Defining Point-Set Surfaces

Nina Amenta*
University of California at Davis

Yong Joo Kil†
University of California at Davis

Abstract

The MLS surface [Levin 2003], used for modeling and rendering with point clouds, was originally defined algorithmically as the output of a particular meshless construction. We give a new explicit definition in terms of the critical points of an energy function on lines determined by a vector field. This definition reveals connections to research in computer vision and computational topology.

Variants of the MLS surface can be created by varying the vector field and the energy function. As an example, we define a similar surface determined by a cloud of surfels (points equipped with normals), rather than points.

We also observe that some procedures described in the literature to take points in space onto the MLS surface fail to do so, and we describe a simple iterative procedure which does.

1 Introduction

Because of improved technologies for capturing points from the surfaces of real objects and because improvements in graphics hardware now allow us to handle large numbers of primitives, modeling surfaces with clouds of points is becoming feasible. This is interesting, since constructing meshes and maintaining them through deformations requires a lot of computation. It is useful to be able to define a two-dimensional surface implied by a point cloud. Such *point-set surfaces* are used for interpolation, shading, meshing and so on.

David Levin's MLS surface [Levin 2003] has proved to be a very useful example of a point-set surface. Levin defined the MLS surface as the stationary points of a map f, so that x belongs to the MLS surface exactly when $f(x) = x$. This definition is useful but it does not give much insight into the properties of the surface.

In Section 2 we give a more explicit definition of the MLS surface, based on an energy function e and an vector field n. Changing e and n produce other similar surfaces; we call them all, including the MLS surface, *extremal surfaces*.

As we discuss in Section 4, extremal surfaces are not new. The name is adopted from Medioni and co-authors [2000], who used them for surface extraction from very noisy point clouds, among other applications in computer vision. Their definition has to be extended slightly in order to include the MLS surface, but the idea is essentially the same. Edelsbrunner and Harer [to appear] have recently described *Jacobi surfaces*, which are also closely related and which have stronger mathematical properties.

Adamson and Alexa [2003a] describe an implicit surface which is a useful "relative" of the MLS surface. This raises the question of the relationship of extremal surfaces and implicit surfaces. As we discuss in Section 5, there is an implicit surface containing every extremal surface, including the MLS surface. This can be quite useful, particularly for defining normals precisely.

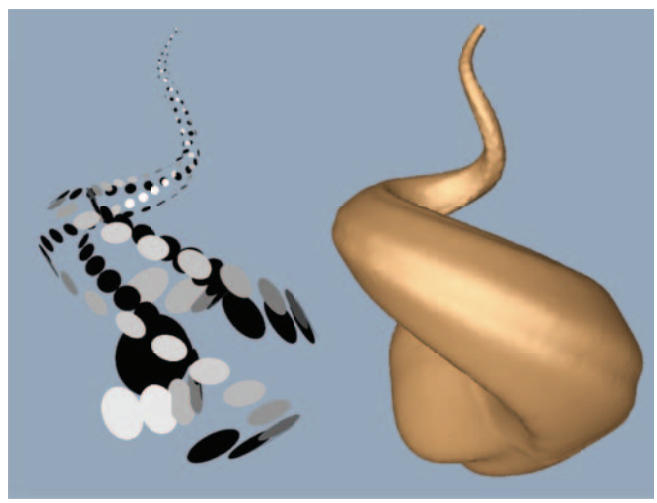

Figure 1: An example of an extremal point-set surface which takes surfels, rather than points as input. The sparse and non-uniformly distributed set of weighted surfels on the left implies the surface on the right.

As an example of another extremal surface construction, we describe in Section 7 a point-set surface for *surfels*, input points equipped with normals. Normals are available when converting a model from a mesh or implicit surface to a point cloud, and they are generally computed as part of the process of cleaning up point clouds produced by laser range scanners or other 3D capture technologies. Figure 1 shows an example of our point-set surface for surfels.

MLS surfaces have been used widely in the last few years. The seminal graphics paper [Alexa et al. 2003] using the MLS surface for point-set modeling and rendering was followed by work on progressive MLS surfaces [Fleishman et al. 2003], ray-tracing [Adamson and Alexa 2003b], and surface reconstruction [Xie et al. 2003; Mederos et al. 2003]. The MLS surface is used in several features of PointShop 3D, an excellent open-source point-cloud manipulation tool [Zwicker et al. 2002; Pauly et al. 2003]. We have used PointShop as our implementation platform.

These papers describe two slightly different procedures for taking a point x in the neighborhood of the point cloud onto the MLS surface, one which first appeared in an earlier, widely circulated manuscript version of Levin's paper, and a more efficient "linear" version used in PointShop. Neither procedure actually produces a point of the MLS surface. We give a simple procedure which does, together with a short proof, in Section 6, and we discuss the theoretical problems with the earlier procedures in Appendix A. The final version of Levin's paper on the MLS surface [Levin 2003] (currently available on his Web site) contains a new projection procedure, different from ours, which also produces points on the sur-

*Supported by NSF ACI-0325934, CCR-0098169, and CCR-0093378.
e-mail: amenta@cs.ucdavis.edu

†Supported by NSF CCR-0093378.
e-mail:kil@cs.ucdavis.edu

face.

2 Surface definition

We begin with the now-standard definition of the MLS surface, given in the early manuscript of Levin's paper and used in a variety of contexts as mentioned above. The MLS surface for a point cloud $P \subseteq \mathbb{R}^3$ is defined as the set of stationary points of a certain function $f : \mathbb{R}^3 \to \mathbb{R}^3$. An optional polynomial fitting step, which we omit here, can be applied after the map f.

