EUROGRAPHICS 2005 / M. Alexa and J. Marks
(Guest Editors)

Volume 24 (2005), Number 3

Interpolatory Refinement for Real-Time Processing of Point-Based Geometry

G. Guennebaud and L. Barthe and M. Paulin†

IRIT - CNRS - Université Paul Sabatier - Toulouse - France

Abstract

The point set is a flexible surface representation suitable for both geometry processing and real-time rendering. In most applications, the control of the point cloud density is crucial and being able to refine a set of points appears to be essential. In this paper, we present a new interpolatory refinement framework for point-based geometry. First we carefully select an appropriate one-ring neighborhood around the central interpolated point. Then new points are locally inserted where the density is too low using a $\sqrt{3}$-like refinement procedure and they are displaced on the corresponding curved Point Normal triangle. Thus, a smooth surface is reconstructed by combining the smoothing property produced by the rotational effect of $\sqrt{3}$-like refinements with the points/normal interpolation of PN triangles. In addition we show how to handle sharp features and how our algorithm naturally fills large holes in the geometry. Finally, we illustrate the robustness of our approach, its real-time capabilities and the smoothness of the reconstructed surface on a large set of input models, including irregular and sparse point clouds.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computational Geometry and Object Modeling]: Curve, surface, solid, and object representations I.3.3 [Computer Graphics]: Viewing algorithms

1. Introduction

Compared with meshes, point-based geometries are connectivity free and no topological consistency has to be satisfied through geometry manipulations. Hence, point sets become increasingly attractive as an alternative surface representation suitable for high quality rendering as well as for flexible processing of complex 3D models [KB04].

Although it is possible to deal with dense cloud of pure points, when performance matters, points are usually enriched by some attributes such as the surface normal and an estimation of the local sampling density. Such a point is commonly represented as an oriented disk and it is called a *splat*. The visualization step is generally performed by a surface splatting based technique [ZPvBG01]: all splats are projected onto the screen and filtered by a Gaussian kernel. Such techniques perform both fast rendering since they are implementable on GPUs [BK03, GP03, ZRB*04] and high quality rendering as long as the screen space size of splats remains small enough, i.e. a few pixels. Indeed, when the point set is not dense or uniform enough, splats radii are large, making the image blurry in the inner part of the object and producing artifacts on the silhouette. "Phong Splatting" like techniques [BK04] significantly improve the inner blur using second order informations. However these approaches consequently increase the memory consumption, reduce the initial simplicity of points and do not solve the geometric artifacts introduced by under-sampled models.

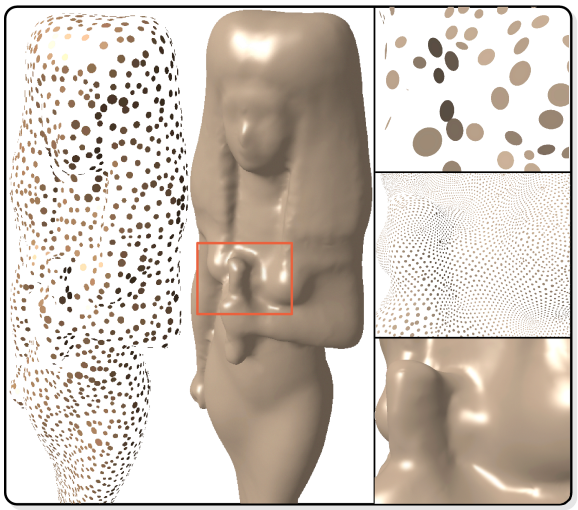

Figure 1: *Illustration of the smooth reconstruction capabilities of our refinement procedure on the Isis model irregularly sampled with 3500 points (left). The right images focus on a particularly under-sampled area, from top to bottom: the initial sampling, after four, then six refinement steps.*

Thus, the idea of a refinement algorithm maintaining the

† e-mail: {guenneba | lbarthe | paulin}@irit.fr

point set with uniform, dense sampling appears to be very relevant, and for this reason, this paper focus on point-based refinement techniques. Ideally, we would like such an algorithm to exhibit the following useful features:

- increase the sampling density,
- regularize scattered sampling,
- converge on a smooth surface,
- fill large holes in the geometry,
- handle boundary and sharp creases.

Related works: Fundamentally, the problem of point cloud refinement can be decompose in two steps: sampling (insertion operators) and displacement (smoothing operators).

In order to reduce or increase its density, the sampling is often controlled by a particle simulation procedure [Tur92, PGK02]. In [ABCO*03], Alexa et al. present an insertion procedure based on a local Voronoï diagram. Even though these two techniques can lead to a locally uniform sampling, their computational cost remains too expensive for real-time applications.

The smoothing issue is related to the problem of displacing the newly inserted points on a smooth surface defined by the original set of points. Most reconstruction techniques are based on implicit representation. For instance, radial basis functions (RBF) reconstruct a C^n implicit surface from a scattered set of point-normals [CBC*01]. Even though the global support of RBFs may be reduced by local approaches [OBA*03, TRS04], preprocessing and surface evaluation remain expensive. In [Lev03], Levin introduces a smooth point-based representation, called moving least-squares (MLS) surface, where the surface is implicitly defined by a local projection operator. Owing to the elegance of the projection idea and the relative locality of the involved computations, MLS surfaces have been used widely in applications ranging from surface editing [PKKG03] and ray-tracing [AA03] to up/down-sampling [PGK02, ABCO*03]. However, the projection operator requires a sufficiently dense input point set. Amenta and Kil [AK04] overcome this limitation by giving an explicit definition of the MLS surfaces which is able to handle a splat-based representation, but however the MLS surface is defined, the projection procedure remains expensive.

In [MMS*04], Moenning et al. present a meshless subdivision framework where mesh connectivity is replaced by intrinsic point proximity information. The subdivision operator is then based on geodesic weighted averages. However, authors simply introduce the concept and it is difficult to evaluate their method.

Targeting the real-time visualization of point clouds, Guennebaud et al. [GBP04] present a fast refinement algorithm of splat-based geometries. The input point set is iteratively refined by selecting a set of neighbors around each point and inserting new points on Bézier curves and Bézier patches. Owing to the extreme locality of these computations, their algorithm is fast enough to generate a million points per second. However, due to the lack of robustness of the refinement procedure, their algorithm is limited to uniform sampling (otherwise, the refined surface exhibits holes) and even though the surface looks globally smooth, it generates artifacts in the form of high frequency oscillations. This is essentially due to the non homogeneity of the smoothing rules (a mix of bivariate and univariate reconstruction techniques).

Beyond these important limitations, this last work demonstrates the efficiency of a pure point-based refinement algorithm for real-time graphics applications.

Our contribution: In this paper we present a new interpolatory subdivision framework which overcomes the weakness of the Guennebaud et al. refinement procedure [GBP04]. First, in order to reconstruct a **smooth surface** we propose to combine a local and fast smooth surface reconstruction method (based on curved Point-Normal triangles [VPBM01], also called PN triangles) with an iterative $\sqrt{3}$-like refinement scheme [Kob00]. The important features of PN triangles are their locality and their interpolation power since they interpolate both the points and their normals. For its part, the $\sqrt{3}$ scheme provides a strong global smoothing of the surface due to its rotation effect [Kob00] (figure 7d). Secondly, from the **robustness** point of view, we present two new accurate one-ring neighborhood computations and a new robust insertion procedure which optimizes the sampling uniformity and guarantees the absence of holes. Our new neighborhood computation procedures have the twin advantages that they are naturally symmetric while also being able to handle sparse point clouds through carefully analysis of the closest points. Finally, we show how to refine sharp features and we show that our approach is sufficiently fast and robust to be suitable for both large hole filling (figure 9) and real-time rendering.

