Algebraic Point Set Surfaces

Gaël Guennebaud Markus Gross
ETH Zurich

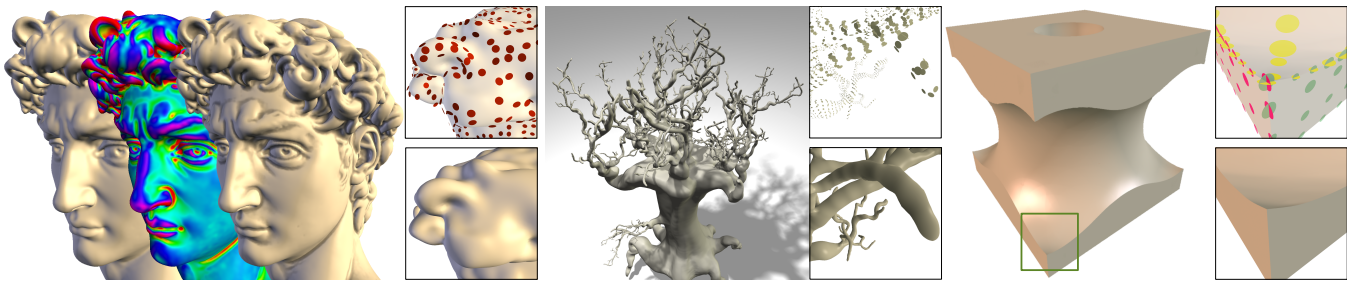

Figure 1: *Illustration of the central features of our algebraic MLS framework. From left to right: efficient handling of very complex point sets, fast mean curvature evaluation and shading, significantly increased stability in regions of high curvature, sharp features with controlled sharpness. Sample positions are partly highlighted.*

Abstract

In this paper we present a new Point Set Surface (PSS) definition based on moving least squares (MLS) fitting of algebraic spheres. Our surface representation can be expressed by either a projection procedure or in implicit form. The central advantages of our approach compared to existing planar MLS include significantly improved stability of the projection under low sampling rates and in the presence of high curvature. The method can approximate or interpolate the input point set and naturally handles planar point clouds. In addition, our approach provides a reliable estimate of the mean curvature of the surface at no additional cost and allows for the robust handling of sharp features and boundaries. It processes a simple point set as input, but can also take significant advantage of surface normals to improve robustness, quality and performance. We also present an novel normal estimation procedure which exploits the properties of the spherical fit for both direction estimation and orientation propagation. Very efficient computational procedures enable us to compute the algebraic sphere fitting with up to 40 million points per second on latest generation GPUs.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve and surface representations

Keywords: point based graphics, surface representation, moving least square surfaces, sharp features.

1 Introduction

A key ingredient of most methods in point based graphics is the underlying meshless surface representation which computes a continuous approximation or interpolation of the input point set. The by far most important and successful class of such meshless representations are point set surfaces (PSS) [Alexa et al. 2003] combining high flexibility with ease of implementation. PSS generally define a smooth surface using local moving least-squares (MLS) approximations of the data [Levin 2003]. The degree of the approximation can easily be controlled, making the approach naturally well suited to filter noisy input data. In addition, the semi-implicit nature of the representation makes PSS an excellent compromise combining advantages both of explicit representations, such as parametric surfaces, and of implicit surfaces [Ohtake et al. 2003].

Since its inception, significant progress has been made to better understand the properties and limitations of MLS [Amenta and Kil 2004a,2004b] and to develop efficient computational schemes [Adamson and Alexa 2004]. A central limitation of the robustness of PSS, however, comes from the plane fit operation that is highly unstable in regions of high curvature if the sampling rate drops below a threshold. Such instabilities include erroneous fits or the limited ability to perform tight approximations of the data. This behavior sets tight limits to the minimum admissible sampling rates for PSS [Amenta and Kil 2004b; Dey et al. 2005].

In this paper we present a novel definition of moving least squares surfaces called algebraic point set surfaces (APSS). The key idea is to directly fit a higher order algebraic surface [Pratt 1987] rather than a plane. For computational efficiency all methods in this paper focus on algebraic sphere fitting, but the general concept could be applied to higher order surfaces as well. The main advantage of the sphere fitting is its significantly improved stability in situations where planar MLS fails. For instance, tight data approximation is accomplished, spheres perform much better in the correct handling of sheet separation (figure 3) and exhibit a high degree of stability both in cases of undersampling (figure 2) and for very large weight functions. The specific properties of algebraic spheres make APSS superior to simple geometric sphere fitting. It allows us to elegantly handle planar areas or regions around inflection points as limits in which the algebraic sphere naturally degenerates to a plane.

Furthermore, the spherical fitting enables us to design interpolatory weighting schemes by using weight functions with singularities at zero while overcoming the fairness issue of previous MLS surfaces. The sphere radius naturally serves as a for-free and reliable estimate of the mean curvature of the surface. This enables us, for instance, to compute realtime accessibility shading on large input objects (figure 1).

Central to our framework are the numerical procedures to efficiently perform the sphere fit. For point sets with normals we designed

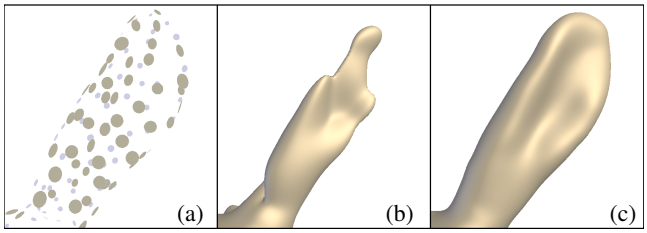

Figure 2: *The undersampled ear of the Stanford bunny (a) using normal averaging plane fit (SPSS) with $h = 1.8$ (b) and our new APSS with $h = 1.7$ (c).*

a greatly simplified and accelerated algorithm whose core part reduces to linear least squares. For point sets without normals, we employ a slightly more expensive fitting scheme to estimate the surface normals of the input data. In particular, this method allows us to improve the normal estimation by [Hoppe et al. 1992]. The tighter fit of the sphere requires on average less projection iterations to achieve the same precision making the approach even faster than the most simple plane fit MLS. Our implementation on the latest generation GPUs features a performance up to 45 millions of points per second, sufficient to compute a variety of operations on large point sets in realtime.

Finally, we developed a simple and powerful extension of [Fleishman et al. 2005] to robustly handle sharp features, such as boundaries and creases, with a built-in sharpness control (figure 1).

2 Related Work

Point set surfaces were introduced to computer graphics by [Alexa et al. 2003]. The initial definition is based on the stationary set of Levin's moving least squares (MLS) projection operator [Levin 2003]. The iterative projection involves a non-linear optimization to find the local reference plane, a bivariate polynomial fit and iteration. By omitting the polynomial fitting step, Amenta and Kil [2004a] showed that the same surface can be defined and computed by weighted centroids and a smooth gradient field. This leads to a significantly simplified implicit surface definition [Adamson and Alexa 2004] and faster algorithms, especially in the presence of normals [Alexa and Adamson 2004]. In particular, the projection can be accomplished on a plane passing through the weighted average of the neighboring samples with a normal computed from the weighted average of the adjacent normals. In the following we will refer to this efficient variant as SPSS for Simple PSS.