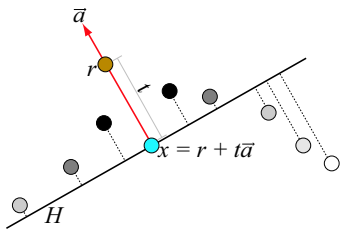

Figure 2: The MLS energy function $e_{MLS}(\vec{a},t)$ sums up the weighted distances from the fixed input points in P to the plane with normal $\vec{a}$ through the point $x = r + \vec{a}t$. The weight on an input point $p_i \in P$, denoted here by its shade of grey, is a function of the distance from p_i to x.

Given an input point cloud P and a point r in a neighborhood of P, the energy of the plane with normal $\vec{a}$ passing through the point $x = r + t\vec{a}$, where t is the distance from r to the plane, is defined to be:

$$e_{MLS}(\vec{a},t) = \sum_{p_i \in P} (\langle \vec{a}, p_i \rangle - \langle \vec{a}, r + t\vec{a} \rangle)^2 \, \theta(p + t\vec{a}, p_i)$$

where the weighting function θ is any monotonic function, usually a Gaussian:

$$\theta(x, p_i) = e^{\frac{-d^2(x-p_i)}{h^2}}$$

Here h is a constant scale factor, and $d()$ is the usual Euclidean distance between points. See Figure 2. The energy measures the quality of the fit of the plane to P, where $p_i \in P$ is weighted by its distance from $x = r + t\vec{a}$.

The local minima of this energy function, over $\mathbb{S}^2 \times \mathbb{R}$ ($\mathbb{S}^2$ is the space of directions, the ordinary two-sphere), occur at a discrete set of inputs $(\vec{a},t)$, each corresponding to a point $x = r + t\vec{a}$. Of these, $f(r)$ is defined to be the x nearest to r. The stationary points of this map f form the MLS surface.

We can get some additional insight into this energy function by restating it using different notation. First, we write it as a function of $\vec{a}$ and the point $x = r + t\vec{a}$, rather than as a function of $\vec{a}$ and t. Second, we notice that the plane and the weights determined by the parameters $(x,\vec{a})$ are the same as those determined by the parameters $(x,-\vec{a})$, so we can write e_{MLS} as a function of x and an *unoriented* direction vector a. This gives us:

$$e_{MLS}(x,a) = \sum_{p_i \in P} (\langle a, p_i \rangle - \langle a, x \rangle)^2 \, \theta(x, p_i)$$

(although the inner product is not defined for unoriented direction vectors, we use this notation since we can evaluate the function using either $\vec{a}$ or $-\vec{a}$ and get the same result). In this new form, the domain of the function is now $\mathbb{R}^3 \times \mathbb{P}^2$, where $\mathbb{P}^2$ (the projective two-sphere) is the space of unoriented directions. The new notation makes it clear that $e_{MLS}(x,a)$ is independent of r, which will help us find a non-algorithmic characterization of the points of the MLS surface in the following section.

The procedure for computing $f(r)$, described above, minimizes $e_{MLS}(\vec{a},t)$ over the three-dimensional domain $\mathbb{S}^2 \times \mathbb{R}$. Therefore using the new notation we do not minimize over all of the five-dimensional domain $\mathbb{R}^3 \times \mathbb{P}^2$, but over a three-dimensional subset: the set J_r of values (x,a) such that $x = r + t\vec{a}$ for some $t \in \mathbb{R}$ and orientation $\vec{a}$ of a. This means that in J_r, every point $x \in \mathbb{R}^3$ other than r is paired with the direction a of the line through x and r; the singular point r is paired with all directions a. Different values of r produce different values of $f(r)$ because each choice of r implies a different domain J_r over which e_{MLS} is minimized.

3 Explicit definition and generalization

Now we want to give an explicit version of the MLS surface definition. We begin by defining an (unoriented) vector field:

$$n(x) = \text{argmin}_a \, e_{MLS}(x,a) \tag{1}$$

This is the normal to the plane *through* $x \in \mathbb{R}^3$ which is a local best-fit to the point cloud P. Since fixing x fixes the weights, the energy function is a quadratic function of a and the minimal direction is usually unique. It can be found efficiently as the smallest eigenvalue of a three-by-three matrix [Alexa et al. 2003]. Where there are two or three smallest eigenvalues, n is not well-defined. The sets of points with multiple smallest eigenvalues form surfaces which divide space into regions, within each of which n is a smooth function of x.

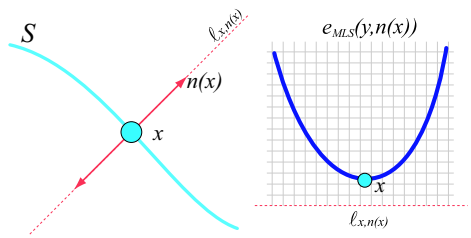

Figure 3: To see if a point x belongs to the MLS surface, we consider e_{MLS} on the line $\ell_{x,n(x)}$. Keeping $n(x)$ fixed, we vary y along the line. If $y = x$ is a local minimum of $e_{MLS}(y,n(x))$, then x belongs to the MLS surface. Using different functions for $n(x)$ and $e(x,a)$ gives variants of the construction, which we call *extremal surfaces*.

Now we give an explicit characterization of the MLS surface; in effect, we describe how to recognize whether a point $x \in \mathbb{R}^3$ belongs to the MLS surface. This characterization is illustrated in Figure 3. Let $\ell_{x,n(x)}$ be the line through x with direction $n(x)$. We adopt the notation arglocalmin$_y$ to refer to the set of inputs y producing local minima of a function of variable y.