2. Overview of our refinement procedure

Let $P^0 = \{\mathbf{p}_i\}$ be the initial point set defining a smooth manifold surface. We assume that we know for each point $\mathbf{p}_i \in P$, its normal $\mathbf{n}_i$ and the local density described by a scalar r_i which must be at least greater than the distance from $\mathbf{p}_i$ to the farthest neighbor of its natural first ring neighborhood.

In a similar fashion to subdivision surfaces, the point set is iteratively refined, leading to a sequence of point set $P^0, P^1, \ldots, P^l, \ldots$. Since our algorithm is interpolatory, we have $P^l \subset P^{l+1}$ (only the radius of points varies between two steps) and the refined point set P^{l+1} is the union of the set P^l itself and the set of points resulting from the local refinement of each point $\mathbf{p} \in P^l$.

The local refinement of a single point $\mathbf{p}$ requires several operations. First, a convenient one-ring neighborhood N_p of $\mathbf{p}$ is computed (section 4). Next, in order to match with the $\sqrt{3}$ scheme a set of triangles from which it is relevant to insert new points at their center of gravity is extracted from the implicit triangle fan formed by the sorted neighborhood

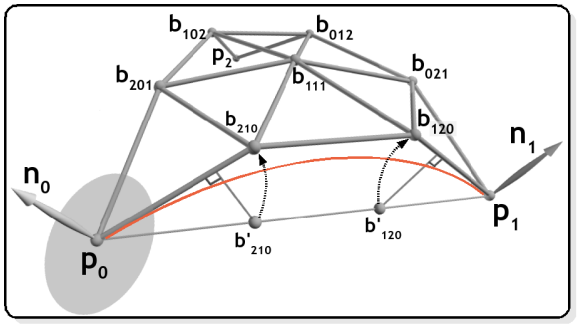

Figure 2: *Construction of the control polygon of a PN triangle interpolating three splats.*

of **p**. These triangles are selected in order to optimize the uniformity of the new neighborhood of **p** by taking into account the relative position of neighbors and the new points of $P^{l+1} - P^l$ which have been already inserted (section 5). Finally, in order to obtain a smooth surface, the centers of gravity of the selected triangles are displaced using our smoothing operator (section 3).

The paper is organized as follow: in the next section (section 3) we review the PN triangle technique from which we derive both our smoothing operator and a set of tools which are used in our neighborhood computations and during our local insertion procedure. Sections 4, 5 and 6 are respectively dedicated to the one-ring neighborhood selection, the local refinement of a single point and the refinement of sharp features. In section 7 we give some details on the data structures and the real-time rendering application. Finally, we present our results and discuss the continuity of the limit surface in section 8.

3. Point-Normal Interpolation Framework

In this section we present the set of tools used to extrapolate local geometric informations of the unknown surface S defined by the set of points with their normals. These tools are based on Point-Normal triangles [VPBM01]. A PN triangle is a Bézier triangle $B(u,v)$ of degree three interpolating three points of the point set and their normals.

$$B(u,v) = \sum_{i+j+k=3} \mathbf{b}_{ijk} \frac{3!}{i!j!k!} u^i v^j w^k, \; w = 1-u-v \qquad (1)$$

Given three points $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$ and their respective normals $\mathbf{n}_0$, $\mathbf{n}_1$, $\mathbf{n}_2$, the nine control points $\mathbf{b}_{ijk}$ of the patch (equation 1) are computed as follow (figure 2):

1. The three extremities $\mathbf{b}_{300}$, $\mathbf{b}_{030}$, $\mathbf{b}_{003}$ are respectively $\mathbf{p}_0$, $\mathbf{p}_1$, $\mathbf{p}_2$.
2. The positions of the six boundary control points ($\mathbf{b}_{ijk}$, $i+j+k=3$, $i \neq j \neq k$) only depend on the two extremities of their respective boundary and they are all computed in the same manner. For instance, the control point $\mathbf{b}_{210}$ is the projection of $\mathbf{b}'_{210} = \mathbf{p}_0 + \frac{1}{3}\mathbf{p}_0\mathbf{p}_1$ onto the tangent plane of $\mathbf{p}_0$, moved such that the length of the vector $\mathbf{p}_0\mathbf{b}_{210}$ is equal to the third of the distance between the two extremities $\mathbf{p}_0$ and $\mathbf{p}_1$. Let $Q_i(\mathbf{x})$ be the orthogonal projection operator, projecting the point $\mathbf{x}$ onto the tangent plane of $\mathbf{p}_i$, then:

$$Q_i(\mathbf{x}) = \mathbf{x} + (\mathbf{p}_i - \mathbf{x}) \cdot \mathbf{n}_i * \mathbf{n}_i \qquad (2)$$

$$\mathbf{b}_{210} = p_0 + \frac{\|\mathbf{p}_0\mathbf{p}_1\|}{3} * \frac{Q_0(\mathbf{b}'_{210}) - \mathbf{p}_0}{\|Q_0(\mathbf{b}'_{210}) - \mathbf{p}_0\|}$$

3. The central point $\mathbf{b}_{111}$ is set to reproduce quadratic polynomials by taking $b_{111} = \mathbf{c} + \frac{3}{2}(\mathbf{e} - \mathbf{c})$ where $\mathbf{c}$ is the center of gravity of the three input points and $\mathbf{e}$ is the average of the six boundary control points.

Our construction varies from the one of Vlachos et al. [VPBM01] only in one point. After projection, we displace the boundary points $\mathbf{b}_{ijk}$ ($i+j+k=3$, $i \neq j \neq k$) while they do not. This is done to avoid the introduction of flatness in the reconstructed surface, especially in areas of high curvature.

Smoothing Operator

We define the smoothing operator ϕ as the displacement of the center of gravity (*cog*) onto the PN triangle. Thus, the position of the new point $\mathbf{p}_{new}$ is:

$$\mathbf{p}_{new} = cog(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2) + \phi(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2)$$

where ϕ is the average of the six tangent vectors t_i:

$$\phi(\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2) = \frac{1}{6}\sum_{i=0}^{5} \mathbf{t}_i \qquad (3)$$

with:

$$t_0 = \mathbf{b}_{210} - \mathbf{p}_0 \,,\; t_1 = \mathbf{b}_{120} - \mathbf{p}_1 \,,\; t_2 = \mathbf{b}_{021} - \mathbf{p}_1 \,,\; \ldots$$

The normal of the new point is the cross product of the two tangent vectors at the center of the PN triangle ($u = v = \frac{1}{3}$):

$$B^u(\tfrac{1}{3},\tfrac{1}{3}) = 7(\mathbf{p}_1 - \mathbf{p}_0) + \mathbf{b}_{120} - \mathbf{b}_{102} + \mathbf{b}_{012} - \mathbf{b}_{210} + 2(\mathbf{b}_{021} - \mathbf{b}_{201})$$
$$B^v(\tfrac{1}{3},\tfrac{1}{3}) = 7(\mathbf{p}_2 - \mathbf{p}_0) + \mathbf{b}_{102} - \mathbf{b}_{120} + \mathbf{b}_{021} - \mathbf{b}_{201} + 2(\mathbf{b}_{012} - \mathbf{b}_{210})$$

