×10^{-4}, and 1.75×10^{-4} for the "Armadillo", "Igea", "Bumps", and "Isis" point sets, respectively. These values are given in normalized units adjusted so that the largest size of the point set's bounding box equals one. Table 1 gives the number of points for each level.

	Armadillo	Igea	Bumps	Isis
Level 0	6510	500	374	1600
Level 1	15218	1512	1140	4368
Level 2	48312	4551	3583	14953
Level 3	172974	14503	10395	53883
Level 4	—	43636	32028	196256
Level 5	—	134345	97284	—

Table 1: Number of points for each synthesized level.

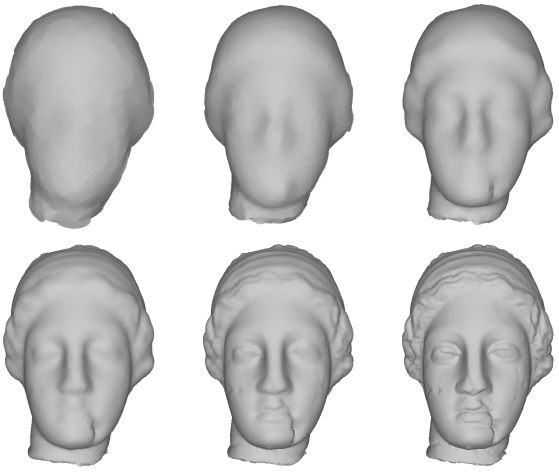
	Armadillo	Igea	Bumps	Isis
Analysis	137.4 / —	27.4 / 60.1	17.9 / 36.7	39.9 / 67.3
Synthesis	9.5 / —	7.6 / 12.2	5.0 / 6.1	10.5 / 14.4

Table 2: Computation time (in seconds) for analysis and synthesis. For each entry, the first number is the time for our technique, and the second is for our implementation of Pauly et al. [38]'s technique. All timings were obtained on an AMD Turion 64 X2 processor.

The computation times required for the complete analysis, from the original point-set surface to level 0, and complete synthesis, from level 0 to the reconstructed surface, are given in Table 2 (number on the left for each pair). The larger analysis time for the "Armadillo" point set is due to the presence of high curvature zones and fine structures in the geometric model. This slows down convergence of the iterative procedure described in Section 4.2.3.

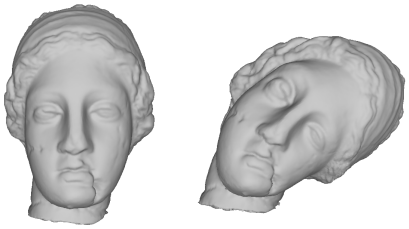
We have compared the computation times with our implementation of Pauly et al. [38]'s technique for the same number of levels and same downsampling factor (used for smoothing in their case). Timings are also shown in Table 2 (number on the right for each pair). Although for each point, our technique requires more computations, because the number of points is reduced from level to level, it is globally faster, though less significantly for synthesis. No results for the "Armadillo" point set is shown for Pauly et al.'s technique because their downsampling does not verify the neighborhood condition (Section 4.1). We could implement it, although their analysis timings would be even slower. Despite that, we still timed the execution for the "Isis" point set with their technique. One clear advantage of reducing the number of points is that processing the lower levels is much faster, and editing can be done interactively there. However, some patience is required to visualize the upper levels. They can be gradually computed as processing time permits.

We have mentioned in Section 4.2 that extracted details roughly represent frequency bands of geometric detail. As do Pauly et al. [38] and Zhang et al. [48], we can take advantage of that fact by scaling these details at different levels to emphasize or smooth out various characteristics of the surface. This creates an interesting set of manipulation tools, as demonstrated in Figures 6(f) and 7(b), where the given scaling values correspond to the scaling factor for each level, from coarsest to finest. For example, in Figure 6(f), the two lowest levels of detail of the "Armadillo" are multiplied by two, emphasizing low frequency components and inflating the



(a) Synthesized levels 0 – 5.

(b) Detail scaling: 0-0-0-1-1; 1-1-1-2-3; 2-2-1-1-1.



(c) Original, Deformed.

Figure 7: Results for the "Igea" point set.

model. As also pointed out by Pauly et al. and Zhang et al., this scaling can also be used to obtain continuous detail resolution in the same fashion.