Claim 1 *The MLS surface consists of the points x for which $n(x)$ is well-defined, and for which*

$$x \in arglocalmin_{y \in \ell_{x,n(x)}} e_{MLS}(y, n(x))$$

Proof : First we argue that every x on the MLS surface has this property. Such a point corresponds to a pair (x,a) which is a local minimum in its own set J_x. The set $A = \{(x,a)\}$, where x is fixed and a ranges over all possible directions, is a subset of J_x, so $(x,n(x))$ is a local minimum in A. Since $n(x)$ is well defined, $n(x)$ is the unique global minimum in A and we have $a = n(x)$. The set of pairs $L = \{(y,n(x)) \mid y \in \ell_{x,n(x)}\}$ is also a subset of J_x, so $(x,n(x))$ is also a local minimum in L.

Now we want to show that any x which has the property in the Claim belongs to the MLS surface. We need therefore to show that $(x,n(x))$ is a local minimum of e_{MLS} in the set J_x.

Consider any direction $m \neq n(x)$. Since $n(x)$ is defined as the direction producing the unique minimum over all pairs (x,a), we have $e_{MLS}(x,m) > e_{MLS}(x,n(x))$. The function e_{MLS} is continuous, so there is some distance $\varepsilon(m)$ such that for all $y \in \ell_{x,m}$ with $d(x,y) < \varepsilon(m)$, we have $e_{MLS}(y,m) > e_{MLS}(x,n(x))$. Also there is some $\varepsilon(n(x))$ such that for all $y \in \ell_{x,n(x)}$ with $d(x,y) < \varepsilon(n(x))$, $e_{MLS}(y,n(x)) > e_{MLS}(x,n(x))$. Let ε be the minimum of $\varepsilon(a)$ over all directions in $a \in \mathbb{P}^2$. Then $(x,n(x))$ is a local minimum in the subset of J_x consisting of pairs (y,a) with $d(x,y) < \varepsilon$. □

We can generalize the MLS construction by considering alternatives for the two functions n and e_{MLS}. We can use any function $n(x)$ to assign directions to points in space, and any function $e(x,a)$ to specify the energy of elements of $\mathbb{R}^3 \times \mathbb{P}^2$. There is no reason why the definition of n has to be related to the definition of e, as it is for the MLS surface. We define an *extremal surface* as follows.

Definition 1 *For any functions* $n : \mathbb{R}^3 \to \mathbb{P}^2$ *and* $e : \mathbb{R}^3 \times \mathbb{P}^2 \to \mathbb{R}$, *let*

$$S = \{x \mid x \in arglocalmin_{y \in \ell_{x,n(x)}} e(y,n(x))\}$$

be the extremal surface of n and e.

4 Extremal surface literature

Not surprisingly, the idea of extremal surfaces is not new. Guy and Medioni [1997] and Medioni, Lee and Tang [2000] defined extremal surfaces, using functions $n : \mathbb{R}^3 \to \mathbb{P}^2$ and $s : \mathbb{R}^3 \to \mathbb{R}$, to define the set $\{x \mid x \in \text{arglocalmin}_{y \in \ell_{x,n(x)}} s(y)\}$. Our definition is a a little more general than theirs in that their function $s(x) = e(x,a)$ does not require the parameter a. In a series of papers, they used extremal surfaces for (among other things) surface reconstruction from very noisy point clouds. Their functions n and s are different from the MLS energy function, and require completely different computational techniques. They represent n and s simultaneously with a tensor, and use tensor operations to smooth them. This *tensor voting* is performed on a voxel grid. The extremal surface is then extracted from the grid with the marching cubes algorithm. Most of their work focuses on the difficult problem of designing of good tensor functions.

Edelsbrunner and Harer [to appear] define *Jacobi surfaces* in $\mathbb{R}^d$. To keep this discussion simple, we give their definition for the special case of two-surfaces in $\mathbb{R}^3$. The input is three Morse functions $f_1(x), f_2(x), f_3(x)$ (intuitively, a Morse function is one whose iso-surfaces are generic; it is everywhere twice-differentiable, its Hessian matrix is everywhere non-singular, and no two critical points have the same function value). Jacobi surfaces are symmetric with respect to the order of the input functions, so that for instance if we exchange f_1 and f_3, we get the same Jacobi surface.

The intersections of the level sets of f_1 and f_2 divide $\mathbb{R}^3$ into a family of curves. The Jacobi surface S is defined as the set of critical points of f_3 over each of these curves. Every point $x \in \mathbb{R}^3$ belongs to one such curve, and we let $n(x)$ be the tangent direction. Every critical point q of f_3 on the curve containing x is a critical point of f_3 on the tangent line $\ell_{x,n(x)}$ as well, so this is similar to an extremal surface with $f_3(x) = s(x)$. The main difference is that all critical points, rather than just minima, are taken. Another difference is that points at which n is undefined (because the intersection of the level sets consists of a single point instead of a curve) are included in the Jacobi surface (this is related to the symmetry of the definition).

With these points included as part of the surface, it seems likely that these singularities in the vector field n might cause non-manifold singularities in the surface S, for instance points at which multiple sheets of surface come together. Edelsbrunner and Harer show, however, that a Jacobi two-surface in $\mathbb{R}^3$ is generically a manifold. This does not extend to higher dimensions; for instance a Jacobi 3-surface in $\mathbb{R}^4$ can be generically non-manifold, indeed at the singular points at which n would be undefined. They prove that Jacobi k-surfaces in $\mathbb{R}^d$, for $d > 2k-2$, are manifolds, and they give a robust algorithm for extracting Jacobi surfaces from functions given on a tetrahedral mesh.