Geodesic Distance

With respect to the Euclidean distance, the geodesic distance is useful for evaluating the relative position of points on a surface. Following our local point-normal surface reconstruction we define the local geodesic distance $\widetilde{G}(\mathbf{p}_0, \mathbf{p}_1)$ between two relatively close points $\mathbf{p}_0$ and $\mathbf{p}_1$ as the length of a cubic Bézier curve interpolating the two points and their normals. In our construction, this curve is the boundary of a PN triangle (figure 2). However the exact computation of the length of a Bézier curve is too expensive for our purpose. We rather use a sufficient approximation given by the length of the control polygon:

$$\widetilde{G}(\mathbf{p}_0, \mathbf{p}_1) = \frac{2}{3}\|\mathbf{p}_0\mathbf{p}_1\| + \|b_{210} - b_{120}\| \qquad (4)$$

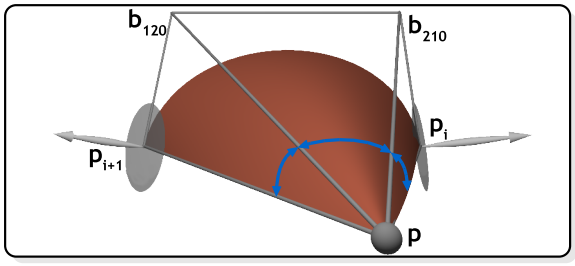

Figure 3: *Our "curved angle" between two point-normals* $\mathbf{p}_i$, $\mathbf{p}_{i+1}$ *relatively two a third point* $\mathbf{p}$ *is specially usefull for areas of high curvature. On this example there is a ratio of two between the geometric angle and our "curved angle".*

"Curved-Angle"

Measuring the angle between two points $\mathbf{p}_0$, $\mathbf{p}_1$ relatively to a third point $\mathbf{p}$ is especially useful when analyzing the neighborhood of $\mathbf{p}$. However when points are bound to a surface, the geometric angle may be not significant enough (figure 3). Thus, following our previous geodesic distance approximation, we define the "curved-angle" $\widetilde{A}_{\mathbf{p}}(\mathbf{p}_0,\mathbf{p}_1)$ as the sum of three angles taken along the control polygon of the boundary curve interpolating $\mathbf{p}_0$, $\mathbf{p}_1$:

$$\widetilde{A}_{\mathbf{p}}(\mathbf{p}_0,\mathbf{p}_1) = \widehat{\mathbf{p}_0\mathbf{p}b_{210}} + \widehat{b_{210}\mathbf{p}b_{120}} + \widehat{b_{120}\mathbf{p}\mathbf{p}_1} \qquad (5)$$

Bounds on Point-Normal Interpolation

The construction by projection of the control points of a PN triangle boundary is not always consistent. Indeed, as illustrated in figure 4a, certain configurations of the positions and normals of the two boundary extremities yield a surface reconstruction which is inconsistent with respect to the normal's orientation (inside/outside). This situation occurs when the point $\mathbf{p}_1$ is inside the infinite cone of apex $\mathbf{p}_0$ and axis $\mathbf{n}_0+\mathbf{n}_1$ (figure 4b). In this case, a specific (global) treatment could be applied in order to re-establish the normal consistency. However this would mean that we try to reconstruct a highly under-sampled surface from a r-sampling P^0 with $r>2$ [ABK98] and hence, it is more natural to consider that the points $\mathbf{p}_0$ and $\mathbf{p}_1$ are not neighbors. Thus two points are not neighbors if the following condition is not satisfied:

$$\left|(\mathbf{n}_0+\mathbf{n}_1)\cdot\frac{\mathbf{p}_0\mathbf{p}_1}{\|\mathbf{p}_0\mathbf{p}_1\|}\right| > 1+\mathbf{n}_0\cdot\mathbf{n}_1 \qquad (6)$$

4. One-ring Neighborhoods

The selection of a pertinent one-ring neighborhood is a critical step of the algorithm. Indeed, it is from this set of points that new points will be inserted around the refined point $\mathbf{p}$, and hence, the robustness of the refinement process as well as its capacity to fill large holes directly depend on the quality of this neighbor selection. A simple neighborhood definition such as the common k-nearest neighbors leads to a very poor selection and the development of a more accurate method is essential. Advanced techniques use a Voronoï diagram [UMA04] or angle criterion [LP02, GBP04] after an orthogonal projection of the nearest neighbors in the local tangent plane. At the same time as it simplifies the neighbor selection, the orthogonal projection to a 2D domain significantly reduces accuracy. Indeed the elevation information which is crucial for dealing with low local density and high curvature areas are lost.

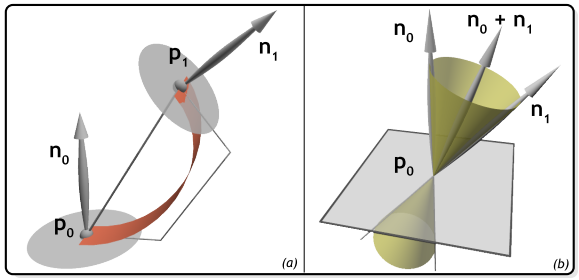

Figure 4: *(a) The relative positions and orientations of the points* $\mathbf{p}_0$ *and* $\mathbf{p}_1$ *are such that the construction by projection is inconsistent. (b) Given the position* $\mathbf{p}_0$ *and the two normals* $\mathbf{n}_0$, $\mathbf{n}_1$*: the point* $\mathbf{p}_1$ *must be outside the yellow cone.*

For these reasons we propose two new neighborhood computation procedures significantly improving the tolerance to under-sampled and/or scattered point sets. The first has the advantage of being naturally symmetric (if $\mathbf{p}_i$ is a neighbor of $\mathbf{p}_j$ then $\mathbf{p}_j$ is also a neighbor of $\mathbf{p}_i$) while the second improves the selection in under-sampled areas in spite of being slightly more expensive and losing the symmetry property.

4.1. Fuzzy BSP Neighborhood

The definition of our first neighborhood N_p^f is based on a BSP neighborhood [Pau03]. However, our selection is performed without projection (allowing the neighborhood relation to be symmetric) and we replace the too selective discriminant planes by more flexible *fuzzy* planes (better suited for scattered sampling). We start by computing the Euclidean neighborhood N_p^ε of $\mathbf{p}$ as the indices of all points

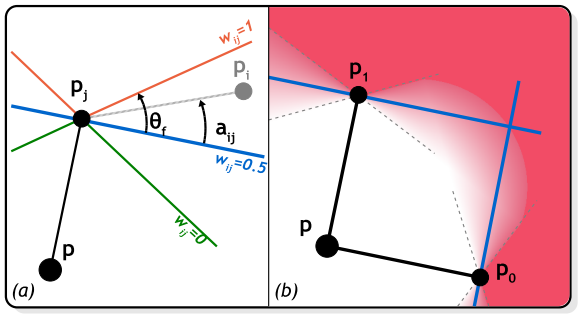

Figure 5: *Illustration of Fuzzy planes. (a) Computation of the badness value* w_{ij} *between two points. (b) The red graduation represents the variation of the badness value* w_i *from* 0 *(white) to* ≥ 1 *(red), when* w_{i1} *(resp.* w_{i0}*) is the max of the successor's (resp. predecessor's) badness.*