7 Conclusion and Future Work

Point-set surfaces offer a flexible representation to simplify many tasks in geometric modeling, animation, and rendering. However it is difficult to find an efficient and robust multiresolution representation for these tasks. This is mainly due to the loss of explicit connectivity between the points.

We have presented a multiresolution representation that, at each level, smoothes the point set, downsamples the point set where special conditions are respected, and encodes efficiently the detail points so they can be reconstructed when needed. We have described a number of criteria and optimizations that allow for greater robustness. Finally, a number of transformations over different point sets have demonstrated the quality of our representation.

Reduced point sets at coarser levels allow faster processing for editing. However, our synthesis time still prevents a fully interactive use of our representation. We believe that better support for interactive editing operations could be achieved using adaptive multiresolution, similar to the work of Zorin et al. [49], where analysis and synthesis is done adaptively based on local flatness of the geometry. Also, for our technique, each point is processed independently at most stages, which makes most computations highly parallelizable. This is also true though for other point-based multiresolution representations, including multiscale representations [38, 48].

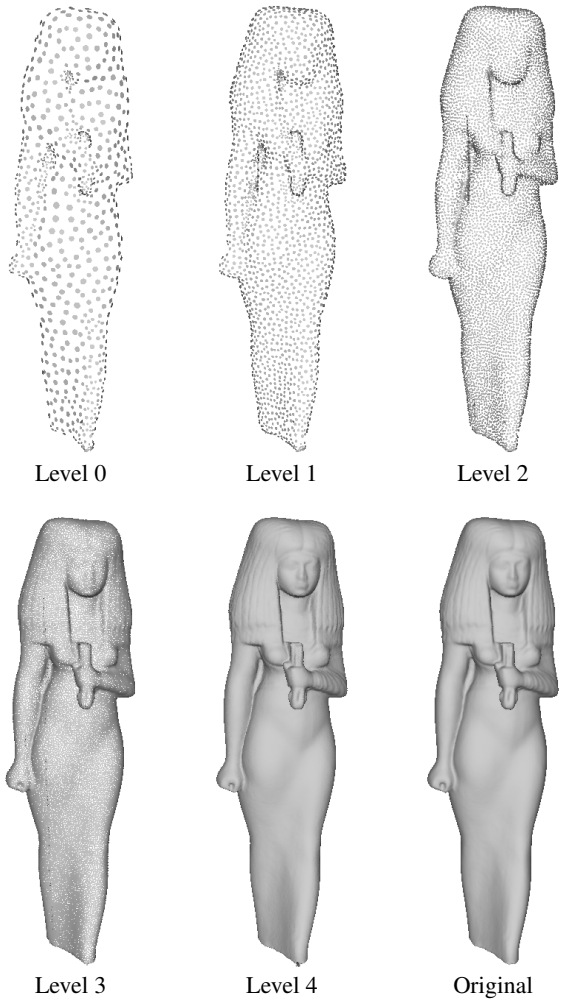


Figure 8: Synthesized levels and original surface for the "Isis" point set, with visible points.

Finally, the selection of our base triangle is an important factor in improving robustness. We have described several heuristics used to select the triangle, but there is room for improvement, especially for the boundary case.

Acknowledgements

The authors thank Di Jiang for her help and support. This work was supported in parts by grants from NSERC, FQRNT, and MITACS.

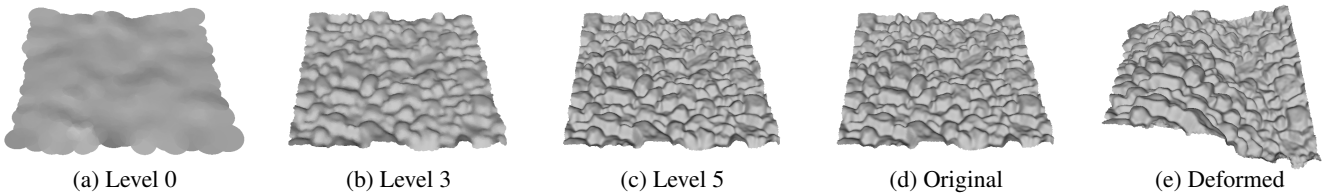
References

[1] A. Adamson and M. Alexa. Approximating bounded, non-orientable surfaces from points. In *Proc. IEEE Shape Modeling and Application*, pages 243–252, 2004.