Boissonnat and Cazals [2000] and more specifically Shen et al. [2004] proposed a similar, purely implicit MLS (IMLS) surface

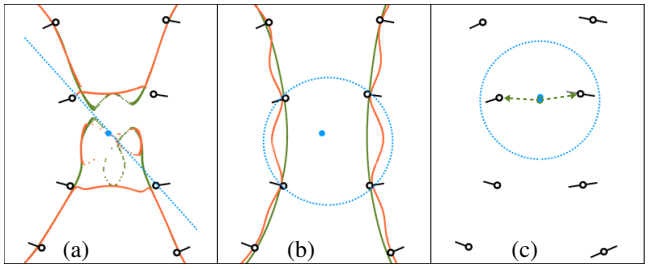

Figure 3: *Sheet separation with conventional PSS compared to our new spherical fit: (a) Standard PSS with covariance analysis (orange) and normal averaging (green). (b) Our APSS without (orange) and with (green) normal constraints. The best plane and best algebraic sphere fits for the blue point in the middle are drawn in blue. For this example the normals were computed using our technique from section 5 which can safely propagate the orientation between the two sheets (c).*

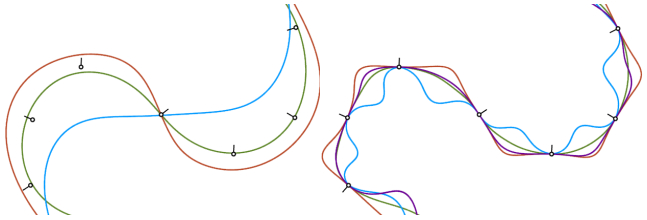

Figure 4: *A 2D oriented point set is approximated (left) and interpolated (right) using various PSS variants: SPSS (blue), IMLS (red), HPSS (magenta) and our APSS (green).*

representation defined by a local weighted average of tangential implicit planes attached to each input sample. This implicit surface definition was initially designed to reconstruct polygon soups, but the point cloud case was recently analyzed by Kolluri [2005] and made adaptive to the local feature size by Dey et al. [2005]. Note that in this paper IMLS refers to the simple definition given in [Kolluri 2005]. If applied without polygon constraints and in the case of sparse sampling, we found that the surface can grow or shrink significantly as a function of the convexity or concavity of the point cloud. To alleviate this problem, Alexa and Adamson [2006] enforced convex interpolation of an oriented point set using singular weight functions and Hermite centroid evaluations (HPSS).

Likewise relevant to our research is prior art on normal estimation. Many point based algorithms, including some of the aforedescribed PSS variants, require surface normals. A standard procedure proposed by [Hoppe et al. 1992] is to estimate their directions using a local plane fit followed by a propagation of the orientations using a *minimum spanning tree* (MST) strategy and a transfer heuristic based on normal angles. Extending this technique, Mitra and Nguyen [2003] take into account the local curvature and the amount of noise in order to estimate the optimal size of the neighborhood for the plane fit step. However, the inherent limitations of the plane fit step and the propagation heuristic require a very dense sampling rate in regions of high curvature.

Intrinsically, a PSS can only define a smooth closed manifold surface. Even though boundaries could be handled by thresholding an *off-center* distance [Adamson and Alexa 2004], obtaining satisfactory boundary curves with such an approach is usually difficult or even impossible. In order to detect and reconstruct sharp creases and corners in a possibly noisy point cloud, Fleishman et al. [2005] proposed a refitting algorithm that locally classifies the samples into multiple pieces of surfaces according to discontinuities of the derivative of the surface. While constituting an important progress, the method requires very dense point clouds, it is rather expensive, and it offers only limited flexibility to the user. Also, potential instabilities in the classification can create discontinuous surface parts. A second important approach is the Point-Sampled Cell Complexes [Adamson and Alexa 2006b] which allows to explicitly represent sharp features by decomposing the object into cells of different dimensions. While this approach allows to handle a wide variety of cases (e.g., non-manifoldness), the decomposition and the balancing of the cells' influence on the shape of the surface demands effort by the user, making the method unsuitable for some applications.

As we will elaborate in the following sections, the algebraic sphere fitting overcomes many of the aforedescribed limitations by the significantly increased robustness of the fit in the presence of low sampling rates and high curvature.

3 Overview of the APSS framework

Given a set of points $P = \{\mathbf{p}_i \in \mathbb{R}^d\}$, we define a smooth surface S_P approximating P using a moving least squares spherical fit to the data. Our approach handles both simple point clouds and point

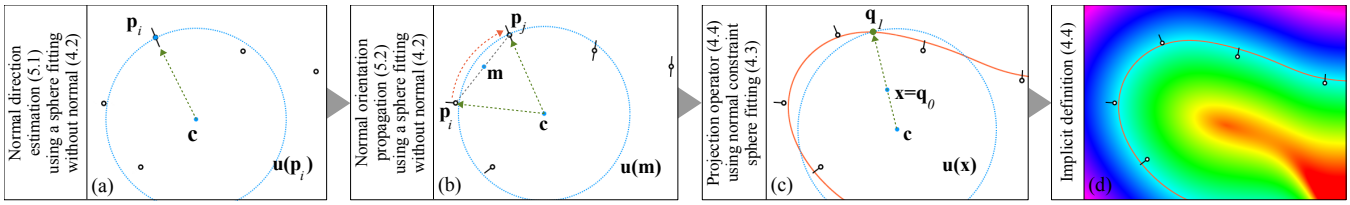

Figure 5: *Overview of our APSS framework. (a) Evaluation of the normal direction at point* $\mathbf{p}_i$*. (b) Propagation of a consistent normal orientation from* $\mathbf{p}_i$ *to* $\mathbf{p}_j$*. (c) First iteration of the projection of the point* $\mathbf{x}$ *onto the APSS. (d) Illustration of the scalar field defined by APSS.*

clouds with normals. The presence of surface normals leads to simplified, more efficient, and more robust fitting algorithms (see section 4.3). We recommend to estimate them from the input point set using the procedures of section 5 and section 4.2 as a preprocessing step.

Figure 5 presents the procedural flow using a 2D example. Starting from a simple point cloud, we first evaluate the normal directions by locally fitting an algebraic sphere $\mathbf{u}$ at each point $\mathbf{p}_i$ (figure 5a). Next, a consistent normal orientation is robustly propagated from the point $\mathbf{p}_i$ to its neighbor $\mathbf{p}_j$ by approximating the surface in-between using an approximating sphere (figure 5b). Given the estimated normals, a simplified spherical fitting technique can be applied to compute our final APSS S_P which can be defined as the set of stationary points of a projection operator projecting a point onto a locally fitted sphere (figure 5c). As an alternative our surface S_P can also be defined as the zero set of a scalar field representing the distance between the evaluation point and a locally fitted sphere (figure 5d). The former definition is well suited for resampling, while the latter one is convenient to raytrace the surface [Adamson and Alexa 2003; Wald and Seidel 2005] or to perform Boolean operations [Pauly et al. 2003].

The key ingredients of our framework are the computational algorithms to robustly and efficiently fit a sphere to a set of points in a moving least squares fashion. These methods will be presented for the cases of a simple point cloud as well as for a point cloud with normals in sections 4.2 and 4.3. Based on the procedures for algebraic sphere fits, we will define and discuss our APSS definition in section 4.4. Two significant extensions include normal estimation in section 5 and the handling of sharp features in section 6.

4 Sphere Fitting and APSS

In this section we will discuss the core mathematical definitions and computational procedures of our APSS framework. We will elaborate on the sphere fitting problem and show how previous methods can be adapted to define a MLS surface based on sphere fits. As a central computational algorithm, we will present a novel and very efficient sphere fitting method taking into account surface normals.

4.1 General Issues

Weighting scheme. Throughout the paper we will utilize the following generic weight function

$$w_i(\mathbf{x}) = \phi\left(\frac{\|\mathbf{p}_i - \mathbf{x}\|}{h_i(\mathbf{x})}\right) \tag{1}$$

describing the weight of the point $\mathbf{p}_i$ for the local evaluation position $\mathbf{x}$. ϕ is a smooth, decreasing weight function and $h_i(\mathbf{x})$ describes the local feature size. $h_i(\mathbf{x})$ can be constant, depend on $\mathbf{x}$ as in [Pauly et al. 2003], or only on $\mathbf{p}_i$, i.e., $h_i(\mathbf{x}) = h_i$ as in [Adamson and Alexa 2006a]. As a proper choice of ϕ we suggest the following compactly supported polynomial

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

This function performs a smooth data approximation and avoids square root evaluations for the distance computation. Interpolation can be achieved using functions with a singularity at zero, see [Shen et al. 2004; Adamson and Alexa 2006b].

