Eurographics Symposium on Point-Based Graphics (2004)
M. Alexa, S. Rusinkiewicz, (Editors)

The Domain of a Point Set Surface

Nina Amenta † and Yong J. Kil ‡

University of California at Davis

Abstract

It is useful to be able to define a two-dimensional point-set surface *determined by a point cloud. One popular definition is Levin's MLS surface. This surface is defined on a domain which is a three-dimensional subset of* $\mathbb{R}^3$*, a narrow region around the input point cloud. If we were to extend the definition outside the domain, we would produce components of the surface which are far from the point cloud. This is important in practice, since when moving points onto the MLS surface, we need to begin with an initial guess which is within the domain.*
We visualize the domain in two dimensions, and explain why it is so narrow. We also consider two MLS variants which can be defined on a wider domain without producing spurious surface components. One is efficient and works well except near sharp corners. The other is computationally expensive but seems to work well everywhere.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Curve, surface, solid, and object representations

1. Introduction

The idea of modeling surfaces with clouds of points is becoming increasingly popular [Lin01, ABCO*03, ZPKG02, PKKG03, MVdF03, AK04], both because of improved technologies for capturing points from the surfaces of real objects and because improvements in graphics hardware enable models composed of many small primitives. Defining a *point-set surface* implied by a cloud of points can be very useful in rendering and manipulating point-set models.

Many of these techniques use a definition due to David Levin [Lev03]. He gives a procedure f for taking points in a neighborhood U of the point cloud towards a two-dimensional surface. The *MLS surface* is then defined as the fixed points of f, that is, the set of points x such that $x = f(x)$. Let us call the neighborhood U the *domain* of the MLS surface. Outside of the domain, there are also fixed points f which are not considered part of the point-set surface. It turns out that the domain of the MLS surface is surprisingly narrow; as we see in Figure 1, when the noise level exceeds expectations it may not even include the entire point cloud, causing some of the points in the cloud to project away from where the surface ought to be.

Levin's projection procedure is based on an energy function. In recent work [AK04], we analyzed this energy function by separating it into two components, one defining a vector field n and the other using n to define the energy at a point in space. In this paper we visualize those two components to see how they break down outside the domain. We consider other variants of MLS and show, again through visualization, that they have a wider domain; see Figure 2. We also observe that the MLS energy function defines a surface which collapses near sharp corners, and, outside of the domain, produces spurious components; one of our variants handles sharp corners more gracefully, although it is expensive to compute.

2. MLS surface

Levin defined the MLS surface as the set of stationary points of a procedure f. That is, a point x is on the surface if $x = f(x)$. This definition was used in a seminal paper [ABCO*03], which developed techniques for efficiently rendering point-set surfaces, which pointed the way for much of the subsequent work.

We argued [AK04] that the MLS surface is an exam-

† amenta@cs.ucdavis.edu
‡ kil@cs.ucdavis.edu

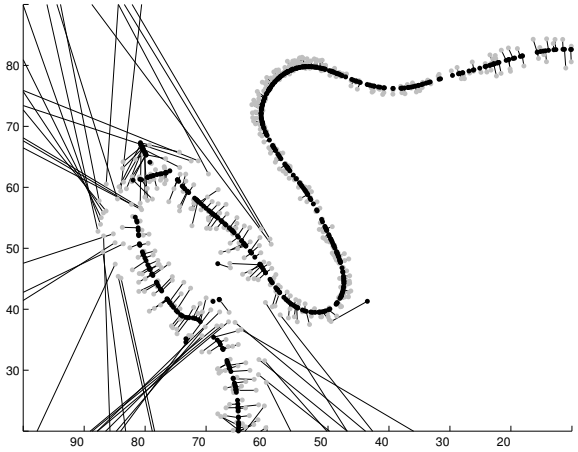

Figure 1: *Mapping points to the MLS surface using Levin's function f, on an input point cloud with variable density and noise level. The lines connect each gray input point x to $f(x)$, a black point. The Gaussian weight on each input point has standard deviation five. Where the width of the point cloud is greater than five, some input points move in directions away from, or parallel to, the surface, rather than towards it. This demonstrates that the domain of the MLS surface does not necessarily include the entire input point cloud; it depends on the width of the point cloud and the weighting factor as well as other factors such as density and distribution. Increasing the weighting factor would eliminate the bad projections at the cost of a small loss of detail.*