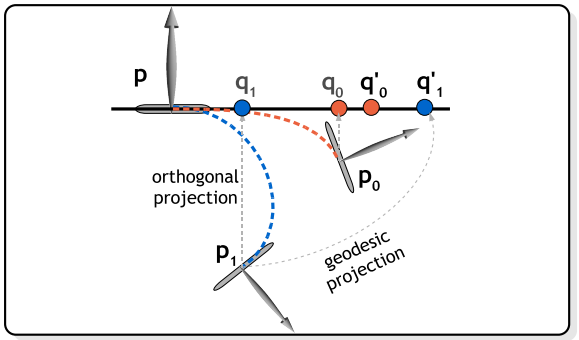

Figure 6: *Illustration of our "geodesic projection". For instance, the "geodesic projection" $\mathbf{q}'_1$ of $\mathbf{p}_1$ onto the tangent plane of $\mathbf{p}$ is the orthogonal projection $\mathbf{q}_1$ of $\mathbf{p}_1$ moved such that its distance to $\mathbf{p}$ is the geodesic distance between $\mathbf{p}$ and $\mathbf{p}_1$.*

$\mathbf{p}_i$ included in the ball of center $\mathbf{p}$ and radius r:

$$N_p^\varepsilon = \{i \,|\, \mathbf{p}_i \in P^l, \mathbf{p}_i \neq \mathbf{p}, \|\mathbf{p}-\mathbf{p}_i\| < r\} \tag{7}$$

Following the previous section we first remove from N_p^ε all neighbors $\mathbf{p}_i$, $i \in N_p^\varepsilon$ which do not satisfy the condition 6.

Next, in order to get a pertinent one-ring we remove all neighbors which are strongly "behind" another one or slightly "behind" two others. For this purpose, we introduce a *badness* value w_{ij} stating to what extent the neighbor $\mathbf{p}_i$ is "behind" the neighbor $\mathbf{p}_j$. This value depends on the signed angle between the vector $\mathbf{p}_j\mathbf{p}_i$ and the plane of normal $\mathbf{pp}_j$ passing through p_j (figure 5). Let α_{ij} be this angle and θ_f a given tolerance angle (typically $\frac{\pi}{6}$). Then:

$$w_{ij} = \begin{cases} 0 & \text{if } \alpha_{ij} < -\theta_f \\ \frac{1}{2}\left(\frac{\sin(\alpha_{ij})}{\sin(\theta_f)} + 1\right) & \text{otherwise} \end{cases}$$

with $\sin(\alpha_{ij}) = \frac{\mathbf{pp}_j}{\|\mathbf{pp}_j\|} \cdot \frac{\mathbf{p}_j\mathbf{p}_i}{\|\mathbf{p}_j\mathbf{p}_i\|}$. Let $Succ_i$ (resp. $Pred_i$) be the set of successors (resp. predecessors) of the point $\mathbf{p}_i$, $i \in N_p^\varepsilon$ such that $Succ_i = \{j \in N_p^\varepsilon \,|\, 0 < \widehat{\mathbf{q}_i\mathbf{p}\mathbf{q}_j} < \pi\}$ where $\mathbf{q}_i$ and $\mathbf{q}_j$ are the orthogonal projections of the points $\mathbf{p}_i$ and $\mathbf{p}_j$ on the tangent plane of $\mathbf{p}$ (resp. $Pred_i = \{j \in N_p^\varepsilon \,|\, -\pi < \widehat{\mathbf{q}_i\mathbf{p}\mathbf{q}_j} < 0\}$). We point out that the projections are only used to sort the points as successors and predecessors. Next, we compute a *badness* value w_i for each neighbor $\mathbf{p}_i$ as the sum of the two maximal values w_{ij} of the successors and w_{ik} of the predecessor:

$$w_i = \max_{j \in Succ_i}(w_{ij}) + \max_{j \in Pred_i}(w_{ij}) \tag{8}$$

As soon as a neighbor $\mathbf{p}_i$ has a *badness* value w_i greater than 1 it is removed from the neighborhood of $\mathbf{p}$ (figure 5b) yielding the final one-ring neighborhood N_p^f. Finally, the neighbors $\mathbf{p}_i$, $i \in N_p^f$ are sorted by increasing angles of their projection $\mathbf{q}_i$ onto the tangent plane of $\mathbf{p}$, so that this neighborhood implicitly forms a triangle fan around $\mathbf{p}$.

4.2. Accurate Neighborhood

Our second neighborhood definition N_p^g is a variant of the previous one where the computation of the weights w_{ij} is performed after the "geodesic projection" of all neighbors $\mathbf{p}_i$, $i \in N_p^\varepsilon$ onto the tangent plane of $\mathbf{p}$. This projection technique differs significantly from the standard orthogonal projection since the geodesic distance between $\mathbf{p}$ and its neighbors $\mathbf{p}_i$ (equation 4) is also the distance between $\mathbf{p}$ and the projection $\mathbf{q}'_i$ of $\mathbf{p}_i$ (the figure 6). Thus $\mathbf{q}'_i$ is computed as follow:

$$\mathbf{q}'_i = \mathbf{p} + \widetilde{G}(\mathbf{p}, \mathbf{p}_i)\frac{\mathbf{q}_i - \mathbf{p}}{\|\mathbf{q}_i - \mathbf{p}\|} \tag{9}$$

where, $\mathbf{q}_i$ is the orthogonal projection of the point $\mathbf{p}_i$ on the tangent plane of $\mathbf{p}$. From here, the neighborhood N_p^g is selected as N_p^f except that the weights w_{ij} are computed with the projections $\mathbf{q}'_i$, $\mathbf{q}'_j$ instead of the initial positions $\mathbf{p}_i$, $\mathbf{p}_j$. The usefulness of this projection is illustrated figure 6: if we apply directly our fuzzy plane filtering without projection, the two points $\mathbf{p}_0$ and $\mathbf{p}_1$ will be selected. An orthogonal projection is even less precise (because only $\mathbf{p}_1$ would be selected) while after our "geodesic projection" it clearly appears that $\mathbf{p}_1$ is not a neighbor of $\mathbf{p}$.

5. The Local Refinement Algorithm

In this section, we detail the local refinement of the current point $\mathbf{p}$ from its neighborhood N_p computed with one of the previous methods (sections 4.1 and 4.2). The choice of the method depends on the sampling quality and we comment on this in next sections. The challenge is now to build a relevant new neighborhood N'_p around $\mathbf{p}$. This new neighborhood corresponds to one refinement step around $\mathbf{p}$ and it must both fill holes and regularize the sampling.

We first initialize the set N'_p with the indices of the points of P^{l+1} which can be considered as newly inserted points i.e. the points which are at a distance from $\mathbf{p}$ smaller than $\lambda_1 r$ with $\lambda_1 = 1/\sqrt{3}$ (figures 7a and 7b). The value of λ_1 is taken according to the scale factor of a $\sqrt{3}$ refinement in the regular case [Kob00].