5 Implicit and extremal surfaces

Adamson and Alexa [2003a] defined an implicit surface which they used for ray-tracing instead of the MLS surface. Their surface has the form

$$g(x) = \vec{n}(x) \cdot (a(x) - x) = 0$$

where $\vec{n}(x)$ is an *oriented* vector field and $a(x)$ is the center of mass of the input point cloud P as weighted by x.

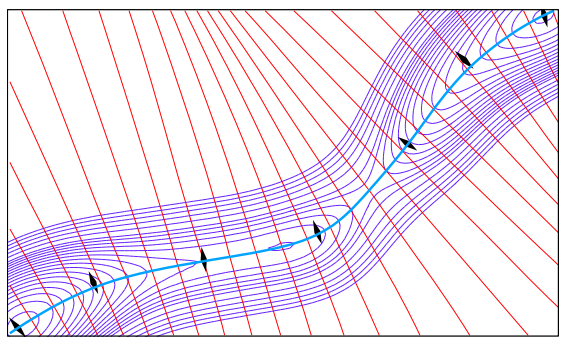

Figure 4: Streamlines (red) of a vector field $n(x)$, and iso-contours (blue) of an energy function $s(x)$. The heavy blue line is the extremal surface determined by n and s, running neatly along the "valley" in the energy landscape and passing through the minima of s. The streamlines of n and the iso-contours of s are tangent at the surface points. Here n and s were computed using the point-set surface for surfels introduced in Section 7; the input surfels are shown as black diamonds, with the long diagonal pointed in the direction of the surfel normal.

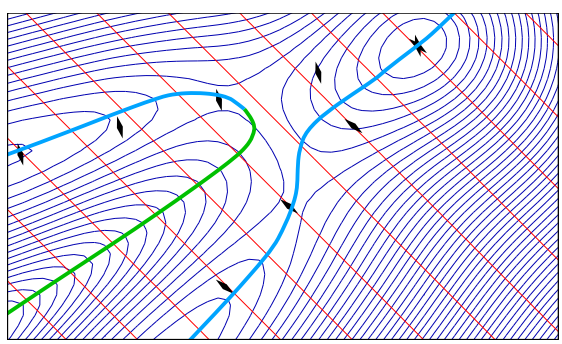

Figure 5: The red streamlines indicate a constant vector field $\vec{n}$. The blue iso-contours show an energy function s again determined by the set of input surfels (black diamonds), here meeting at a sharp corner. There are two valleys in the energy landscape meeting to form a third valley. The implicit surface $g(x) = \vec{n}(x) \cdot \nabla s(x) = 0$ includes both minima in the extremal surface definition (heavy blue curve) and also maxima (green curve). The extremal surface (heavy blue curves) appears to have a junction but is actually composed of two manifold components. Using the best-fitting plane to determine the vector field $\vec{n}$, as defined for the MLS surface in Equation 1, produces a similar structure near the sharp corner, but the somewhat larger picture is complicated by singularities in the vector field.

They prove that $g(x)$ is a smooth function on any domain on which $\vec{n}$ is well-defined everywhere, and therefore that the surface

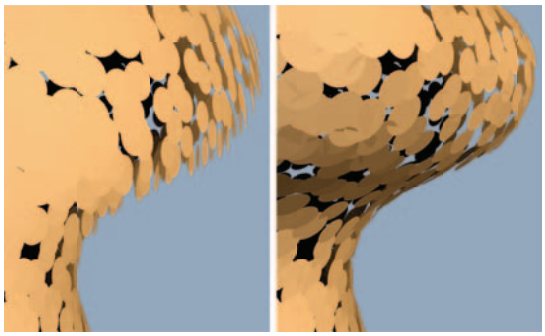

Figure 6: Surfels distributed on a part of the extremal surface of Figure 8 where the optimal direction $n(x)$ at a surface point x is very different from the surface normal at x. On the left, the surfels are oriented by $n(x)$, and on the right by the surface normal.

$g(x) = 0$ is a manifold on the domain, assuming it avoids critical points of g. They argue that it does, generically (meaning that if $g(x) = 0$ does contain a critical point of g, a small perturbation of the input fixes the problem).

Just considering a domain within which the vector field $\vec{n}$ is well-defined neatly avoids the singularities found in higher dimensions with Jacobi surfaces. Medioni et al. pointed out [2000] that in a similar context there is an implicit surface associated with any extremal surface. We consider a domain $U \subseteq \mathbb{R}^3$ within which not only is n always defined, but it also allows a consistent orientation, meaning that adopting this orientation produces a smooth oriented vector field $\vec{n}$. Since a point of the extremal surface is a critical point of s on the line $\ell_{x,n(x)}$, the directional derivative of s in direction $n(x)$ has to be zero at x, and we can define the implicit surface

$$g(x) = \vec{n}(x) \cdot \nabla s(x) = 0$$

That is, at a point x of the extremal surface, $n(x)$ is perpendicular to the gradient of s, and tangent to the iso-surface $s(y) = s(x)$, for $y \in \mathbb{R}^3$. This is illustrated in Figure 4. Although it is tempting to assume that the orientation of n is unnecessary since the zero-set of $g^2(x)$ would be the same with either orientation, the zero-set of $g^2(x)$ consists entirely of critical points and so may not be a manifold, particularly where n does not admit a consistent orientation.

Using our more general formulation of extremal surfaces, which includes the MLS surface, we can define the surface

$$g(x) = \vec{n}(x) \cdot \nabla_y \, e(y, \vec{n}(x))|_x) = 0$$

where $\nabla_y e(y, \vec{n}(x))|_x$ is the gradient of e as a function of y, keeping $n(x)$ fixed, and then evaluated at x.