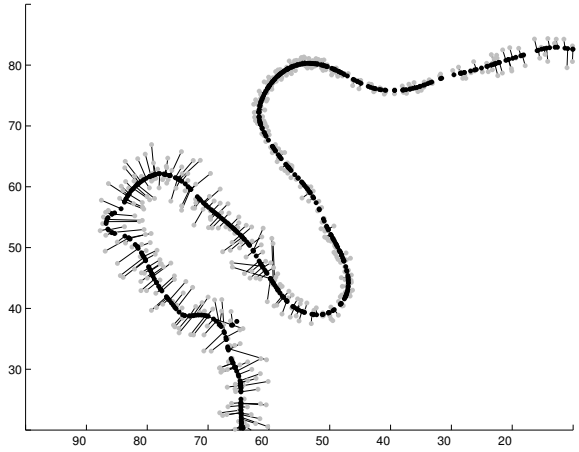

Figure 2: *Our (more expensive) integral variant of the MLS projection function, on the same inputs and using the same Gaussian weights. Almost all of the input points project correctly to the surface.*

ple of an *extremal surface*. Extremal surfaces were introduced by Guy and Medioni [GM97] and used by Medioni et. al. [MLT00] in a variety of applications; we modify the definition slightly for our purposes. An extremal surface is defined by two functions, a vector field $n(x)$, and a scalar-valued energy function $e(x,a)$ which takes a point x and a direction a as input. Intuitively, we can describe the extremal surface with a procedure to recognize its points. Given a point x, we first find its the direction vector $n(x)$. We consider the line that passes through x in direction $n(x)$. If x is a local minimum of $e(x,a)$ along the line, x is on the extremal surface; otherwise it is not.

Here is a more formal definition. Let a be an unoriented direction, by which we mean that we consider direction n and $-n$ to be the same; $\mathbb{P}^2$ is the space of unoriented directions in $\mathbb{R}^3$. Let $\ell_{x,a}$ be the line through point x with unoriented direction a, and let the notation arglocalmin_x refer to the set of inputs x producing local minima of a function of variable x.

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

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

be the extremal surface of n and e.

Since the MLS surface is an extremal surface we can understand its behavior by studying the functions $e(x,a)$ and $n(x)$. The MLS surface determined by an input point cloud P uses the function $e_{MLS}(x,a)$:

$$e_{MLS}(x,a) = \sum_{p_i \in P} \left(\langle a, p_i \rangle - \langle a, x \rangle\right)^2 \theta(x, p_i) \qquad (1)$$

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

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

The $d^2()$ is the square of the Euclidean distance and h is a constant scale factor, determining the region of influence around x. The function n_{MLS} is defined in terms of e_{MLS}:

$$n_{MLS}(x) = \text{argmin}_a \, e_{MLS}(x,a)$$

When the minimum is not unique, $n(x)$ is undefined.

An extremal surface consists of points which are local minima of e in direction n. If instead of considering only minima we consider both minima and maxima, we get a larger surface containing the extremal surface. Since at a critical point of e in line $\ell_{x,n(x)}$ the gradient of e is perpendicular to the line, this larger surface is characterized by the implicit equation:

$$g(x) = \vec{n}(x) \cdot \nabla_y \, e(y, \vec{n}(x))|_x) = 0 \qquad (2)$$

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. Since this is an implicit surface, it is a manifold wherever n is well-defined and assuming it avoids critical points of g. In Figure 9 and Figure 10, we show all of $g(x)$; interestingly, in neither case is the iso-surface $g(x) = 0$ a manifold.