We consider that the refinement of $\mathbf{p}$ is complete as soon as the maximal "curved angle" (equation 5) between two consecutive points of N'_p is smaller than a given threshold $\theta_c = \frac{\pi}{2}$. Hence if the initialization does not provide a complete neighborhood, new points must be inserted. To do so, we define three terms (figure 7b and 7c):

- Y_p is the set of points already inserted which are sufficiently close to $\mathbf{p}$ but not close enough to be selected in N'_p:

$$Y_p = \{h \,|\, \mathbf{p}_h \in P^{l+1} - P^l, \lambda_1 r < \|\mathbf{p}_h - \mathbf{p}\| < r\}, \tag{10}$$

- $\mathcal{D}_p$ is the discard space avoiding oversampling and redundancy. It is the union of the spheres of radius $\frac{r}{2}$ centered on the points of Y_p:

$$\mathcal{D}_p = \{x; h \in Y_p, \|x - \mathbf{p}_h\| < \lambda_2 r_h\} \tag{11}$$

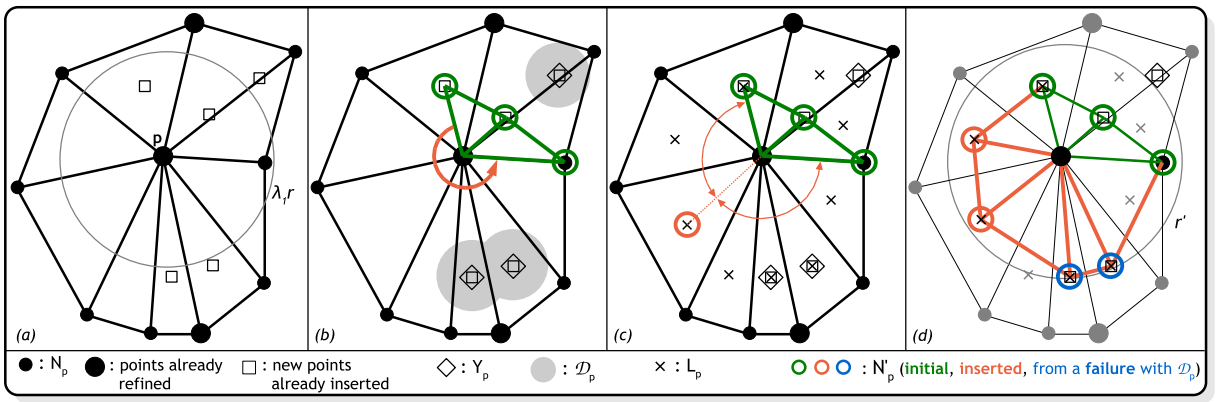

Figure 7: *Local refinement of the current point* $\mathbf{p}$. *(a) The neighborhood of the point* $\mathbf{p}$ *before its own refinement. It is composed of its neighbors* N_p *and the closest points already inserted during the previous refinement of two of its neighbors. (b) Initialization of the new neighborhood* N'_p *(in green) and illustration of the discard space* $\mathcal{D}_p$ *around the points of* Y_p. *The arrow shows the area where the insertion of new points will begin in order to complete* N'_p. *(c) New points of* L_p *are computed and the best one filling the previous area is selected and inserted. (d) After the insertion of two new points (in red) and the failure of two other insertions due to the discard space (in blue) the new neighborhood* N'_p *of* $\mathbf{p}$ *is complete. Four new points remain in* L_p*; these are ignored because they are superfluous. Note the rotation effect between* N_p *and* N'_p *of our* $\sqrt{3}$*-like refinement.*

- L_p is the set of all possible new points, i.e. it is the set of points resulting from the application of our smoothing operator (equation 3) on the center of gravity of all triangles of the implicit triangle fan formed by the sorted neighborhood N_p:

$$L_p = \{cog(\mathbf{p}, \mathbf{p}_i, \mathbf{p}_{i+1}) + \phi(\mathbf{p}, \mathbf{p}_i, \mathbf{p}_{i+1}) \,|\, i \in N_p\}, \quad (12)$$

The insertion procedure is as following:

While the neighborhood N'_p is not complete **repeat**

1. Select the pair of consecutive points $\mathbf{p}_j$, $\mathbf{p}_{j+1}$ in N'_p having the maximal "curved angle" (figure 7b).
2. Select the new point in L_p which best balances point sampling (figure 7c). A good candidate is the point $\mathbf{p}_k \in L_p$ such that the minimum of the two angles $\widehat{\mathbf{p}_j\mathbf{p}\mathbf{p}_k}$ and $\widehat{\mathbf{p}_k\mathbf{p}\mathbf{p}_{j+1}}$, is maximal.
3. If this point is not in the discard space $\mathcal{D}_p$, it is inserted in N'_p and P^{l+1}, otherwise no new point is inserted in P^{l+1} and the point $\mathbf{p}_k$ is replaced in N'_p by the closest point in Y_p (figure 7d). Hence, if the samples are locally dense enough, no new point is inserted.

When this process terminates, the radius of the point $\mathbf{p}$ is updated according to its new neighborhood N'_p. The new radius r' is set to the maximum distance between $\mathbf{p}$ and the points of N'_p: $r' = \max_{j \in N'_p}(\|\mathbf{p} - \mathbf{p}_j\|)$ (figure 7d). The radius r_j of each new neighbor $\mathbf{p}_j$, $j \in N'_p$ is set to the maximum of the four values: r_j, $\|p_j - p\|$, $\|p_j - p_{j-1}\|$ and $\|p_j - p_{j+1}\|$.

6. Sharp features

A common and efficient way to handle sharp creases with point-based geometry is to use clipped splats [ZRB*04]. Although clipped splats are sufficient for the rendering, geometry processing requires, in addition, that splats share the same center [PKKG03, GBP04]. Thus our crease splat is just a single point with two different normals.

With our $\sqrt{3}$-like up-sampling strategy the refinement of boundaries and creases is more fussy than with a diadic refinement as used in [GBP04] because new points are never explicitly inserted between two points. With the mesh-based $\sqrt{3}$ subdivision scheme [Kob00] Kobbelt proposes the insertion of two vertices on each boundary and crease segment at each odd refinement step only. However, with point-based geometry, there is no connectivity and after two refinement steps, the crease (or boundary) points will probably not be neighbors anymore so that no special treatment can be applied between them. Hence we explicitly store a list of crease and boundary segments; this solution has the advantage of being robust and functional. This list is computed during the first refinement step. Crease segments between two crease splats are detected in the same manner as in [GBP04]. For the boundaries, a point $\mathbf{p} \in P^0$ is said to be a potential boundary splat if there exist two consecutive points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$, $i \in N_p$ such that the angle $\mathbf{p}_i\mathbf{p}\mathbf{p}_{i+1}$ is greater than a given threshold $\theta_b \in \left]\frac{2\pi}{3}, \pi\right]$. Then if $\mathbf{p}$ is a potential boundary point and $\mathbf{p}_i$, $i \in N_p$ is also a potential boundary point, the pair $(\mathbf{p}, \mathbf{p}_i)$ is inserted in the list of boundary segments. Note that the choice of θ_b allow us to choose how strongly to smooth the boundary: if $\mathbf{p}_i$, $\mathbf{p}$, $\mathbf{p}_{i+1}$ have an angle smaller than θ_b but they are effectively on a boundary then the inserted point between them will have a larger angle, and hence the boundary will be effectively detected after a few refinement steps. The interpolation between two crease/boundary points uses a cubic Bézier curve (as in [GBP04]) except that two points are inserted at the thirds of the curve at each odd refinement step instead of inserting a single points in its middle at every step.

7. Data structures and Implementation

Closest points query

As in a lot of point-based processing methods, a critical time-consuming part of our method is the closest points query necessary to compute the Euclidean neighborhood N_p^ε (equation 7). To improve efficiency, points must be spatially sorted into a data structure, like a *kd*-tree or a 3D grid, with a fine granularity.