Spherical fitting. The problem of fitting a sphere to a set of points is not new and several methods have been proposed in the past. It is important to distinguish *geometric* approaches from *algebraic* ones. Geometric fitting denotes algorithms minimizing the sum of the squared Euclidean distances to the given points (e.g., see [Gander et al. 1994]). Geometric sphere fits have several drawbacks. First, because of the non-linear nature of the problem, the solution can only be found using an iterative, expensive approach. More importantly, since these algorithms are either based on a center-radius or parametric representation of the sphere, they become unstable when the best fit tends to a plane. This limits the practical utility of such methods for our purposes.

An elegant alternative is to substitute the geometric distance by an algebraic distance as in [Pratt 1987]. An algebraic sphere is thus defined as the 0-isosurface of the scalar field $s_{\mathbf{u}}(\mathbf{x}) = [1, \mathbf{x}^T, \mathbf{x}^T\mathbf{x}]\mathbf{u}$, where $\mathbf{u} = [u_0, ..., u_{d+1}]^T \in \mathbb{R}^{d+2}$ is the vector of scalar coefficients describing the sphere. For $u_{d+1} \neq 0$ the corresponding center $\mathbf{c}$ and radius r are easily computed as:

$$\mathbf{c} = -\frac{1}{2u_{d+1}}[u_1, ..., u_d]^T, \quad r = \sqrt{\mathbf{c}^T\mathbf{c} - u_0/u_{d+1}} \tag{3}$$

with d being the dimension. In degenerate cases, $\mathbf{u}$ corresponds to the coefficients of a plane equation with u_0 representing the plane's distance from the origin and $[u_1, .., u_d]^T$ being its normal.

4.2 Fitting Algebraic Spheres Without Normals

Let n be the number of points and let $\mathbf{W}(\mathbf{x})$ and $\mathbf{D}$ respectively be the $n \times n$ diagonal weight matrix and the $n \times (d+2)$ design matrix defined as:

$$\mathbf{W}(\mathbf{x}) = \begin{bmatrix} w_0(\mathbf{x}) & & \\ & \ddots & \\ & & w_{n-1}(\mathbf{x}) \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} 1 & \mathbf{p}_0^T & \mathbf{p}_0^T\mathbf{p}_0 \\ \vdots & \vdots & \vdots \\ 1 & \mathbf{p}_{n-1}^T & \mathbf{p}_{n-1}^T\mathbf{p}_{n-1} \end{bmatrix}. \tag{4}$$

Then the solution $\mathbf{u}(\mathbf{x})$ of our algebraic sphere fit at a given point $\mathbf{x} \in \mathbb{R}^d$ can be expressed as:

$$\mathbf{u}(\mathbf{x}) = \underset{\mathbf{u},\, \mathbf{u}\neq\mathbf{0}}{\arg\min} \left\| \mathbf{W}^{\frac{1}{2}}(\mathbf{x})\mathbf{D}\mathbf{u} \right\|^2. \tag{5}$$

In order to avoid the trivial solution $\mathbf{u}(\mathbf{x}) = 0$, $\mathbf{u}$ has to be constrained by a metric. The choice of this constraint significantly influences the solution of the above minimization problem. Ideally we want the algebraic fit to behave as closely as possible to a geometric fit, but at the same time to reliably handle the planar case. Among all constrained methods that have been proposed, we found Pratt's constraint [Pratt 1987] the most suitable for our purposes.

Pratt's constraint fixes the norm of the gradient at the surface of the sphere to unit length 1. This is accomplished by $\|(u_1, ..., u_d)\|^2 - 4u_0u_{d+1} = 1$ and ensures that the algebraically fitted sphere is close

to the least squares Euclidean best fit, i.e., the geometric fit (see also figure 6). By rewriting the above constraint in matrix form $\mathbf{u}^T\mathbf{C}\mathbf{u} = 1$, the solution $\mathbf{u}(\mathbf{x})$ of our minimization problem yields as the eigenvector of the smallest positive eigenvalue of the following generalized eigenproblem:

$$\mathbf{D}^T\mathbf{W}(\mathbf{x})\mathbf{D}\,\mathbf{u}(\mathbf{x}) = \lambda\mathbf{C}\,\mathbf{u}(\mathbf{x}), \text{ with } \mathbf{C} = \begin{bmatrix} 0 & 0 & \cdots & 0 & -2 \\ 0 & 1 & & & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & & & 1 & 0 \\ -2 & 0 & \cdots & 0 & 0 \end{bmatrix} . \quad (6)$$

Note that this method is only used to estimate the missing normals of input point sets. An efficient algorithm for running the APSS in the presence of normals will be derived in the following section.

4.3 Fitting Spheres to Points with Normals

While the problem of fitting a sphere to points has been investigated extensively, there exists to our knowledge no method to take specific advantage of normal constraints at the sample positions. We derive a very efficient algorithm by adding the following derivative constraints $\nabla s_{\mathbf{u}}(\mathbf{p}_i) = \mathbf{n}_i$ to our minimization problem (5). Note that, by definition, it holds that $\|\mathbf{n}_i\| = 1$, which constrains both direction and magnitude of the normal vector. This is especially important to force the algebraic distance to be close to the Euclidean distance for points close to the surface of the sphere. Furthermore, the normal constraints lead to the following standard *linear* system of equations that can be solved very efficiently:

$$\mathbf{W}^{\frac{1}{2}}(\mathbf{x})\mathbf{D}\,\mathbf{u} \quad = \quad \mathbf{W}^{\frac{1}{2}}(\mathbf{x})\,\mathbf{b} \quad (7)$$

where

$$\mathbf{W}(\mathbf{x}) = \begin{bmatrix} \ddots & & & \\ & w_i(\mathbf{x}) & & \\ & & \beta w_i(\mathbf{x}) & \\ & & & \ddots \\ & & \beta w_i(\mathbf{x}) & \\ & & & \ddots \end{bmatrix}, \; \mathbf{D} = \begin{bmatrix} \vdots & \vdots & \vdots \\ 1 & \mathbf{p}_i^T & \mathbf{p}_i^T\mathbf{p}_i \\ 0 & \mathbf{e}_0^T & 2\mathbf{e}_0^T\mathbf{p}_i \\ \vdots & \vdots & \vdots \\ 0 & \mathbf{e}_{d-1}^T & 2\mathbf{e}_{d-1}^T\mathbf{p}_i \\ \vdots & \vdots & \vdots \end{bmatrix}, \; \mathbf{b} = \begin{bmatrix} \vdots \\ 0 \\ \mathbf{e}_0^T\mathbf{n}_i \\ \vdots \\ \mathbf{e}_{d-1}^T\mathbf{n}_i \\ \vdots \end{bmatrix} . \quad (8)$$

Here, $\{\mathbf{e}_k\}$ denote the unit basis vectors of our coordinate system. The scalar β allows us to weight the normal constraints. We will discuss the proper choice of this parameter subsequently. We solve this equation using the pseudo-inverse method, i.e.:

$$\mathbf{u}(\mathbf{x}) = \mathbf{A}^{-1}(\mathbf{x})\hat{\mathbf{b}}(\mathbf{x}) \quad (9)$$

where both the $(d+2)\times(d+2)$ weighted covariance matrix $\mathbf{A}(\mathbf{x}) = \mathbf{D}^T\mathbf{W}(\mathbf{x})\mathbf{D}$ and the vector $\hat{\mathbf{b}}(\mathbf{x}) = \mathbf{D}^T\mathbf{W}(\mathbf{x})\mathbf{b}$ can be computed directly and efficiently by weighted sums of the point coefficients.

The issue of affine invariance of this method deserves some further discussion. While the method's invariance under translations and rotations is trivial, the mix of constraints representing algebraic distances and distances between unit vectors makes it slightly sensitive to scale. A simple and practical solution is to choose a large value for β, e.g. $\beta = 10^6 h(\mathbf{x})^2$ where $h(\mathbf{x}) = \frac{\sum_i w_i(\mathbf{x})h_i(\mathbf{x})}{\sum_i w_i(\mathbf{x})}$ is a smooth function describing the local neighborhood size (1). This effectively assigns very high importance to the derivative constraints, which, prescribed at given positions and with a fixed norm, are sufficient to fit a spherical isosurface to the data. The positional constraints still specify the actual isovalue. Not only does this choice leave the fitting invariant under scale, but it also makes it more stable, much less prone to oscillations (figure 3b) and less sensitive to outliers. This choice does not unbalance the relative importance of the positions and normals of the samples because the derivative constraints depend on both the sample positions and normals.