Any of these iso-surface are manifolds over the domain U, so long as they avoid critical points of g, which they do generically. Notice that a point x on one of the iso-surfaces might be a maximum on $\ell_{x,n(x)}$ rather than a minimum. Taking only the minima, as in the MLS surface definition, might further eliminate parts of each surface, as in Figure 5, where the part which is eliminated indeed should not be included in the point-set surface.

We summarize as follows.

Observation 2 *The MLS surface, within a domain where n is well-defined and admits a consistent orientation, is, generically, a subset of a manifold.*

This seems to contradict our experience, for instance when the input points come from surfaces with sharp corners, in the vicinity of which two sheets of the MLS surface seem to collapse into one. Looking Figure 5, however, we see that that just before the apparent juncture one of the sheets ends at a boundary where the critical points on $\ell_{x,n(x)}$ switch from minima to maxima.

Describing an extremal surface as a subset of an iso-surface gives an analytic expression for its normal. In the case of the MLS surface this formula includes the derivatives of the weights and is rather complicated. Note that in general the surface normal at a point x is different from the direction $n(x)$; see Figure 6.

6 Finding surface points

At least two different procedures for taking arbitrary points in $\mathbb{R}^3$ to the MLS surface have been proposed. The projection procedure given by Levin [Levin 2003], based on the definition above, was used in the seminal paper [Alexa et al. 2003]. A somewhat different procedure, designed so as to avoid numerical optimization, is used in PointShop [Zwicker et al. 2002]. In this section we give another simple procedure for generating surface points.

We also give a simple proof of correctness. Neither of the earlier procedures were rigorously proved correct, and interestingly upon very close examination neither of them actually produces a point on the MLS surface, although in practice the error is negligible. We discuss the technical difficulties with these procedures in Appendix A.

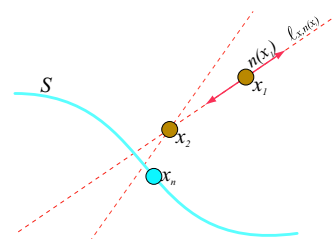

Figure 7: Diagram of the process for taking a point to an extremal surface. Point x_1 moves to a local minimum on the line $\ell_{x_1,n(x_1)}$, represented by the point at which the dashed lines meet. This becomes x_2. When the process converges, x_n lies on S.

Our process for taking a point onto the extremal surface S implied by n and e is illustrated in Figure 7. At each iteration, we find $n(x_i)$ and consider the line $\ell_{x_i,n(x_i)}$. We search for a nearby local minimum of $e(y, n(x_i))$ over the set $y \in \ell_{x_i,n(x_i)}$. The nearest local minimum becomes x_{i+1}. Notice that as long as resetting $n(x_{i+1})$ at each new point does not increase e, the energy decreases at every step so that this process is likely to converge. The energy does indeed decrease for the MLS function and also for any function $e(x,a) = s(x)$ which does not depend on the direction parameter.

Claim 3 *If the procedure above converges, it produces a point of S.*

Proof: At convergence, repeating the procedure for x_n produces the same point x_n. Since x_n is a local minimum of $e(y, n(x_n))$ within $y \in \ell_{x_n,n(x_n)}$, this means that $x_n \in S$ according to Definition 1. □

7 Point-set surface for surfels

Now we define a point-set surface which takes as input a set of surfels rather than a point cloud. Normals are often available, and using surfels rather than points makes the surface more robust in the face of both undersampling and of irregularities in the distribution of samples; see Figure 8. Our input surfels are represented as point-direction pairs (p_i, a_i). Following the intuition that $n(x)$

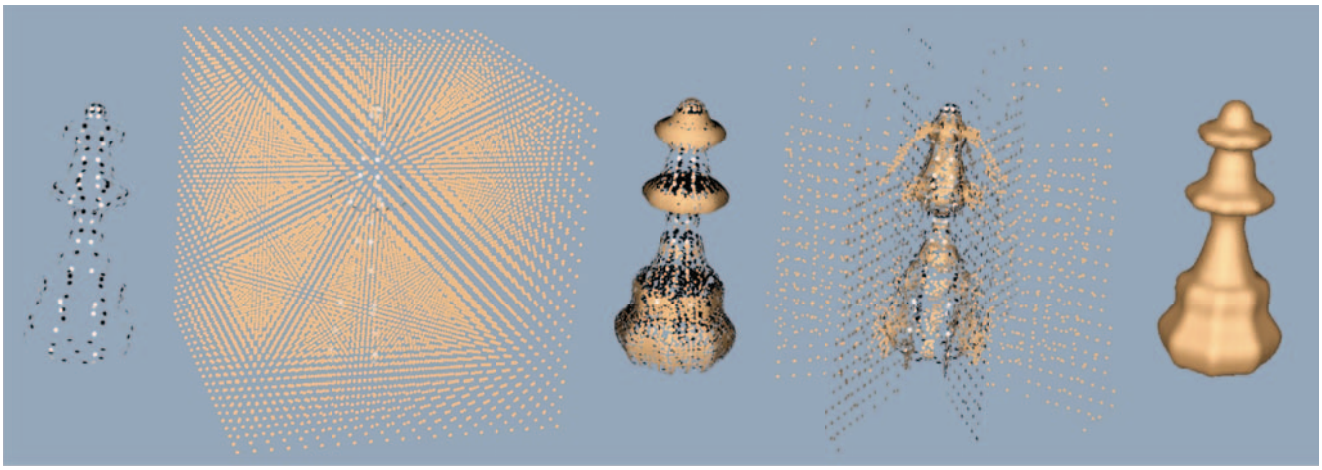

Figure 8: From left to right, a sparse set of surfels defining a chess-piece. Next, we take a 3D grid of points onto the surface using our point-set surface (with $c = 0$). In the center, we find that the points do indeed go to a two-dimensional surface. The MLS surface, as implemented in PointShop, has trouble on this example. Without using the normal information, the very sparse, non-uniform distribution of points makes the MLS energy function give very good scores to the planes through the vertical rows of points; we show the grid points as projected by MLS. Finally, at the far right, the complete surface produced by our point-set surface.