Moreover, the local refinement step of a single point $\mathbf{p} \in P^l$ (section 5) also requires us to find the closest points already inserted into P^{l+1} (to compute the sets N'_p and Y_p). Assuming that new points are sequentially inserted into a list of points, our solution is to associate to each point $\mathbf{p} \in P^l$ the indices of the first and last new points inserted during its own refinement. We call these points the *children* of $\mathbf{p}$. Thus, the set of new points already inserted into P^{l+1} close to $\mathbf{p}$ is inferred from all children of all neighbors $\mathbf{p}_i$, $i \in N_p^\varepsilon$ of $\mathbf{p}$. This solution has the advantage that it naturally creates a hierarchy of bounding spheres (a radius is associated with each point) which is also used to perform efficient closest points queries with a very low memory consumption: we only store two indices per point. We only have to structure the initial point set P^0 once: no additional data structure has to be built for other levels and no insertion has to be performed, and hence a static data structure (e.g. a *kd*-tree) can be efficiently used. A closest points query around the current point $\mathbf{p}$ at a level l with $l \neq 0$ is done by performing a recursive traversal of the bounding spheres hierarchy while the starting bounding spheres (points of the set P^0) are found by performing a closest points query using the initial data structure.

Real-Time Refinement

On one hand, the data structure presented above is especially useful for real-time applications, i.e. when some parts of the model must be dynamically refined. For instance, in a real-time rendering system, such as the one presented in [GBP04], the refinement procedure is used to maintain a screen space splat size smaller than a given threshold (e.g. two pixels). According to the relative position of the camera, under-sampled regions of the models are refined (yielding to the insertion of new points stored in a cache) while outdated generated points are removed in order to free the memory cache. In this context, our neighbor search algorithm is particularly well suited because insertion and deletion of points is trivial, the memory cost is very low and levels are well separated. The clear separation of levels is essential when the model is not globally refined since different parts of the model are refined at different levels while new points of the level $l+1$ must be interpolated from points of the set P^l only.

On the other hand, still in the context of a rendering application, the selection of points that have to be refined is equivalent to a visibility and level-of-details (LOD) point selection which is generally performed by spatially sorting points into a hierarchy of bounding volumes (e.g. *kd*-tree or octree). Our hierarchy of bounding spheres could also be used for this purpose (in a way it looks like the *QSplat* representation [RL00]), however, it has been shown that a too fine data structure is inefficient for high-level point selection [DVS03, EF04]; such a data structure should contain approximately one or two thousands points per cell. Moreover, no regular hierarchical data structure matches the $\sqrt{3}$ refinement. Thus, in our real-time rendering system, we have opted for a more flexible hierarchy of bounding boxes (similar to the *point-octree* [Sam89]) where the goal is to keep a constant number of points per node. We start from a set of axis aligned bounding boxes (the root nodes). Then, when a node is refined, according to its actual number of points, it is:

- not split (the node has only a single child),
- split in two or three along its maximal dimension,
- split in four along its two maximal dimensions.

With respect to our fine hierarchy of points, to memory consumption and to efficient GPU rendering requirements, the points of a node must be stored sequentially in a single chunk of video memory (shared by all nodes). Thus, a node has just to store an axis aligned box, its points as a range of indices (the indices of its first and last points) and from zero to four children. In order to respect the sequential storage of points per nodes, a node is refined as follow:

- split the current node into 1, 2, 3 or 4 children (see above),
- for each child, refine all points which are in its bounding box.

At the end of this process the bounding box of each child must be updated, i.e. contracted and/or extended in each direction in order to be as small as possible and to guarantee that it effectively contains all its points (the new points inserted during the refinement of one point of the child may be outside its initial box).

In order to maintain a real-time frame rate (above 24 fps) the time allowed for the refinement procedure must be bound (implying a breadth-first order traversal). Owing to the time consumed by the point selection and the rendering itself, the remaining time per frame for the refinement procedure is very limited, and we have measured with our implementation that a point generation rate above 300k points per second is the minimal performance required for a comfortable navigation.

Simplifications/Optimizations

The technique presented above has been designed to handle under-sampled and scaterred point clouds. However, for relatively well sampled models and/or after a few refinement steps a lot of expensive tests can be safely optimized:

1. Approximate the geodesic distance by the Euclidean distance: $\widetilde{G}(p_0, p_1) \approx \|p_1 - p_0\|$
2. Approximate the "curved-angle" by the simple geometric angle: $\widetilde{A}_p(p_0, p_1) = \widehat{p_0 p p_1}$
3. Use the N_p^f instead of the N_p^g neighborhood.
4. Approximate the position of a new point by the center of gravity during the refinement process and apply the

smoothing operator if and only if the new point is effectively inserted.

These optimizations allow us to significantly improve the performance of the algorithm (by a speedup factor from 1.5 to 2).

8. Results and Discussion

We have tested our new approach on a wide variety of point-based models. The refinement of textured models is illustrated in figure 12: the color of a new point is linearly interpolated from the three extremities of the PN triangles. Figure 10 illustrates the refinement of sharp features.

Robustness evaluation: In order to evaluate the robustness of our method, it has been tested on several irregularly downsampled models: for instance in figure 1 our refinement algorithm is applied to 3500 points randomly selected from a set of 150k points representing a statue of Isis. Figure 9 illustrates the use of our refinement method on an especially large hole. To fill this hole we have simply adjusted the radius of points such that they overlap the hole and we have applied our refinement algorithm several times. Figure 11 illustrates the usefulness of our "geodesic projection" and our "curved angle" on an highly under-sampled area. In this example, the boundary of the David's eye exhibits holes if these tools are not used.

Performances: We have tested our implementation on an Athlon 3500+ system with an nNidia GeForce 6800 graphics card. Our algorithm is able to generate from 350k to 450k points per second depending of the local regularity of the sampling. The optimizations presented in the previous section allow us to reach a point generation rate around 700k points per second. The time consumption of each part of the refinement procedure is (approximately) as follow: 23% for the Euclidean neighborhood, 40% for the filtering of the neighborhood, 29% for the selection of new points and 8% for the interpolation and radii updates.

Comparisons: Figure 8 illustrates the superiority in the reconstructed surface smoothness of our new $\sqrt{3}$ refinement algorithm over the butterfly mesh based interpolatory subdivion scheme [DLG90, ZSS96] (C^1 surface but with large oscillations) and the Guennebaud et al. diadic refinement method [GBP04] (high-frequency oscillation artifacts).

8.1. Convergence and Continuity issues

From a practical point of view, our results show that our new refinement algorithm provides a high quality smooth surface and allows flexibility in the quality of the input point sampling. However, from the theoretical point of view, all the analysis mechanism developed during the last decade for meshes does not hold for point-based geometry. Hence, questions remain: what are the convergence and the continuity ? The heuristic character of our refinement procedure and its dependence on the point processing order make it complicated to undertake rigorous analysis of the limit surface. Nevertheless, we can expect some good properties.