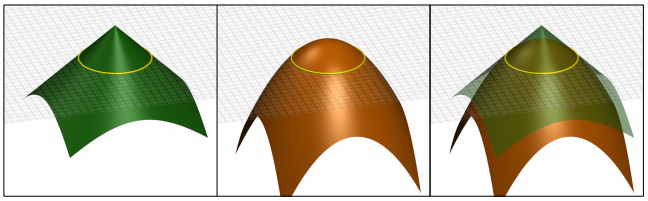

Figure 6: *The Euclidean distance (green cone) versus an algebraic distance (orange paraboloid) to the 2D yellow circle. Pratt's normalization makes these two surfaces tangent to each other at the given circle.*

4.4 Algebraic Point Set Surfaces

Implicit surface definition. The previous sections provide all ingredients to define an MLS surface based on sphere fits. Our APSS S_P, approximating or interpolating the point set $P = \{\mathbf{p}_i \in \mathbb{R}^d\}$, yields as the zero set of the implicit scalar field $f(\mathbf{x})$ representing the algebraic distance between the evaluation point $\mathbf{x}$ and the fitted sphere $\mathbf{u}(\mathbf{x})$. This field is illustrated in figure 5d:

$$f(\mathbf{x}) = s_{\mathbf{u}(\mathbf{x})}(\mathbf{x}) = \left[1,\, \mathbf{x}^T,\, \mathbf{x}^T\mathbf{x}\right]\mathbf{u}(\mathbf{x}) = 0\,. \quad (10)$$

If needed the actual Euclidean distance to the fitted algebraic surface can be computed easily by its conversion to an explicit form.

Gradient. This implicit definition allows us to conveniently compute the gradient of the scalar field which is needed to obtain the surface normal or to perform an orthogonal projection. We compute the gradient $\nabla f(\mathbf{x})$ as follows:

$$\nabla f(\mathbf{x}) = [1,\, \mathbf{x}^T,\, \mathbf{x}^T\mathbf{x}]\nabla\mathbf{u}(\mathbf{x}) + \begin{bmatrix} 0 & \mathbf{e}_0^T & 2\mathbf{e}_0^T\mathbf{x} \\ \vdots & \vdots & \vdots \\ 0 & \mathbf{e}_{d-1}^T & 2\mathbf{e}_{d-1}^T\mathbf{x} \end{bmatrix}\mathbf{u}(\mathbf{x}) . \quad (11)$$

The value of the gradient $\nabla\mathbf{u}(\mathbf{x}) = \left(\frac{d\mathbf{u}(\mathbf{x})}{dx_0}, \frac{d\mathbf{u}(\mathbf{x})}{dx_1}, \ldots\right)$ depends on the fitting method. With the Pratt's constraint, the derivatives can be computed as the plane fit with covariance analysis case (see [Alexa and Adamson 2004]). With our normal constraint, the derivatives can be directly computed from equation (9):

$$\frac{d\mathbf{u}(\mathbf{x})}{dx_k} \quad = \quad \mathbf{A}^{-1}(\mathbf{x})\left(-\frac{d\mathbf{A}(\mathbf{x})}{dx_k}\mathbf{u}(\mathbf{x}) + \frac{d\hat{\mathbf{b}}(\mathbf{x})}{dx_k}\right) . \quad (12)$$

Curvature. The implicit definition can be used to compute higher order differential surface operators, such as curvature. In practice, however, the evaluation of the shape matrix involves expensive computations. Our sphere fit provides an elegant estimate of the mean curvature readily available by the radius of the fitted sphere (3). This estimate of the mean curvature is in general very accurate, except when two pieces of a surface are too close to each other. In this case the samples of the second surface can lead to an overestimation of the curvature. Note that in such cases our normal constraint fitting method still reconstructs a correct surface without oscillations (figure 3b). The sign of this inexpensive mean curvature estimate is determined by the sign of u_{n+1} and can be utilized for a variety of operations, such as accessibility shading shown in figure 1.

Projection procedure. Since the presented APSS surface definition is based on a standard MLS, all the projection operators for the plane case can easily be adapted to our setting by simply replacing the planar projection by a spherical projection. Following the concept of *almost orthogonal* projections in [Alexa and Adamson 2004] we recommend the following procedure as a practical recipe to implement APSS: Given a current point $\mathbf{x}$, we iteratively define a series of points $\mathbf{q}_i$, such that $\mathbf{q}_{i+1}$ is the orthogonal projection of $\mathbf{x}$ onto the algebraic sphere defined by $\mathbf{u}(\mathbf{q}_i)$. Starting with $\mathbf{q}_0 = \mathbf{x}$,

the projection of $\mathbf{x}$ onto the surface is asymptotically $\mathbf{q}_\infty$. The recursion stops when the displacement is lower than a given precision threshold. This procedure is depicted in figure 5c.

Including surface normals into the definition comes with additional significant advantages. First, it makes the scalar field $f(\mathbf{x})$ to be consistently signed according to the inside/outside relative position of $\mathbf{x}$. Such consistent orientation is fundamental to perform boolean operations or collision detections for instance. Furthermore, point-normals provide a first order approximation of the underlying surface and thus allow for smaller neighborhoods at the same accuracy. Tighter neighborhoods do not only increase stability, but also accelerate the processing. To this end, we recommend the normal estimation procedure described in the following section to preprocess a raw input point cloud.

5 Estimation of Surface Normals

Our method for normal estimation draws upon the same principle as the one proposed by Hoppe et al. [1992]. The novelties include the normal direction estimation and new heuristics for the propagation of the normal orientation. We will focus on these two issues.

5.1 Normal Direction

To estimate the normal direction $\mathbf{n}_i$ of a point $\mathbf{p}_i$ of our input point set, we essentially take the gradient direction at $\mathbf{p}_i$ of our MLS surface definition described in the previous section, i.e., $\mathbf{n}_i = \nabla f(\mathbf{p}_i)$. Here, the computation of the algebraic sphere is based on the eigenvector method described in section 4.2. In practice, we can approximate the accurate gradient direction of the scalar field f (appendix) with the gradient of the fitted algebraic sphere $\mathbf{u}(\mathbf{p}_i)$, i.e.:

$$\mathbf{n}_i \approx \nabla s_{\mathbf{u}(\mathbf{p}_i)}(\mathbf{p}_i) = \begin{bmatrix} 0 & \mathbf{e}_0^T & 2\mathbf{e}_0^T\mathbf{p}_i \\ \vdots & \vdots & \vdots \\ 0 & \mathbf{e}_{d-1}^T & 2\mathbf{e}_{d-1}^T\mathbf{p}_i \end{bmatrix} \mathbf{u}(\mathbf{p}_i)\,. \qquad (13)$$

This step is illustrated in figure 5a. Although the sphere normal can differ from the actual surface normal, the above simplification is reasonable, because the approximation is usually very close to the accurate surface normal for points evaluated close to an input sample. In practice, we never experienced any problems with it.

During this step we also store for each point a confidence value μ_i that depends on the sanity of the neighborhood used to estimate the normal direction. We take the normalized residual value, i.e., $\mu_i = \bar{\lambda}/\sum_{k=0}^{d+1}|\lambda_k|$ where λ_k are the $d+2$ eigenvalues and $\bar{\lambda}$ is the smallest positive one. Similar confidence estimates were used before by several authors (e.g. [Pauly et al. 2004]) in combination with covariance analysis to perform adaptive resampling. We will use this confidence value in the next section for our propagation algorithm.