3. Generating surface points

In practice(eg. [ABCO*03, ZPKG02]), points on point-set surfaces are often generated by projecting or simply moving nearby points onto the surface. There are variety of procedures for doing this (see [ABCO*03, ZPKG02], the discussion and procedure in [AK04], and [AA04] in this proceedings). Most of them, however, share the following property: they need to begin from a starting point which is within the domain in order to produce a part of the desirable surface near the point cloud. In some cases this starting point may be the point x which is to be moved onto the surface, and in others it may be an initial guess for the desired surface point. Usually a good starting point is simply assumed, although Adamson and Alexa [AA03] discuss the problem of choosing a good starting point in the context of ray-tracing and give a heuristic for guessing whether a starting point is within the domain or not. This issue motivates our study of the domain of the MLS surface and its variants.

4. Visualizing n and e

We now visualize e_{MLS} and n_{MLS} to understand the domain of the MLS surface. We begin with e_{MLS}. In Figure 9, lower left, we show iso-surfaces of $e_{MLS}(x, n_{MLS}(x))$. Notice that because the Gaussian weights on the points of P fade out as we move away from the point cloud, $e_{MLS}(x, n_{MLS}(x))$ will be very small and flat far from the surface. This is true however n_{MLS} is chosen. Weak local minima far from the surface might end up being defined as surface components.

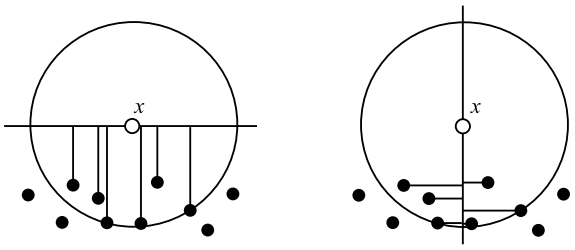

Figure 3: *Two possible planes at the white point x. Points of R near enough to x to strongly influence the choice of plane lie within the large circle around x. The sum of the distances of these points to the plane parallel to the surface, on the left, are larger than those for the plane perpendicular to the surface, on the right.*

As pointed out by Alexa et al. [ABCO*03] (conference version), this is easily corrected without having to limit the domain. If the Gaussian weights are replaced with *normalized* Gaussian weights

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

the value of $e_{MLS}(x, n(x))$ will be large far from the surface no matter what direction is chosen as $n(x)$. Suppose, in addition, that $n_{MLS}(x)$ ends up being roughly parallel to the surface normal at the point of S nearest to x. Then looking at Equation 1 and replacing θ with θ_N, we see that $e_{MLS}(x, n_{MLS}(x))$ should be a rough estimate of the squared distance from x to S (formed by a weighted average of the distances in direction n_{MLS} from nearby points of P). Intuitively, in that case the minimum of e_{MLS} along the line $\ell_{x,n_{MLS}(x)}$ should be near the point cloud since that is where the minima of the unsigned distance function are found.

Things work out well, then, when n_{MLS} is perpendicular to S at the point of S nearest to x. Sadly, this is not the case outside of a narrow region around the center of the point cloud. In Figure 5, we consider $n_{MLS}(x)$ at a selection of points x in the plane. Around each point we draw a spherical plot of $e_{MLS}(x, a)$, as a function of a with x fixed; see Figure 4. Since $n_{MLS}(x)$ is the minimal energy direction, the narrowest part of the plot is the direction chosen as $n_{MLS}(x)$.

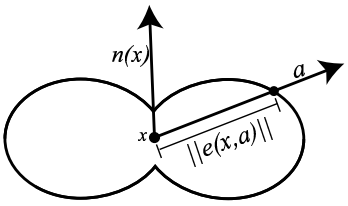

Figure 4: *The circular plots used in Figure 5, below, and elsewhere in the paper, to visualize the choice of $n(x)$. The energy associated with each direction a at point x is represented by the point $ce(x,a)a+x$ on the circular plot, where c is a constant that remains fixed within each image. Hence the direction of minimum width of the circular plot - the direction in which it is narrowest - represents the optimal direction $n(x)$. Dumbell-shaped plots like this one represent a strong minimum energy direction, while ellipsoidal plots are less strong, and the minimum direction represented by a circular or squarish plot is unstable.*