For convergence, we show that the radius of any points $p \in P^l$, noted r^l, has for limit zero when l tends to infinity. Indeed an upper bound of the radius r' is given by the distance between $\mathbf{p}$ and its farthest new neighbor $\mathbf{p}_k$ which is the center of a PN triangle formed by the neighbors of $\mathbf{p}$ (at a maximal distance r from $\mathbf{p}$):

$$r^{l+1} = \|p - p_k\| = \|p - \tfrac{1}{6}(\mathbf{b}_{210} + \mathbf{b}_{120} + \mathbf{b}_{021} + \mathbf{b}_{012} + \mathbf{b}_{102} + \mathbf{b}_{201})\|$$

Since:

$$\|p - \tfrac{1}{2}(\mathbf{b}_{210} + \mathbf{b}_{120})\| \leq \tfrac{5}{6} r^l \;,\quad \|p - \tfrac{1}{2}(\mathbf{b}_{201} + \mathbf{b}_{102})\| \leq \tfrac{5}{6} r^l$$
$$\|p - \tfrac{1}{2}(\mathbf{b}_{021} + \mathbf{b}_{012})\| \leq (\cos(\beta/2) + \tfrac{2}{3}\sin(\beta/2)) r^l$$

we have:

$$r^{l+1} \leq c_\beta r^l \tag{13}$$

$$c_\beta = \tfrac{1}{3}(\tfrac{5}{6} + \tfrac{5}{6} + (\cos(\beta/2) + \tfrac{2}{3}\sin(\beta/2))) \tag{14}$$

where β is the angle between the farthest neighbor of p and its successor (or predecessor). Whatever the value of β is, $c_\beta < 0.96$ that is strictly less that 1 and hence, the limit of r^l is zero.

For the continuity issue, considering a given point $\mathbf{p} \in P^l$, we define α_p^l as the angle of a double cone of apex $\mathbf{p}$ and axis $\mathbf{n}$ (the normal of $\mathbf{p}$) such that the complement of this double cone (co-cone) contains all the neighbors of $\mathbf{p}$ at the level l. Moreover, the symmetry property of our first neighborhood computation guarantee that the point $\mathbf{p}$ is one of the extremities of each built PN triangle which has yield to the insertion of a new points in its new neighborhood. Hence, assuming that after a finite number of refinement steps, no Bézier patch presents an inflexion point, we show that the angle α_p^l has for limit 0 when l tends to infinity. As above, we find a constant $c < 1$ such that $\alpha^{l+1} < c\,\alpha^l$. This means that, at the limit, all neighbors of $\mathbf{p}$ are in its tangent plane that is a necessary condition for the G^1 continuity. Our assumption has always been satisfied in our experimentations, even in very distorted areas, however there is no theoretical guaranty since it has not been proved.

9. Conclusions

We have presented a fast and robust interpolatory refinement method suitable for real-time processing of point based geometry. The refinement procedure is based on a PN triangle interpolation associated with a $\sqrt{3}$-like refinement. Our results show the efficiency of this combination which generates smooth reconstructed surfaces. We have presented a set of tools allowing the extrapolation of local geometric information from a point-normal surface representation. These tools have been particularly useful in the development of neighborhood computations handling both undersampled and scattered point clouds. Our technique is also robust enough to smoothly fill large holes in the geometry. The connectivity free of point cloud has allowed us to develop an efficient adaptive refinement strategy with trivial reverse refinement steps.

Even though the surface looks very smooth, there is no fundamental theoretical machineries allowing a rigorous analysis of the limit surface (as for subdivision surfaces

on meshes). Nevertheless, we have proved some important properties of our refinement: refinement steps after refinement steps the inserted points converge on the central old point and at the limit, the inserted points lie in the tangent plane of the old central point. Hence, as future works, we intend to follow the analysis of the limit surface. We will also investigate the capabilities of our refinement procedure in interactive applications such as multi-resolution modeling (where the user will simply interact with oriented disks) and for the efficient compression of point sets.

Acknowledgements

We would like to thank Neil Dodgson from the University of Cambridge for proof-reading the paper.

References

[AA03] ADAMSON A., ALEXA M.: Approximating and intersecting surfaces from points. In *Proceedings of the Eurographics Symposium on Geometry Processing* (2003), pp. 245–254. 2

[ABCO*03] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Computing and rendering point set surface. *IEEE Transaction on Visualization and Computer Graphics 9*, 1 (2003), 3–15. 2

[ABK98] AMENTA N., BERN M., KAMVYSSELIS M.: A new voronoï-based surface reconstruction algorithm. In *Proceedings of ACM SIGGRAPH 98* (1998). 4

[AK04] AMENTA N., KIL Y. J.: Defining point set surfaces. In *Proceedings of ACM SIGGRAPH 2004, Computer Graphics Proceedings* (2004). 2

[BK03] BOTSCH M., KOBBELT L.: High-Quality Point-Based Rendering on Modern GPUs. In *11th Pacific Conference on Computer Graphics and Applications* (2003), pp. 335–343. 1

[BK04] BOTSCH M., KOBBELT L.: Phong splatting. In *Proceedings of Point-Based Graphics 2004* (2004). 1

[CBC*01] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: Reconstruction and representation of 3D objects with radial basis functions. In *Proceedings of ACM SIGGRAPH 2001* (2001), pp. 67–76. 2

[DLG90] DYN N., LEVIN D., GREGORY J.: A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transaction on Graphics, 9 (2)* (1990), 160–169. 8

[DVS03] DACHSBACHER C., VOGELGSANG C., STAMMINGER M.: Sequential point trees. In *Proceedings of ACM SIGGRAPH 2003* (2003), pp. 657–662. 7

[EF04] ENRICO G., FABIO M.: Layered point clouds: a simple and efficient multiresolution structure for distributing and rendering gigantic point-sampled models. *Computers & Graphics 28* (2004), 815–826. 7

[GBP04] GUENNEBAUD G., BARTHE L., PAULIN M.: Dynamic surfel set refinement for high-quality rendering. *Computers & Graphics 28* (2004), 827–838. 2, 4, 6, 7, 8

[GP03] GUENNEBAUD G., PAULIN M.: Efficient screen space approach for Hardware Accelerated Surfel Rendering. In *Proceedings of Vision, Modeling and Visualization* (2003), IEEE Signal Processing Society, pp. 41–49. 1

[KB04] KOBBELT L., BOTSCH M.: A survey of point-based techniques in computer graphics. *Computers & Graphics 28* (2004), 801–814. 1

[Kob00] KOBBELT L.: $\sqrt{3}$ subdivision. In *Proceedings of ACM SIGGRAPH 2000* (2000). 2, 5, 6

[Lev03] LEVIN D.: Mesh-independent surface interpolation. In *Geometric Modeling for Data Visualization* (2003). 2

[LP02] LINSEN L., PRAUTZSCH H.: Fan clouds - an alternative to meshes. In *Dagstuhl Seminar 02151 on Theoretical Foundations of Computer Vision - Geometry, Morphology and Computational Imaging* (2002). 4

[MMS*04] MOENNING C., MÉMOLI F., SAPIRO G., DYN. N., DODGSON N. A.: *Meshless geometric subdivision*. Tech. rep., IMA Preprint Series number #1977, 2004. 2

[OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.-P.: Multi-level partition of unity implicits. *ACM Transactions on Graphics 22*, 3 (July 2003), 463–470. 2

[Pau03] PAULY M.: *Point Primitives for Interactive Modeling and Processing of 3D geometry*. Master's thesis, ETH Zürich, 2003. 4

[PGK02] PAULY M., GROSS M., KOBBELT L. P.: Efficient simplification of point-sampled surfaces. In *Proceedings of the 13th IEEE Visualization Conference* (2002), pp. 163–170. 2

[PKKG03] PAULY M., KEISER R., KOBBELT L. P., GROSS M.: Shape modeling with point-sampled geometry. In *Proceedings of ACM SIGRAPH 2003* (2003), pp. 641–650. 2, 6

[RL00] RUSINKIEWICZ S., LEVOY M.: QSplat: A multiresolution point rendering system for large meshes. In *Proceedings of SIGGRAPH 2000, Computer Graphics Proceedings* (2000), pp. 343–352. 7