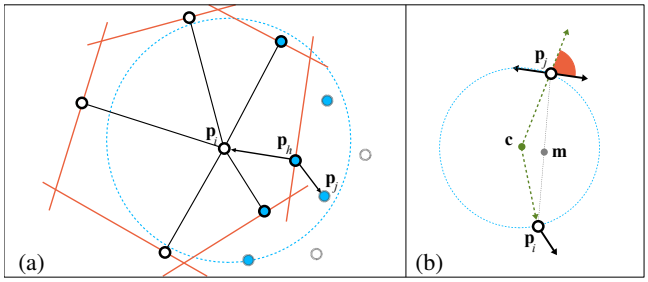

Figure 7: *(a) Illustration of the BSP neighborhood of a point $\mathbf{p}_i$: each neighbor defines a halfspace limiting the neighborhood. The blue points correspond to a naive kNN with $k = 6$. (b) Illustration of Ψ_{ij}. The angle between the gradient of the fitted sphere (green arrows) and the normal direction (shown in red here) serves as a measure for the reliability of the normal propagation.*

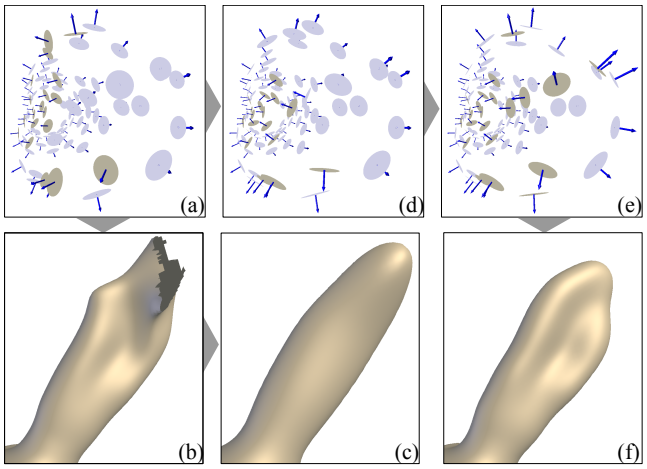

Figure 8: *(a) Closeup view of the tip of the ear of figure 2 after recomputing the normals using our method. A few normals are incorrectly oriented and eventually break the APSS reconstruction (b) when using a weight radius of $h = 2$. A smoothed manifold reconstruction (c) can still be obtained when increasing the weight radius significantly ($h = 3.5$). Figures (d) and (e) show the normals after 15 and 30 iterations of our normal optimization procedure respectively. (f) APSS reconstruction of (e) with $h = 2$.*

5.2 Propagation of Orientation

Our second contribution concerns the propagation of normal orientation. Following [Hoppe et al. 1992], we use a *minimum spanning tree* (MST) strategy which is initialized by setting the normal orientation of a single point in contact with the point set's bounding box to point outside.

The major difference of our approach lies in the propagation method and the weight function for the MST. To propagate the orientation from a point $\mathbf{p}_i$ of normal $\mathbf{n}_i$ to a point $\mathbf{p}_j$ of normal $\mathbf{n}_j$ we locally fit an algebraic sphere at $\mathbf{m} = \frac{\mathbf{p}_i+\mathbf{p}_j}{2}$ halfway between the two points. Figures 3c and 5b serve for illustration. The sphere gives us a reliable approximation of the surface *connecting* the two points. It hence allows us to reliably transfer a given normal orientation even across sharp features or along two close surface sheets. The normal $\mathbf{n}_j$ of the point $\mathbf{p}_j$ is flipped if $\nabla s_{\mathbf{u}(\mathbf{m})}(\mathbf{p}_i)^T\mathbf{n}_i \cdot \nabla s_{\mathbf{u}(\mathbf{m})}(\mathbf{p}_j)^T\mathbf{n}_j < 0$.

The purpose of the weight function of the MST is to favor edges along which it is safe to transfer the normal orientation using the previous procedure. The MST assigns an edge i-j if and only if the two involved samples $\mathbf{p}_i$ and $\mathbf{p}_j$ are neighbors. Unlike Hoppe's k-nearest neighborhood definition (kNN), we utilize a somewhat more elaborate *BSP neighborhood* definition. This definition removes from a large kNN of the point $\mathbf{p}_i$ every point that is *behind* another neighbor, i.e., every point $\mathbf{p}_j$ such that there exist a third point $\mathbf{p}_h$ satisfying $(\mathbf{p}_i - \mathbf{p}_h)^T(\mathbf{p}_j - \mathbf{p}_h) < 0$. Figure 7a depicts an example. This neighborhood definition allows us to select reliably the closest neighbors in each direction, even in cases of strongly non-uniform sampling. In addition it avoids that the propagation procedure *jumps* over relevant samples.

To compute the weight of an edge i-j we consider the following two terms: The first one takes the sum of the confidence values of its adjacent vertices: $\mu_{ij} = \mu_i + \mu_j$. The rationale of this heuristic is

to push the samples with the lowest confidence values to the bottom (i.e. the leaves) of the tree. The second term Ψ_{ij} is computed by first fitting a sphere at the center $\mathbf{m}$ of the edge and by quantifying the difference between the gradient of the sphere and the estimated normal direction. Ψ_{ij} is defined as the dot product between the two vectors, i.e., $\Psi_{ij} = 1 - (|\nabla s_{\mathbf{u}(\mathbf{m})}(\mathbf{p}_i)^T \mathbf{n}_i| + |\nabla s_{\mathbf{u}(\mathbf{m})}(\mathbf{p}_j)^T \mathbf{n}_j|)/2$. As illustrated in figure 7b, this confirms our intuition that the propagation of normal directions is less obvious when the two vectors deviate strongly. We finally weight each edge i-j by a weighted sum of μ_{ij} and Ψ_{ij}, an empirical choice being $8\mu_{ij} + \Psi_{ij}$.

For some rare difficult cases, we propose to use the actual normals to further optimize the normals with low confidence. This is accomplished by using our stable spherical fitting method from section 4.3. For the fit, we weight each normal constraint according to its respective confidence value. This ensures that samples of higher confidence have increased importance for the fit. To further optimize accuracy, we process the samples starting with the ones having the highest average confidence of their neighborhood. Our normal estimation procedure is depicted figure 8.

6 Sharp Features

In this section we present a new approach to handle sharp features such as creases, corners, borders or peaks. We will explain it in the context of APSS, but the discussed techniques can easily be used with any PSS definition. Our approach combines the local CSG rules proposed in [Fleishman et al. 2005] with a precomputed partial classification, and we present extensions for the robust handling of borders and peaks. During the actual projection of a point $\mathbf{x}$ onto the surface, we first group the selected neighbors by tag values. Samples without tags are assigned to all groups. Then, the APSS is executed for each group and the actual point is projected onto each algebraic sphere. Next, we detect for each pair of groups whether the sharp crease they form originates from a boolean intersection or from a union and apply the corresponding CSG rule. We refer to [Fleishman et al. 2005] for the details of this last step, and focus on the novel aspects of our approach.

6.1 Tagging a Point Cloud

In a first preprocessing step or at run time, we compute a partial classification of the samples by assigning a tag value to each sample in the adjacency of a sharp feature. Two adjacent samples must have the same tag if they are on the same side of the discontinuity and a different tag otherwise. This classification can be achieved in several ways from fully automatic to entirely manual processing. Tags can easily be set automatically during a CSG operation [Pauly et al. 2003; Adams and Dutré 2003]. Tags can also be conveniently painted by using a brush tool. Figure 9 displays a simple tag stroke. Such interactive painting provides full user control over the position of sharp features and it can be applied very efficiently.