should follow the surface normal at the nearest surface point to x, we compute n as a weighted combination of the nearby surfel normals. If we have oriented normals, we can take a vector average.

$$n(x) = \sum_i a_i \, \theta_N(x, p_i)$$

where

$$\theta_N(x, p_i) = \frac{e^{-d^2(x,p_i)/h^2}}{\sum_j e^{-d^2(x,p_j)/h^2}}$$

is a *normalized* Gaussian weighting function.

If the surfels have unoriented normals, we instead use the principal component of the normal vectors, again weighted by θ; this is the direction of the line through x which is the weighted least-squares best fit to the vectors, and it can be computed as the eigenvector of largest eigenvalue of the 3×3 covariance matrix B where

$$b_{jk} = \sum_i \theta_N(x, p_i) \, a_{i,j} \, a_{i,k} \;\; \forall \, j,k = \{1,2,3\}$$

and the (x,y,z) coordinates of vector a_i are $(a_{i,1}, a_{i,2}, a_{i,3})$. This is not quite as efficient as the vector average, but in either case computing $n(x)$ is faster than minimizing e.

Our intuition is that e is an estimate of unsigned squared distance function, based on surfel position and normal. We define the distance of x from a surfel as a Mahalanobis distance (like Euclidean distance but with elliptical rather than spherical unit ball), where distance in direction a_i is emphasized over directions perpendicular to a_i.

$$d_M(p_i, a_i, x) = \langle (x - p_i), a_i \rangle^2 + c \, || \, (x - p_i) \; - \; \langle (x - p_i), a_i \rangle \, a_i \, ||^2$$

With the scaling factor $c = 1$ we have the Euclidean distance from x to p_i, and with $c = 0$ we have the squared distance from x to the plane through p_i normal to a_i. Figure 9 shows the effect of different choices of the parameter c. Finally we define

$$e(x, a) = e(x) = \sum_i d_M(p_i, a_i, x) \theta_N(x, p_i)$$

We implemented the procedure for taking points in space to this extremal surface as part of a plug-in for PointShop. To find the minimum of e on the line $\ell_{x,n(x)}$, we define $q = x + \tau n(x)$, and minimize $e(q)$ as a function of τ. We used an implementation of Brent's method for this one-dimensional non-linear optimization from *Numerical Recipes in C* [1992], similar to Alexa et.al. [2003]. For efficiency, we used PointShop's kd-tree to find nearby surfels, and used contributions only from the nearest 30.

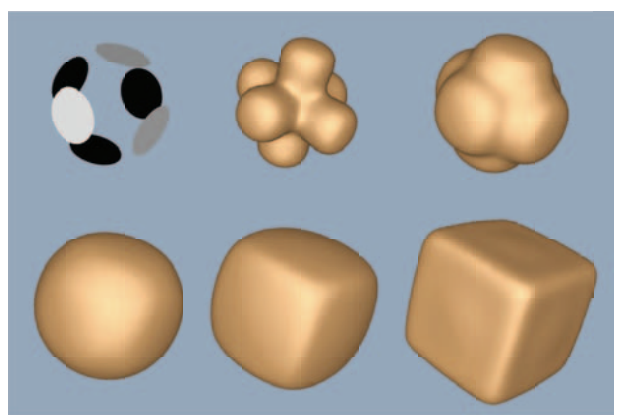

Figure 9: The point-set surface produced by six surfels. The constant c in the energy function is one at the upper center, then halved for each successive image, and finally zero at the lower right.

Using θ_N instead of θ is important since with the simple Gaussian the energy would be effectively zero far from the surface. When the Gaussian weight on every point is numerically zero we cannot compute θ_N. In that case we let θ_N be one for the nearest surfel and zero for all others, which is nearly correct.

Our implementation allows input surfels to have variable weights, so that we can vary their distribution on the surface, as in Figure 1. This is implemented by storing a separate weight h_i with each surfel.

$$\theta_N(x, p_i) = \frac{e^{-d^2(x,p_i)/h_i^2}}{\sum_j e^{-d^2(x,p_j)/h_j^2}}$$

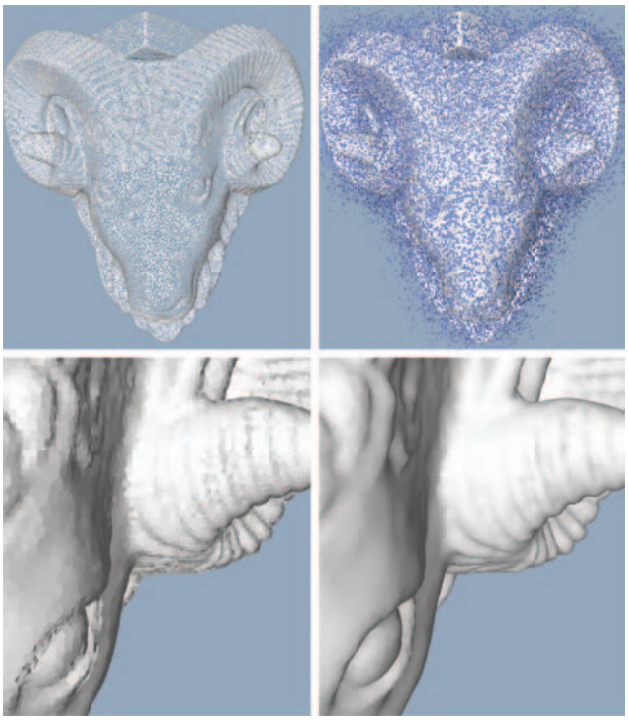

Figure 10: Using our surfel point-set surface definition to fill in a set of samples for rendering. The input data is the vertex set of a polygonal model. We produced more surfels by generating a cloud of new points and taking the new (blue) points onto the surface implied by the input (white) surfel cloud. Below, PointShop's renderings of the original and the filled-in set of surfels.