[2] A. Adamson and M. Alexa. Anisotropic point set surfaces. In *Proc. AFRIGRAPH*, pages 7–13, 2006.

[3] A. Adamson and M. Alexa. Point-sampled cell complexes. *ACM Trans. on Graphics*, 25(3):671–680, 2006.

[4] M. Alexa and A. Adamson. Interpolatory point set surfaces — convexity and hermite data. *ACM Trans. on Graphics*. To appear.



(a) Level 0 (b) Level 3 (c) Level 5 (d) Original (e) Deformed

Figure 9: Synthesized levels and original surface for the “Bumps” point set.

[5] M. Alexa and A. Adamson. On normals and projection operators for surfaces defined by point sets. In *Proc. Symp. on Point-based Graphics*, pages 149–156, 2004.

[6] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. *IEEE Trans. on Visualization and Computer Graphics*, 9(1):3–15, 2003.

[7] N. Amenta and Y. J. Kil. Defining point-set surfaces. *ACM Trans. on Graphics*, 23(3):264–270, 2004.

[8] M. Botsch, A. Wiratanaya, and L. Kobbelt. Efficient high quality rendering of point-sampled geometry. In *Eurographics Workshop on Rendering*, pages 53–64, 2002.

[9] T. Boubekeur, W. Heidrich, X. Granier, and C. Schlick. Volume-surface trees. *Proc. Eurographics*, 25(3):399–406, 2006.

[10] T. Boubekeur, O. Sorkine, and C. Schlick. SIMOD: Making freeform deformation size-insensitive. In *Proc. Symp. on Point-based Graphics*, pages 47–56, 2007.

[11] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proc. ACM SIGGRAPH*, pages 173–182, 1995.

[12] S. Fleishman, D. Cohen-Or, M. Alexa, and C. T. Silva. Progressive point set surfaces. *ACM Trans. on Graphics*, 22(4):997–1011, 2003.

[13] M. Gross and H. Pfister, editors. *Point-based Graphics*. The Morgan Kaufmann Series in Computer Graphics. Morgan Kaufmann, 2007.

[14] G. Guennebaud, L. Barthe, and M. Paulin. Dynamic surfel set refinement for high quality rendering. *Computers & Graphics*, 28(6):827–838, 2004.

[15] G. Guennebaud, L. Barthe, and M. Paulin. Interpolatory refinement for real-time processing of point-based geometry. *Proc. Eurographics)*, 24(3):657–667, 2005.

[16] G. Guennebaud and M. Gross. Algebraic point set surfaces. *ACM Trans. on Graphics*, 26(3):23, 2007.

[17] I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. In *Proc. ACM SIGGRAPH*, pages 325–334, 1999.

[18] I. Guskov, K. Vidimce, W. Sweldens, and P. Schröder. Normal meshes. In *Proc. ACM SIGGRAPH*, pages 95–102, 2000.

[19] A. Hubeli and M. Gross. Multiresolution methods for nonmanifold models. *IEEE Trans. on Visualization and Computer Graphics*, 7(3):207–221, 2001.

[20] L. Kobbelt and M. Botsch. A survey of point-based techniques in computer graphics. *Computers & Graphics*, 28(6):801–814, 2004.

[21] L. Kobbelt, S. Campagna, J. Vorsatz, and H.-P. Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *Proc. ACM SIGGRAPH*, pages 105–114, 1998.

[22] R. Kolluri. Provably good moving least squares. In *Proc. Symp. on Discrete Algorithms*, pages 1008–1017, 2005.

[23] C. Lange and K. Polthier. Anisotropic smoothing of point sets. *Computer Aided Geometric Design*, 22(7):680–692, 2005.

[24] A. Lee, H. Moreton, and H. Hoppe. Displaced subdivision surfaces. In *Proc. ACM SIGGRAPH*, pages 85–94, 2000.

[25] D. Levin. The approximation power of moving least-squares. *Mathematics of Computation*, 67(224):1517–1531, 1998.

[26] D. Levin. *Geometric Modeling for Scientific Visualization*, chapter Mesh-Independent Surface Interpolation, pages 37–49. Springer-Verlag, 2003.