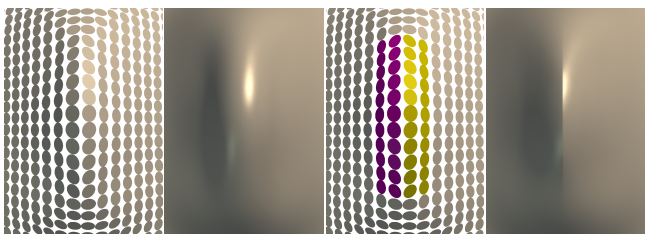

Figure 9: *A sharp crease is obtained by painting two strokes on an initially smooth point cloud.*

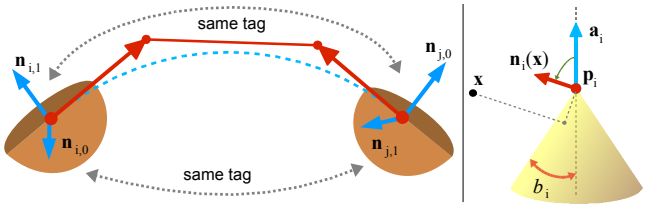

Figure 10: *Left: A cubic Bézier curve connects two samples to propagate the tagging and to correctly align the respective normals. Right: Definition of a* peak sample $\mathbf{p}_i$ *of axis* $\mathbf{a}_i$ *and angle* b_i*. Its normal* $\mathbf{n}_i(\mathbf{x})$ *depends on the actual evaluation point* $\mathbf{x}$*.*

Automatic Tagging by Local Classification

Another way to automatically tag the point cloud would be to use the local classification scheme of Fleishman et al. [2005]. While being very efficient, this method, however, cannot guarantee a consistent global classification. Therefore, we developed the following grouping method that guarantees global consistency. The local classification is applied to each sample in a breadth first order of the Euclidean minimum spanning tree, each sample counting the number of times it is assigned a given tag value. If only one group is found (smooth part) then the algorithm continues with the next sample. Otherwise, the tag value of each group is determined from the tag value with the maximum count of all samples. In addition, each local group must have different tag values. If required, new tags are created. Next, the group tags are propagated to their samples. The final tag value of a sample is the one of maximum count.

Tagging from "Sharp" Points

A common way to represent sharp features in point sampled geometry is to use "sharp" points, i.e., points with multiple normals [Pauly et al. 2003; Wicke et al. 2004; Guennebaud et al. 2005]. Usually, two normals are assigned to a point on a crease and three normals to a corner point. We handle such explicit representations within our framework by the following two step tagging procedure:

The *first* step focuses on tagging the normals of the sharp samples only. For simplicity, we will first only consider the case of edge points, i.e., points having two normals. We start with any sharp sample, assign a different tag value to each of its normals, and propagate the tag values along the crease lines into each direction. Our propagation method is mainly based on the construction of an interpolating cubic Bézier curve between sharp samples. This enables us to safely and naturally handle high curvature of the crease lines. Such a curve can be constructed easily by taking the cross product of the two normals of the samples as the curve's end tangent vectors. Figure 10 shows the idea.

We scale the tangent vectors such that their length is equal to the third of the Euclidean distance between the two samples and orient them such that the length of the resulting curve is minimal. The best possible successor of a sharp point $\mathbf{p}_i$ along a crease line is thus the sharp point $\mathbf{p}_j$ that leads to the shortest curve connecting it to $\mathbf{p}_i$.

In practice, we approximate the length of the Bézier curve by taking the length of its control polygon. We also propagate the tags of the point's normals through tangent orientation. If the normals of the successor are already tagged, the propagation for this branch stops, and the corresponding tag values are marked. Otherwise, the procedure continues by searching the best successor of $\mathbf{p}_j$ in the outgoing direction until all sharp samples are tagged. Handling corner points is accomplished considering them as three distinct edge samples, and when a corner is reached, a new tag value is assigned to the third normal.

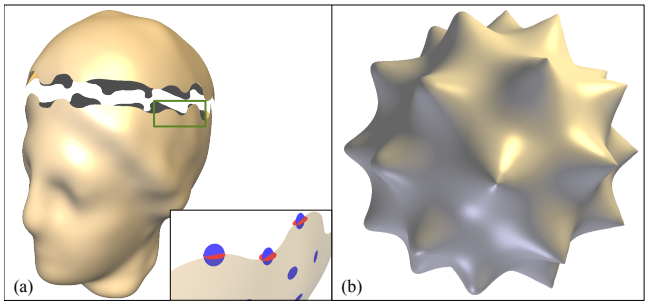

Figure 11: *(a) Illustration of boundaries handled using "clipping samples" and an interpolatory weight function. (b) Illustration of peaks handled without any treatment (left part) and with our special "peak" samples (right part).*

The *second* step involves a propagation of the tag values to the samples close to a sharp edge, i.e., where distance is defined by the actual weighting scheme. Tagging only the samples close to a sharp crease specifically enables us to deal with non-closed creases as in figure 9. If $\mathbf{p}_i$ is a smooth point and $\mathbf{p}_j$ its closest sharp point having a non-zero intersection of influence radii, the tag value of $\mathbf{p}_i$ yields as the tag value of the normal of $\mathbf{p}_j$ defining the plane closest to $\mathbf{p}_i$. A result of this procedure is depicted in figure 1-right.

6.2 Extensions

Object boundaries can be treated using "clipping samples". A set of clipping points defines a surface that is only used to clip another surface. In order to simplify both the representation and the user control, we place such clipping samples at the positions of boundary samples. Similar to the above notion of sharp points it is sufficient to add a *clipping normal* to each sample at the desired boundary. A clipping normal is set orthogonal to the desired boundary curve and should be tangent to the surface. Figure 11a illustrates the concept. Obviously, clipping samples may also be tagged to define boundary curves with discontinuities. Finally, the evaluation of the surface or the projection onto the surface only requires a minor additional modification. Clipping samples are always treated in separate groups and the reconstructed clipping surface is only used to clip the actual surface by a local intersection operation. Automatic detection of boundaries in a point cloud can easily be done by analyzing the neighborhood of each sample, such as in [Guennebaud et al. 2005].

Handling "peak" discontinuities also deserves some special attention. We propose to explicitly represent a peak by a point $\mathbf{p}_i$ equipped with an axis $\mathbf{a}_i$ and an angle b_i, i.e., by a cone, as illustrated in figure 10. During the local fitting step, we take as the normal $\mathbf{n}_i(\mathbf{x})$ of such a peak sample the normal of the cone in the direction of the evaluation point $\mathbf{x}$. $\mathbf{n}_i(\mathbf{x})$ thus yields as the unit vector $\mathbf{a}_i$ rotated by angle $\frac{\pi}{2} - b_i$ and axis $\mathbf{a}_i \times (\mathbf{x} - \mathbf{p}_i)$. We also guarantee that the surface will be C^0 at the peak by attaching an interpolating weight function to peak samples. The results in figure 11b demonstrate that our method produces high quality peaks.

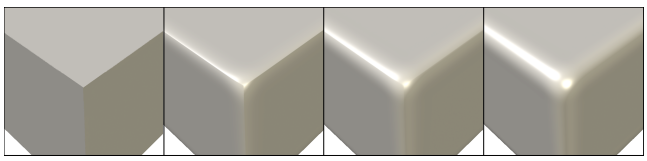

Figure 12: *Illustration of the sharpness control, from left to right:* $\alpha = 0$, $\alpha = 0.15$, $\alpha = 0.5$, $\alpha = 1$.

The sharpness control of a crease feature by a user parameter can easily be added to our concept. To this end we insert all samples into all groups but assign a lower weight to a sample when inserted into a group with a tag different from its own one. We define a global or local parameter $\alpha \in [0,1]$ which modulates the weight of these samples. For $\alpha = 0$ we obtain a fully sharp feature, for $\alpha = 1$ the feature is completely smoothed since all groups are weighted identically. Values in-between permit a continuous transition from smooth to sharp (figure 12).

7 Results

7.1 Implementation and Performance

In addition to a software implementation, we accelerated the projection operator described in section 4.4 on a GPU. Our implementation relies on standard GPGPU techniques and computes all the steps of the projection, including neighbor queries, the fitting step and the orthogonal projection, in a single shader. We use grids of linked lists as spatial acceleration structures. Our implementation performs about 45 million projection iterations per second on a NVidia GeForce 8800-GTS for APSS, compared to 60 million for SPSS. As detailed in table 1, our APSS projection operator converges about *two times* faster than SPSS making APSS overall about 1.5 times faster for the same precision. As we can see, a single iteration is usually sufficient to obtain a reasonable precision, while SPSS achieves a similar accuracy only after 2 to 3 iterations.