A dense, uniform point sample from a natural surface inherently contains good normal information, so using surfels is not really helpful with such data. To check the efficiency of our implementation, however, we used it to fill in the surface of a densely sampled model, a typical application for point-set surfaces. The results are shown in Figure 10. We found that even though we use a non-linear optimization, we are less than twice as slow as the MLS projection heuristic implemented in the ScanTools plugin in PointShop. We took a cloud of 77,428 surfels onto the extremal surface implied by an input cloud, also consisting of of 77,428 surfels, in 16 seconds, while PointShop's procedure (also without using the second polynomial approximation step) required 9 seconds.

8 Discussion

Many questions about the MLS surface and other extremal surfaces remain.

In practice, the main question is which extremal surfaces, if any, are good choices for solving specific modeling problems. For instance, surface reconstruction from noisy laser range data seems to require a surface definition which incorporates information such as normals, scan direction, and a good error model, with less emphasis on performance. Is there a good extremal surface solution, and what computational methods are appropriate in this case?

There does not yet seem to be an ideal way to find the vector field n induced by an input point cloud. The vector field n produced by the MLS energy function has singularities quite close to the point cloud, which makes it difficult to work with. One possibility might be to assign approximate normals at the input points using MLS, and then use the resulting surfels as input to our point-set surface definition, but there may be simpler or more robust approaches.

There are many issues to be resolved with respect to procedures for taking points in space to an extremal surface. Despite the theoretical problems with both Levin's original procedure and with the heuristic used in PointShop, both methods have advantages that ours does not: the projection idea is very elegant, while the heuristic is efficient. The new projection procedure in the final version of Levin's paper [Levin 2003] could be applied with any extremal surface, and may be useful in practice.

While the iso-surface definition of an extremal surface gives an expression for the normal, it is often complicated, since it includes the derivatives of the weight functions. Are there extremal surfaces for which the normals have a particularly simple form? Or is there some way to use a simpler implicit function, such as that proposed by Adamson and Alexa, without including parts such as the green curve in Figure 5?

Finally, it would be good to prove connections between some extremal surface and the real surfaces from which point samples are taken. Under what sampling conditions can we guarantee that the extremal surface defined by a sample point cloud is everywhere close to the original surface? Under what conditions can we guarantee that there is an isotropy between them?

9 Acknowledgments

We thank David Levin for corresponding with us on these questions. We thank an anonymous referee for suggesting the second half of the proof of Claim 1, and Peter Schröder and an anonymous referee for suggestions on the presentation. We are grateful to the team at ETH Zurich who developed and published PointShop 3D. We thank Holly Rushmeier and the IBM TJ Watson Research Center for the use of the ram model.

References

ADAMSON, A., AND ALEXA, M. 2003. Approximating and intersecting surfaces from points. In *Proceedings of EG Symposium on Geometry Processing 2003*, 245–254.

ADAMSON, A., AND ALEXA, M. 2003. Ray tracing point set surfaces. In *Proceedings of Shape Modeling International 2003*, 272–279.

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 Visualization and Computer Graphics 9*, 1, 3–15. An earlier version appeared in IEEE Visualization 2001.

CARR, J., BEATSON, R., CHERRIE, J., MITCHELL, T., FRIGHT, W., MCCALLUM, B., AND EVANS, T. 2001. Reconstruction and representation of 3d objects with radial basis functions. *ACM SIGGRAPH '01*, 67–76.

CURLESS, B., AND LEVOY, M. 1996. A volumetric method for building complex models from range images. *ACM SIGGRAPH '96*, 303–312.

EDELSBRUNNER, H., AND HARER, J. to appear. Jacobi sets of multiple morse functions. In *Foundations of Computational Mathematics*, F. Cucker, Ed. Cambridge University Press.

FLEISHMAN, S., COHEN-OR, D., ALEXA, M., AND SILVA, C. T. 2003. Progressive point set surfaces. *ACM Transactions on Graphics 22*, 4, 997–1011.

GUO, X., AND QUIN, H. 2003. Dynamic sculpting and deformation of point set surfaces. In *11th Pacific Conference on Computer Graphics and Applications (PG'03)*, 123–130.

GUY, G., AND MEDIONI, G. 1997. Inference of surfaces, 3d curves and junctions from sparse, noisy, 3d data. *IEEE Trans. on Pattern Analysis and Machine Intelligence 19*, 11, 1265–1277.

LEVIN, D. 2003. Mesh-independent surface interpolation. In *Geometric Modeling for Scientific Visualization*, G. Brunnett, B. Hamann, K. Mueller, and L. Linsen, Eds. Springer-Verlag.

MEDEROS, B., VELHO, L., AND DE FIGUEIREDO, L. H. 2003. Moving least squares multiresolution surface approximation. In *Proceedings of SIBGRAPI 2003 - XVI Brazilian Symposium on Computer Graphics and Image Processing*.