[27] L. Linsen and H. Prautzsch. Fan clouds — an alternative to meshes. In *Theoretical Foundations of Computer Vision*, pages 39–57, 2003.

[28] Y. Lipman, D. Cohen-Or, and D. Levin. Data-dependent MLS for faithful surface approximation. In *Proc. Symp. on Geometry Processing*, pages 59–67, 2007.

[29] Y. Lipman, D. Cohen-Or, D. Levin, and H. Tal-Ezer. Parameterization-free projection for geometry reconstruction. *ACM Trans. on Graphics*, 26(3):22, 2007.

[30] M. Lounsbery, T. D. DeRose, and J. Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Trans. on Graphics*, 16(1):34–73, 1997.

[31] C. Moenning, F. Mémoli, G. Sapiro, N. Dyn, and N. A. Dodgson. Meshless geometric subdivision. *Graphical Models*, 69(3-4):160–179, 2007.

[32] R. Pajarola. Efficient level-of-details for point based rendering. In *Proc. IASTED Computer Graphics and Imaging*, 2003.

[33] R. Pajarola, M. Sainz, and R. Lario. XSplat: External memory multiresolution point visualization. In *Proc. IASTED Visualization, Imaging and Image Processing*, pages 628–633, 2005.

[34] S.-B. Park, S.-U. Lee, and H. Choi. Multiscale surface representation and rendering for point clouds. In *Proc. Intl. Conf. in Image Processing*, volume 3, pages 1939–1942, 2004.

[35] M. Pauly. *Point Primitives for Interactive Modeling and Processing of 3D Geometry*. PhD thesis, Eidgenössische Technische Hochschule Zürich, 2003.

[36] M. Pauly and M. Gross. Spectral processing of point-sampled geometry. In *Proc. ACM SIGGRAPH*, pages 379–386, 2001.

[37] M. Pauly, M. Gross, and L. Kobbelt. Efficient simplification of point-sampled surfaces. In *IEEE Visualization*, pages 163–170, 2002.

[38] M. Pauly, L. Kobbelt, and M. Gross. Point-based multiscale surface representation. *ACM Trans. on Graphics*, 25(2):177–193, 2006.

[39] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *Proc. ACM SIGGRAPH*, pages 335–342, 2000.

[40] P. Reuter, P. Joyot, J. Trunzler, T. Boubekeur, and C. Schlick. Surface reconstruction with enriched reproducing kernel particle approximation. In *Proc. Symp. on Point-based Graphics*, pages 79–87, 2005.

[41] S. Rusinkiewicz and M. Levoy. QSplat: A multiresolution point rendering system for large meshes. In *Proc. ACM SIGGRAPH*, pages 343–352, 2000.

[42] J. M. Singh and P. J. Narayanan. Progressive decomposition of point clouds without local planes. In *Indian Conf. on Computer Vision, Graphics and Image Processing*, volume 4338 of *Lecture Notes in Computer Science*, pages 364–375, 2006.

[43] R. Wang, S. Zhang, and X. Ye. A novel simplification algorithm for point-sampled surfaces. In *Intl. Conf. on Multimedia and Ubiquitous Engineering*, pages 573–578, 2007.

[44] J. Wu, Z. Zhang, and L. Kobbelt. Progressive splatting. In *Proc. Symp. on Point-based Graphics*, pages 25–32, 2005.

[45] C. Xiao, Y. Miao, S. Liu, and Q. Peng. A dynamic balanced flow for filtering point-sampled geometry. *The Visual Computer*, 22(3):210–219, 2006.

[46] Z. Yang and T.-W. Kim. Moving parabolic approximation of point clouds. *Computer Aided Design*, 39(12):1091–1112, 2007.

[47] Z. Yu and H.-S. Wong. An efficient local clustering approach for simplification of 3D point-based computer graphics models. In *Intl. Conf. on Media & Expo*, pages 2065–2068, 2006.

[48] D.-H. Zhang, T. Yue, and Y.-H. Chen. Multi-scale surface representation of point-sampled geometry. In *Intl. Symp. on Image and Signal Processing and Analysis*, pages 371–376, 2005.

[49] D. Zorin, P. Schröder, and W. Sweldens. Interactive multiresolution mesh editing. In *Proc. ACM SIGGRAPH*, pages 259–268, 1997.