# iter.	1	2	3	4	5	6
APSS	2.01e-4	3.72e-5	1.9e-5	1.53e-5	1.38e-5	1.28e-5
SPSS	1.94e-3	4.62e-4	1.67e-4	7.53e-5	3.95e-5	2.32e-5

Table 1: *Comparison of the convergence of APSS and SPSS. The values indicate the relative average precision obtained after the given number of iterations for a typical set of models.*

The results of MLS surfaces depend significantly on the weight functions. Although we tried to find best settings for each method, other weight functions could lead to other, potentially better results. In particular, we performed all our comparisons on uniform point clouds using the weight function described in section 4.1 with a constant weight radius $h_i(\mathbf{x}) = h \cdot r$ where h is an intuitive scale factor and r is the average point spacing.Moreover, our IMLS implementation corresponds to the "simple" definition given in [Kolluri 2005]:

$$f(\mathbf{x}) = \frac{\sum w_i(\mathbf{x})(\mathbf{x} - \mathbf{p}_i)^T \mathbf{n}_i}{\sum w_i(\mathbf{x})} = 0. \qquad (14)$$

In order to conveniently handle arbitrary point clouds we suggest to use $h_i(\mathbf{x}) = h \cdot r_i$ where r_i is the local point spacing that is computed as the distance to the farthest neighbor of the point $\mathbf{p}_i$ using our BSP neighborhood definition. An open source library of our APSS framework is available at: `http://graphics.ethz.ch/apss`.

7.2 Analysis of Quality and Stability

Approximation and interpolation quality
Figure 4 compares the ability of our spherical MLS to perform tight approximation and convex interpolation. For interpolation we used $\phi(x) = log(x)^4$ for both APSS and IMLS and $\phi(x) = 1/x^2$ for the two others. The values of h used for this figure are summarized in the following table (interpol./approx.):

SPSS	HPSS	IMLS	APSS
3.35 / na	na / na	1.95 / 1.95	1.95 / 3.25

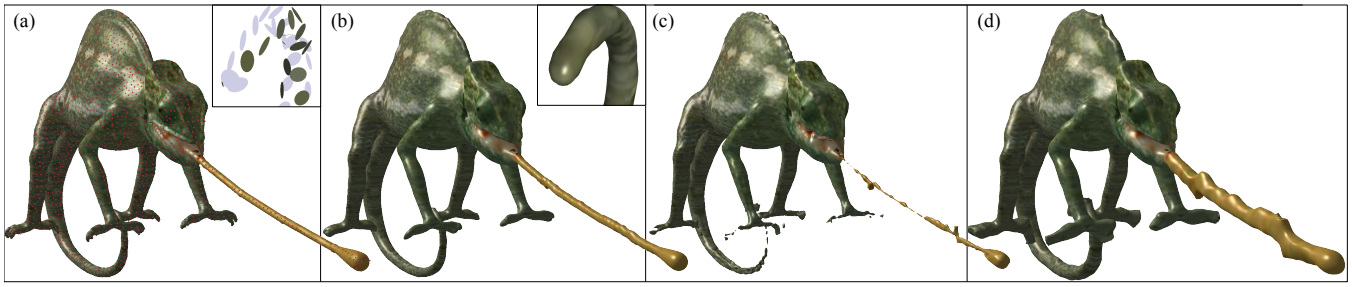

Figure 13: *(a) Initial dense chameleon model from which 4k samples have been uniformly picked (depicted with red dots). This low sampled model is reconstructed using APSS ($h = 1.9$) (b), SPSS ($h = 1.8$) (c) and IMLS ($h = 2.6$) (d).*

Stability for Low Sampling Rates

One of the central advantages of our spherical MLS is its ability to robustly handle very low sampling densities as illustrated in figure 13. The figure compares the approximation quality of a uniformly sampled 4K chameleon model for APSS, SPSS and IMLS. We can clearly see the superior quality of APSS. We computed the texture colors from the original dense model. In order to quantitatively evaluate the robustness under low sampling and the approximation quality, we iteratively created a sequence of subsampled point sets $P_0, P_1, \ldots$ by first decomposing the current point set P_i into disjunctive point pairs using a minimum weight perfect matching (see [Waschbüsch et al. 2004] for a similar scheme). A subsequent projection of the pair centers onto the actual PSS S_i yields a reduced point set P_{i+1}. The graph in figure 14 shows the relative average distance of the initial point set P_0 onto various PSS S_i as a function of the number of samples. Note that the graph is given on a logarithmic scale. The results demonstrate that the APSS is about three times more precise than the planar MLS variants. Furthermore, the planar versions break already at much higher sampling rates (crosses in the diagram), while APSS allows for stable fit at very low sampling rate. The instability of Levin's operator is mainly due to the polynomial projection which requires a relatively large and consistent neighborhood. The distribution of the error is shown for an example in figure 15.

Stability for Changes of Weights

Figure 16 compares the stability of APSS versus SPSS as a function of the size of the weight radii. We utilized a higher order genus model with very fine structures and a low sampling density. As can be seen SPSS exhibits undesired smoothing already at small radii and breaks quickly at larger sizes. In this example, APSS always delivers the expected result, and the surface stays reasonably close to original data even for large radii.

7.3 Unwanted Extra Zeros

Our spherical fitting algorithm can exhibit an increased sensitivity with respect to unwanted stationary points away from the surface. We refer to [Adamson and Alexa 2006a] for a discussion of the planar fit. With spheres, such extra zeros can specially occur close to the boundary of the surface definition domain $\mathscr{D}$ when for instance a small set of points yields a small and stable sphere that lies entirely inside $\mathscr{D}$. $\mathscr{D}$ is the union of the balls centered at the sample positions $\mathbf{p}_i$ and having the radius of the weight function ($h \cdot r_i$). We handle such spurious extra zeros by limiting the extent of $\mathscr{D}$, i.e. by both reducing the ball radii and removing regions in which fewer than a given number (e.g., 4) of samples have non-zero weights. We found that this solution is working well in practice, but admit that in-depth theoretical analysis is needed as part of future work.

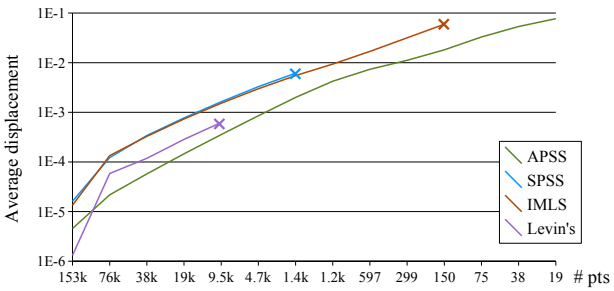

Figure 14: *Logarithmically scaled graph of the average projection error for the Chinese Dragon model.*

8 Conclusion and Future Work

We have demonstrated that the planar fit utilized in conventional PSS implementations can successfully be replaced by spherical fits. Our sphere fit MLS has numerous significant advantages and overcomes some of the intrinsic limitations of planar MLS. In particular, we obtain an increased stability for low sampling rates, a tighter fit, a better approximation quality, robustness, and for-free mean curvature estimation. Fast numerical procedures and efficient implementations allow for the processing of very complex, high-quality models and make APSS as fast as simple MLS. Extensions of the method handle discontinuities of various kinds. An apparent natural extension of the method would be to employ ellipsoids, especially for highly anisotropic objects, such as cylinders. Algebraic ellipsoidal fits, however, do not naturally degenerate to planes. In contrast, our method handles such objects correctly, as can be seen on the saddle configuration of figure 1-right.

Future work comprises applications of the representation in interactive modeling. We believe that the method is well-suited for deformations of point sampled geometry where the surface normals could be estimated directly from the gradient operator. Another important issue is the theoretical analysis of both unwanted extra zeros and the sampling requirements for our setting. Finally, we want to explore its extension to non-manifold geometry as well as realtime ray-tracing of dynamic APSS.