[Sam89] SAMET H.: *The Design and Analysis of Spatial Data Structures*. Reading, Mass.: Addison Wesley, 1989. 7

[TRS04] TOBOR I., REUTER P., SCHLICK C.: Multiresolution reconstruction of implicit surfaces with attributes from large unorganized point sets. In *Proceedings of Shape Modeling International 2004* (2004). 2

[Tur92] TURK G.: Re-tiling polygonal surface. In *Proceedings of ACM SIGGRAPH 92* (1992). 2

[UMA04] ULRICH C., MARTIN R., ALEXANDRU T.: Surface processing methods for point sets using finite elements. *Computers & Graphics 28* (2004), 851–868. 4

[VPBM01] VLACHOS A., PETERS J., BOYD C., MITCHELL J. L.: Curved pn triangles. In *Proceedings of the 2001 symposium on Interactive 3D graphics* (2001). 2, 3

[ZPvBG01] ZWICKER M., PFISTER H., VAN BAAR J., GROSS M.: Surface splatting. In *Proceedings of ACM SIGGRAPH 2001* (2001), pp. 371–378. 1

[ZRB*04] ZWICKER M., RÄSÄNEN J., BOTSCH M., DACHSBACHER C., PAULY M.: Perspective accurate splatting. In *Graphics Interface 2004* (2004). 1, 6

[ZSS96] ZORIN D., SCHRÖDER P., SWELDENS W.: Interpolating subdivision for meshes with arbitrary topology. In *Proceedings of ACM SIGGRAPH 1996* (1996), pp. 189–192. 8

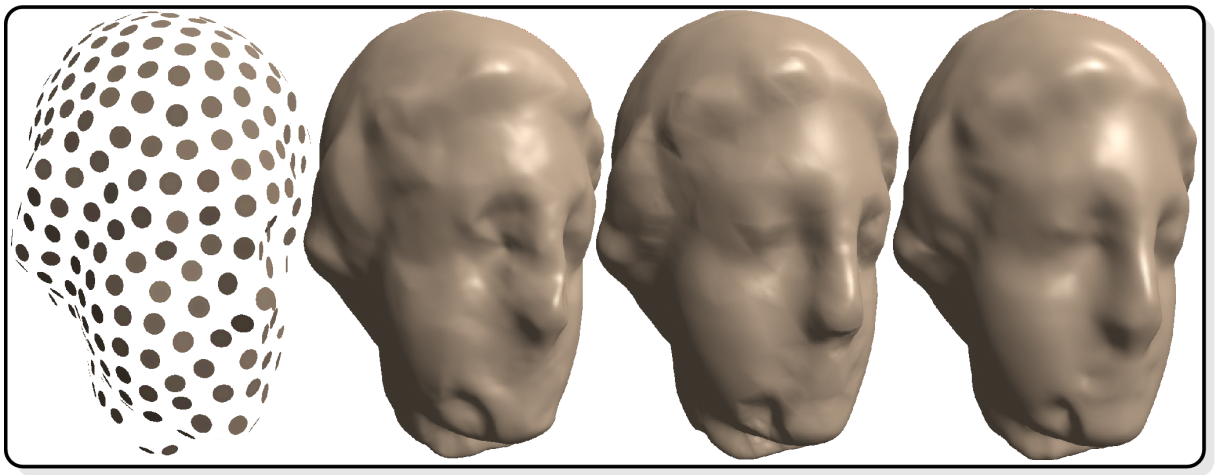

Figure 8: *The Igea model uniformly sampled by 600 points is refined to 150k points with various techniques. From left to right: the butterfly (after a meshing step), the Guennebaud et al. diadic refinement and our new $\sqrt{3}$-like refinement.*

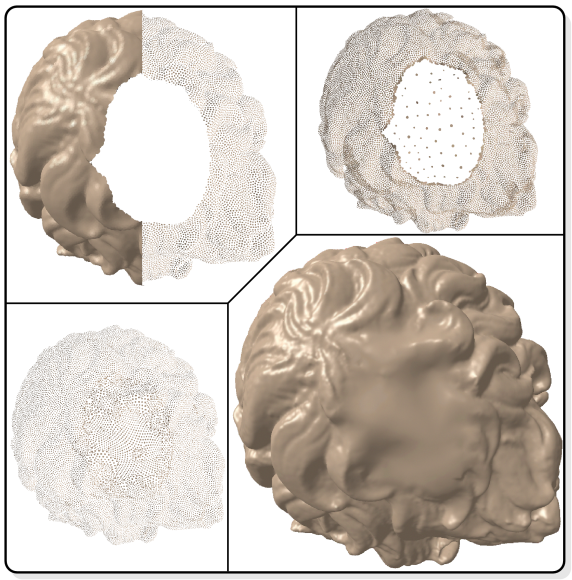

Figure 9: *Illustration of the hole filling capability of our algorithm. A large hole in the David's hair is filled by adjusting the radius of boundary points such that they are greater than the hole and applying our refinement algorithm. The final image is obtained after eight refinement steps while the two others show intermediate steps.*

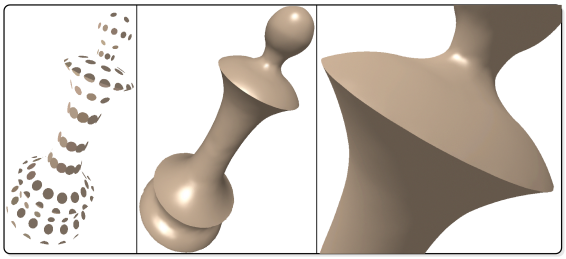

Figure 10: *Illustration of the refinement of creases.*

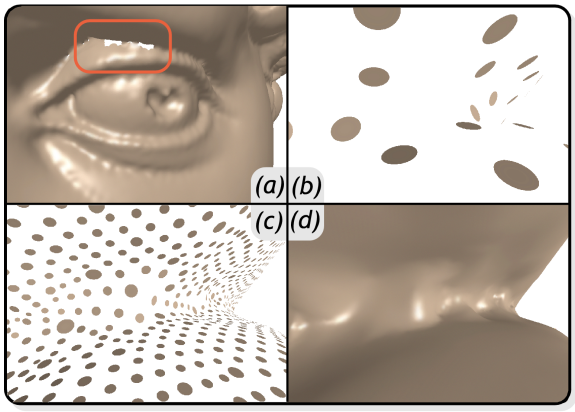

Figure 11: *Illustration of the usefulness of our "geodesic projection" and our "curved angle". (a) If they are disabled high curvature areas are not reconstructed (holes appear). The three other images are close views of the refinement process when our "geodesic projection" and "curved angle" are enabled. (b) The initial sampling. (c-d) Intermediate step and final refinement: the previous holes are smoothly reconstructed.*

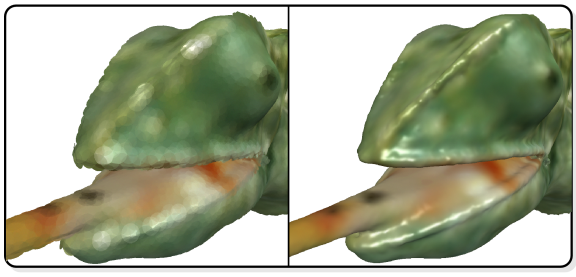

Figure 12: *Refinement of a textured model. The right image is obtained after 4 refinement steps.*