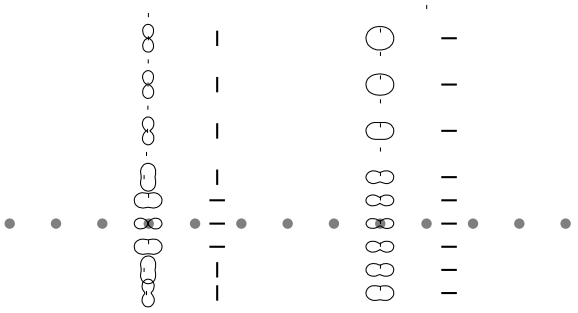

Figure 5: *We take points on a horizontal line as input. On the left, we show circular plots (see Figure 4 for an explaination) of the MLS energy for points x at various heights, and the best-fitting line, normal to $n_{MLS}(x)$, at each height. Notice that near the surface, the best fitting line is parallel to S, but as x moves away the best-fitting line turns as $n_{MLS}(x)$ becomes parallel to S. The right pair shows our e_1 energy plots and the corresponding best-fitting lines.*

We observe an interesting phenomenon. Although near

the center of the point cloud the optimal direction is (correctly) roughly perpendicular to the surface, only a little distance away from the center of the point cloud, the direction chosen as $n_{MLS}(x)$ is parallel to, rather than perpendicular to, the surface.

This can also be observed by looking at the stream lines of n_{MLS} in the larger example in Figure 9.

Why do we see this sudden change in n_{MLS}? Recall that h is the scale factor in the weighting function θ which roughly determines the range of influence for each point $p_i \in P$. Let us consider $n_{MLS}(x)$ as x moves away from the surface. When the distance of x from the nearby points of P is slightly greater than h, a tangent plane through x with normal similar to S accumulates an error of about h for every point near x, while a tangent plane through x perpendicular to S accumulates an error of at most h for any of these points, and an error of less than h for most of them. See Figure 3.

Thus, for many points farther than h from S, the direction $n_{MLS}(x)$ will be nearly parallel to S. Along line $\ell_{x,n_{MLS}(x)}$ the value of $e_{MLS}(x,a)$ will vary if only because P is a discrete point set. If the domain were to be extended to include this region in which n_{MLS} is parallel to the surface, local minima of $e_{MLS}(x,a)$ due to these slight variations would be counted as surface components. See the bottom left of Figure 6, and the blue curve of minima at the lower right in Figure 9. In both of these figures, a random set of points within the entire rectangular region are used as starting points for the function $f(x)$, converging to the set of stationary points on the MLS surface both inside and outside the domain. (In Figure 9, we also use color to show the rest of the implicit surface (Equation 2) containing the MLS surface. To do this, we changed the projection procedure to converge to points which are local *maxima* of e_{MLS} on the line $\ell_{x,n_{MLS}}$, rather than local minima. The minima are shown as blue points and the maxima as green points.)

We see, then, that it is n_{MLS} which requires the domain to be so narrow; the domain is essentially restricted to the area of radius roughly h around the surface itself within which n_{MLS} is well-behaved. When the point cloud has width greater than h, the point cloud itself extend outside the domain, causing spurious pieces of surface to be produced, as in Figure 1.

Notice that $e_{MLS}(x,a)$ (Equation 1) measures the sum-squared distance from the weighted points of P to the nearest point on the hyperplane $P(x,a)$ through x with normal a. Contemplating Figure 3 leads to the following insight: a better choice for the optimal direction would consider the squared distance from the weighted points on the plane $P(x,a)$ to the nearest point of P; in that case a plane which contains heavily weighted points far from P, as on the right in Figure 3, would be penalized.

5. Alternate point-set surface definition

By using different functions e and n we can produce different surfaces. In this section we consider alternative choices for which the domain is larger.