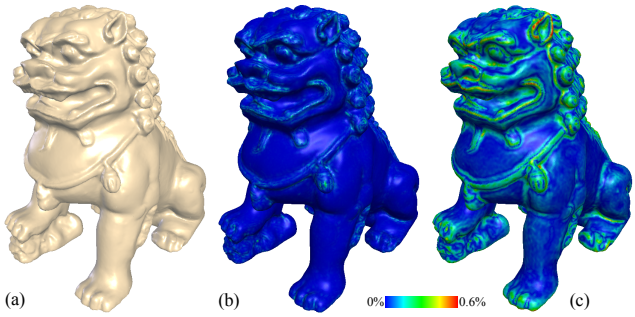

Figure 15: *Chinese Dragon model (150 K) approximated with APSS (a) and color-coded magnitude of the displacement between the input samples and a low resolution model (19 K) using APSS (b) and SPSS (c).*

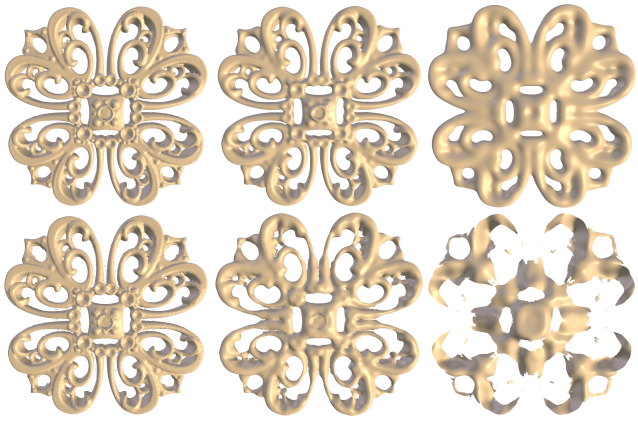

Figure 16: *A sparsely sampled point model (19 K) handled with APSS (top row) and with SPSS (botton row) using different sizes of the weight radii:* 1.6*,* 4*,* 8*.*

Acknowledgments

This work was supported in part by the ERCIM "Alain Bensoussan" Fellowship Programme. We would like to thank all reviewers for their insightful comments and discussions of the method, as well as Mario Botsch and Ronny Peikert for proof reading the paper. The models of figures 15 and 16 are provided courtesy of INRIA and SensAble by the AIM@SHAPE Shape Repository. The tree of figure 1 is provided courtesy of Vincent Forest.

References

ADAMS, B., AND DUTRÉ, P. 2003. Interactive boolean operations on surfel-bounded solids. *ACM Transactions on Graphics (SIGGRAPH 2003 Proceedings) 22*, 3, 651–656.

ADAMSON, A., AND ALEXA, M. 2003. Approximating and intersecting surfaces from points. In *Proceedings of the Eurographics Symposium on Geometry Processing 2003*, 230–239.

ADAMSON, A., AND ALEXA, M. 2004. Approximating bounded, non-orientable surfaces from points. In *Proceedings of Shape Modeling International 2004*, IEEE Computer Society.

ADAMSON, A., AND ALEXA, M. 2006. Anisotropic point set surfaces. In *Afrigraph '06: Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, ACM Press, 7–13.

ADAMSON, A., AND ALEXA, M. 2006. Point-sampled cell complexes. *ACM Transactions on Graphics (SIGGRAPH 2003 Proceedings) 25*, 3, 671–680.

ALEXA, M., AND ADAMSON, A. 2004. On normals and projection operators for surfaces defined by point sets. In *Proceedings of the Eurographics Symposium on Point-Based Graphics*, 149–156.

ALEXA, M., AND ADAMSON, A. 2006. Interpolatory point set surfaces - convexity and hermite data. *Submitted paper*.

ALEXA, M., BEHR, J., COHEN-OR, D., FLEISHMAN, S., LEVIN, D., AND SILVA, C. T. 2003. Computing and rendering point set surfaces. *IEEE Transactions on Computer Graphics and Visualization 9*, 1, 3–15.

AMENTA, N., AND KIL, Y. 2004. Defining point-set surfaces. *ACM Transactions on Graphics (SIGGRAPH 2004 Proceedings) 23*, 3, 264–270.

AMENTA, N., AND KIL, Y. 2004. The domain of a point set surface. In *Proceedings of the Eurographics Symposium on Point-Based Graphics 2004*, 139–147.

BOISSONNAT, J.-D., AND CAZALS, F. 2000. Smooth shape reconstruction via natural neighbor interpolation of distance functions. In *Proceedings of the 16th Annual Symposium on Computational Geometry*, ACM Press, 223–232.

DEY, T. K., AND SUN, J. 2005. An adaptive MLS surface for reconstruction with guarantees. In *Proceedings of the Eurographics Symposium on Geometry Processing 2005*, 43–52.

DEY, T. K., GOSWAMI, S., AND SUN, J. 2005. Extremal surface based projections converge and reconstruct with isotopy. *manuscript*.

FLEISHMAN, S., COHEN-OR, D., AND SILVA, C. T. 2005. Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics (SIGGRAPH 2005 Proceedings) 24*, 3, 544–552.

GANDER, W., GOLUB, G. H., AND STREBEL, R. 1994. Least-squares fitting of circles and ellipses. *BIT Numerical Mathematics 34*, 4, 558–578.

GUENNEBAUD, G., BARTHE, L., AND PAULIN, M. 2005. Interpolatory refinement for real-time processing of point-based geometry. *Computer Graphics Forum (Proceedings of Eurographics 2005) 24*, 3, 657–666.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1992. Surface reconstruction from unorganized points. In *Proc. of ACM SIGGRAPH '92*, ACM Press, 71–78.

KAZHDAN, M., BOLITHO, M., AND HOPPE, H. 2006. Poisson surface reconstruction. In *Proceedings of the Eurographics Symposium on Geometry Processing 2006*, 43–52.

KOLLURI, R. 2005. Provably good moving least squares. In *ACM-SIAM Symposium on Discrete Algorithms*, 1008–1018.

LEVIN, D. 2003. Mesh-independent surface interpolation. *Geometric Modeling for Scientific Visualization*, 181–187.

MITRA, N. J., NGUYEN, A., AND GUIBAS, L. 2004. Estimating surface normals in noisy point cloud data. *International Journal of Computational Geometry and Applications 14*, 4–5, 261–276.

OHTAKE, Y., BELYAEV, A., ALEXA, M., TURK, G., AND SEIDEL, H.-P. 2003. Multi-level partition of unity implicits. *ACM Transactions on Graphics (SIGGRAPH 2003 Proceedings) 22*, 3, 463–470.

PAULY, M., KEISER, R., KOBBELT, L. P., AND GROSS, M. 2003. Shape modeling with point-sampled geometry. *ACM Transactions on Graphics (SIGGRAPH 2003 Proceedings) 22*, 3.

PAULY, M., MITRA, N. J., AND GUIBAS, L. 2004. Uncertainty and variability in point cloud surface data. In *Proceedings of the Eurographics Symposium on Point-Based Graphics*, 77–84.

PRATT, V. 1987. Direct least-squares fitting of algebraic surfaces. In *Proc. of ACM SIGGRAPH '87*, ACM Press, 145–152.

SHEN, C., O'BRIEN, J. F., AND SHEWCHUK, J. R. 2004. Interpolating and approximating implicit surfaces from polygon soup. *ACM Transactions on Graphics (SIGGRAPH 2004)*, 896–904.

WALD, I., AND SEIDEL, H.-P. 2005. Interactive ray tracing of point based models. In *Proceedings of the Eurographics Symposium on Point Based Graphics 2005*.

WASCHBÜSCH, M., GROSS, M., EBERHARD, F., LAMBORAY, E., AND WÜRMLIN, S. 2004. Progressive compression of point-sampled models. In *Proceedings of the Eurographics Symposium on Point-Based Graphics 2004*, 95–102.

WICKE, M., TESCHNER, M., AND GROSS, M. 2004. CSG tree rendering of point-sampled objects. In *Proceedings of Pacific Graphics 2004*, 160–168.