MEDIONI, G., LEE, M.-S., AND TANG, C.-K. 2000. *A Computational Framework for Segmentation and Grouping*. Elsevier.

PAULY, M., KEISER, R., KOBBELT, L., AND GROSS, M. 2003. Shape modeling with point-sampled geometry. *ACM SIGGRAPH 2003*, 641–650.

PAULY, M. 2003. *Point Primitives for Interactive Modeling and Processing of 3D Geometry*. PhD thesis, ETH Zurich.

PRESS, W., FLANNERY, B., TEUKOLSKY, S., AND VETTERLING, W. 1992. *Numerical Recipes in C*, 2nd ed. Cambridge University Press.

XIE, H., WANG, J., HUA, J., QUIN, H., AND KAUFMAN, A. 2003. Piecewise c^1 continuous surface reconstruction of noisy point clouds via local implicit quadric regression. *IEEE Visualization 2003*, 91–98.

ZWICKER, M., PAULY, M., KNOLL, O., AND GROSS, M. 2002. Pointshop 3d: An interactive system for point-based surface editing. *ACM SIGGRAPH 2002*, 322–329.

A Appendix - Projection procedures

Interestingly, two procedures described in the literature for producing points on the MLS surface both output points very near, but not actually on, the surface, for almost all inputs. While these procedures obviously work well in practice, it seems important to recognize these subtleties in the effort to develop a good theory of MLS and other extremal surfaces. In this appendix we discuss the technical problems with these procedures. As a concrete way of showing what goes wrong, we include a Mathematica file as supplementary material with this paper, giving an example of applying both procedures to a generic point r in space and observing that indeed the resulting point x does not meet the definition of a point on the MLS surface.

Since the stationary points of Levin's projection function f, described in Section 2 are defined to be the points of the MLS surface, we know that if we iterate the procedure and find that it converges to a point x, then $x \in S$. An early argument of Levin's (in the manuscript version of his paper) suggested that one iteration suffices, that is that $f(r) \in S$ for any r. While this seems very plausible, in fact there is a subtle problem, specifically in the following proposition, which turns out to be false: that for any y on the line segment connecting r and $x = f(r)$, we also have $f(y) = x$. The actual situation is shown in Figure 11.

Recall that in the projection procedure we find x by finding a pair (x,a) which is a local minimum of e over the set J_r. The statement that $f(r) = x$ seems plausible because if we want to compute $f(y)$ we consider the set J_y, and (x,a) certainly belongs to J_y and of course the energy value $e(x,a)$ is the same in both J_y and J_r. The problem is that $e(x,a)$ is not necessarily a *minimum* in J_y; the neighborhood of (x,a) in J_y is different from the neighborhood of (x,a) in J_r, and elements (x',a') in the neighborhood of (x,a) in J_y, not belonging to J_r, may (and generally do) have a lower values of the energy function e. We cannot claim, "$e(x,a)$ is a minimum in J_r, (x,a) belongs to J_y, therefore $e(x,a)$ is a minimum in J_y."

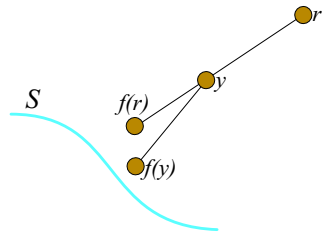

Figure 11: Function f is described in Section 2. We note here that it is generally the case that $f(y) \neq f(r)$ for a point y on the line segment connecting r with $f(r)$.

In PointShop, Pauly [Pauly 2003] describes different iterative procedure, illustrated in Figure 12. In this procedure, we begin with an estimated projection point x_1 for r. We use x_1 to assign weights $\theta(x, p_i)$ to the points p_i of the input point cloud P. We then find the total-least-squares best-fit plane H_{x_1} to the weighted set P; notice that although H_{x_1} passes through the centroid c of the weighted point cloud P, in general it does not pass through x_1. We project r onto H_{x_1}, giving a new estimate of the projection x_2, and we iterate. If this procedure converges (which it does very reliably), we output the resulting point x. This procedure has the distinct advantage of requiring no non-linear optimization.

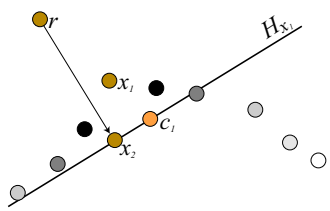

Figure 12: Pauly's procedure: the points of P are weighted by a guess x_i, and the total-least-squares best-fitting plane H_{x_i}, passing through the weighted centroid c, is computed. The new guess x_{i+1} is the projection of r onto H_{x_i}.

For most r, however, the output point x is not a point of the MLS surface S. Again, it is tempting to think so: at convergence $x \in H_x$, and the normal to H_x is indeed $n(x)$ as defined by the MLS energy function. And since H_x is the total-least-squares best-fit plane, x is a local minimum, along the line $\ell_{x,n(x)}$, of the energy function

$$e'(y,a) = \sum_i (\langle a, p_i \rangle - \langle a, y \rangle)^2 \, \theta(x, p_i)$$

Notice that the weights on the points are fixed in e', while for e_{MLS} they vary with y; so e' is not precisely the same function as e_{MLS}. It is certainly true that at the point of convergence x, $e'(x,a) = e_{MLS}(x,a)$. But we cannot claim that, "Since (x,a) is a local minimum of $e'(y,n(x))$ on $\ell_{x,n(x)}$, and $e'(x,a) = e_{MLS}(x,a)$, therefore (x,a) is a local minimum of $e_{MLS}(y,n(x))$ on $\ell_{x,a}$." The two different functions generally have very slightly different minima.