We begin with a simple fix which works well for smooth surfaces. Instead of choosing n as the normal of the best-fitting tangent plane $P(x,a)$ through x (which is what $n_{MLS}(x)$ does), we choose n to be the normal to the overall best-fitting plane to the points of P as weighted from x. This plane passes not through x but through the center of mass of the points of P as weighted from x. This is essentially the approach taken in the linear heuristic used to bring points onto the surface in PointShop3D [PKKG03, Pau03]. Formally, the direction n is given by:

$$n_{COM}(x) = \text{argmin}_a \sum_i (\langle a, p_i \rangle - \langle a, c \rangle)^2 \; \theta_N(x, p_i)$$

where the center of mass c is

$$c = \sum_i p_i \theta_N(x, p_i)$$

This reasonably simple fix works very well for smooth surfaces. See the third column in Figure 6. As we see in the Figure, however, it breaks down near sharp corners; unless the domain is restricted to be a very narrow region around the point cloud, a spurious sheet of the surface "shooting off" from the corner is produced. This motivated us to consider yet another alternative.

Recall that in the last section we realized that we would like to choose the direction $n(x)$ by minimizing the distance from hyperplane $P(x,a)$ to P, rather than the distance from P to $P(x,a)$. Unfortunately it seems that this must be expressed as an integral over $P(x,a)$. We first express the squared distance from a point x in space to the point cloud P, by averaging the distances from x to the nearby points of P.

$$\delta(x) = \sum_i \theta_N(x, p_i) d^2(x, p_i)$$

This approximate distance function is shown in Figure 7.

At every point $x \in \mathbb{R}^3$, $\delta(x)$ is an upper bound on the distance from x to the nearest point of P. This is because it is a weighted average of the distances from x to all of the points of P, all of which are at least the distance to the nearest point. Visualization of this function δ shows that it is smooth and a good approximation to the squared distance function in the sense that it is low near the point cloud and high near its medial axis, as shown in Figure 7.

We define an energy function e as the weighted integral of δ^2 over all of $P(x,a)$:

$$e_I(x,a) = \int_{y \in P(x,a)} \delta^2(y) \; \theta(y,x)$$

where

$$\theta(x,a) = e^{-d^2(x,r)/h^2}$$

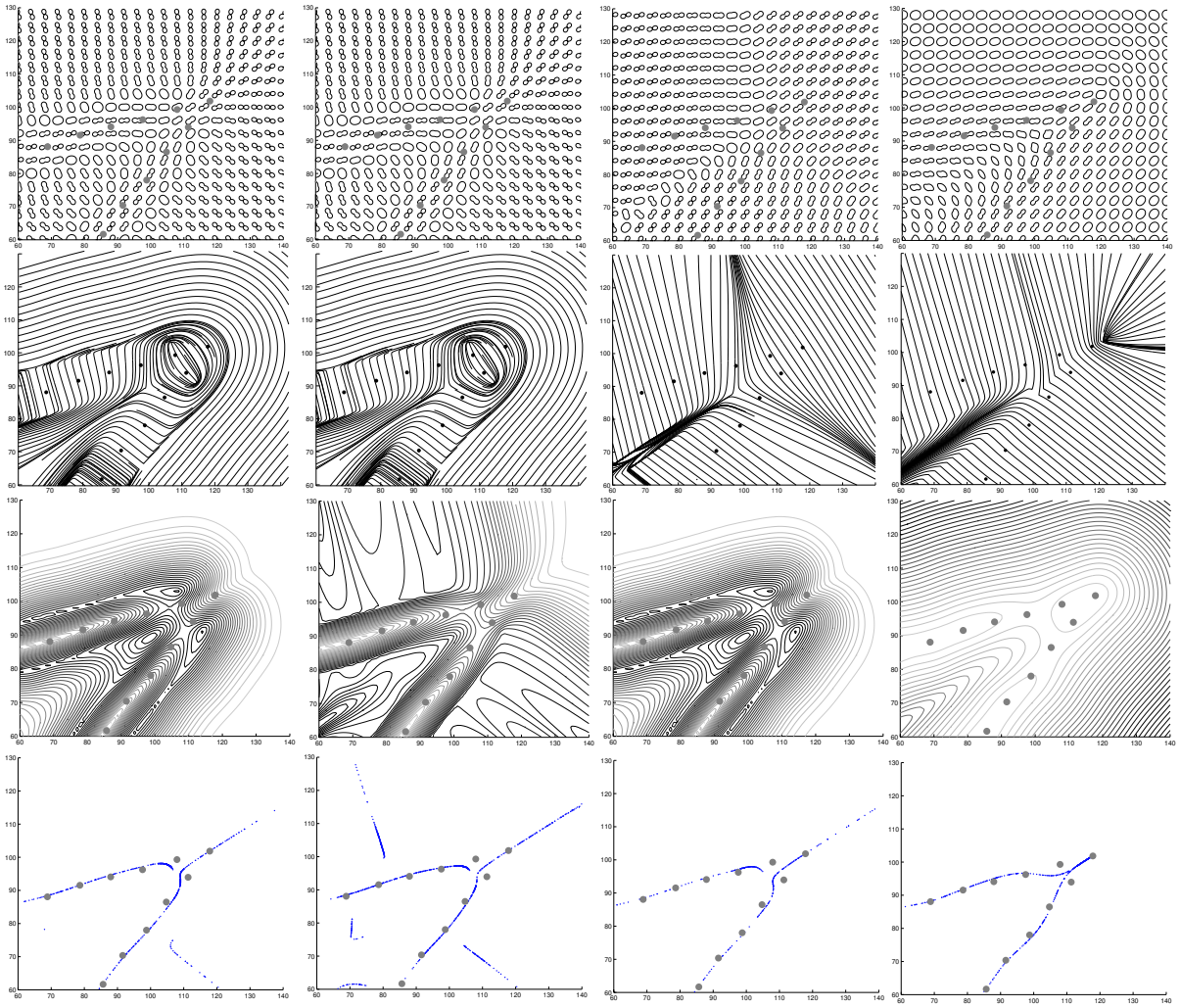

Figure 6: *Circular energy plots (as defined in Figure 4), stream lines of the vector field $n(x)$, iso-contours of $e(x,n(x))$, and the points recognized as surface points for four different energy functions. The lighter grey iso-contours have lower energy values. The input points (solid circles) are slightly perturbed so as to be in general position. From left to right, we show Levin's original definition of MLS, then MLS with the weights normalized as in [ABCO*03], then the variant which chooses the normal to the total-least-squares best fitting plane as $n_{COM}(x)$ (similar to the idea used in PointShop), and on the right our integral method. The n_{COM} variant works well except near the sharp corner.*

is the *unnormalized* Gaussian weighting function. (We discuss the rationale for using δ^2 rather than δ in the appendix). Expanding this out and swapping the sum and the integral, we get

$$e_I(x,a) = \sum_{r \in R} \int_{y \in P(x,a)} \theta(y,x)\ (\theta_N(y,r) d^2(y,r))^2$$

It is impossible to solve the integral analytically since it includes the normalized weighting function θ_N, whose value at y depends on all of the points of the point cloud P. Instead we estimate the integral by evaluating δ^2 at a collection of points $y \in P(x,a)$ near x.

Again, we use this energy function to choose n_I, as with the MLS energy function:

$$n_I(x) = \text{argmin}_a\, e_I(x,a)$$

Figure 10 shows iso-contours of $e_I(x,n_I(x))$ and stream lines of n_l. The direction n_l appears to be pointing correctly towards the nearest part of the point cloud everywhere except near the medial axis, while the energy landscape of e_I has a valley near the point cloud and increases steadily as we move away. The surface they produce is reasonable even near sharp corners.

Here we are using the integral energy function e_I as the

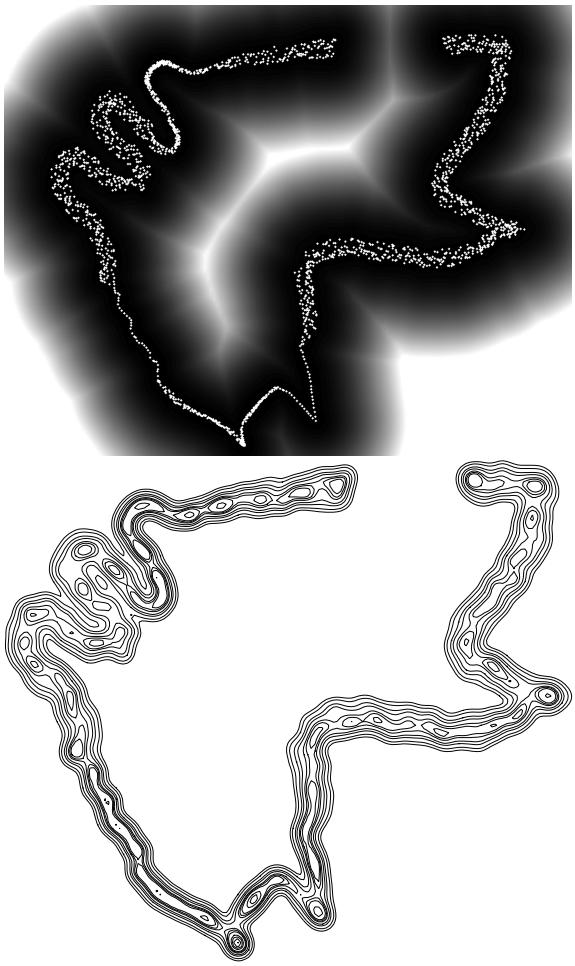

Figure 7: *Visualizations of the approximate squared distance function δ. Near the point cloud P, δ is roughly h, the weighting factor of the Gaussian, and we can see from the intensity-mapped image on the top that it increases smoothly with distance, so that far from P, δ is a very good approximation of the squared distance function. Near P, as we see from the iso-contours in the bottom figure, δ forms an uneven valley with discrete minima.*

energy function e in the extremal surface definition. A simpler alternative, with almost identical behavior, is to use δ as the energy function e in the extremal surface definition.

6. Noise and the domain

The domain of a point-set surface is interesting since it has to contain the starting point for processes taking a point onto the surface. This is particularly relevant to noisy point clouds, where an input point itself may not be a good starting point. In Figure 1 we showed an example using the procedure proposed by Levin and implemented by the non-linear projection method given by Alexa et. al. [ABCO*03], where with the given scale factor $h = 5$ on the Gaussian weights, the noise level in the data was large enough (slightly greater than 5) that some of the input points lay outside the domain and failed to move towards the desired surface. Using our integral point-set surface definition, for which the domain is less narrow, we saw in Figure 2 that almost all the points move towards the surface (using the same scale factor $h = 5$).

Here we give some other visualizations of this example, to see exactly how the vector field n and e differ, and how this affects the motion of the input points.

7. Conclusions and Open Questions

The definition of useful point-set surfaces and procedures to use them is a very interesting question, which we certainly have not answered definitively in this paper. While n_{COM} is efficient and works well for smooth surfaces, it would be nice to improve the behavior near sharp corners. It might even be possible to define functions n and e which produce real sharp corners in the point-set surface. Perhaps the intuition we have provided through these visualizations will help move the field towards this goal.

References

[AA03] ADAMSON A., ALEXA M.: Ray tracing point set surfaces. In *Proceedings of Shape Modeling International 2003* (2003), pp. 272–279.

[AA04] ALEXA M., ADAMSON A.: On normals and projection operators for surfaces defined from points. In *Proceedings of the Symposium on Point-Based Graphics 2004* (2004).

[ABCO*03] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics 9*, 1 (2003). An earlier version appeared in IEEE Visualization 2001.

[AK04] AMENTA N., KIL Y. J.: Defining Point Set Surfaces. manuscript, later version to appear in Siggraph 2004, March 2004.

[GM97] GUY G., MEDIONI G.: Inference of surfaces, 3d curves and junctions from sparse, noisy, 3d data. *IEEE Trans. on Pattern Analysis and Machine Intelligence 19*, 11 (1997), 1265–1277.

[Lev03] LEVIN D.: Mesh-independent surface interpolation. In *Geometric Modeling for Scientific Visualization*, Brunnett G., Hamann B., Mueller K.,, Linsen L., (Eds.). Springer-Verlag, 2003.

[Lin01] LINSEN L.: *Point cloud representation*. Tech. rep., Faculty of Computer Science, University of Karlsruhe, 2001.

[MLT00] MEDIONI G., LEE M.-S., TANG C.-K.: *A Computational Framework for Segmentation and Grouping*. Elsevier, 2000.

[MVdF03] MEDEROS B., VELHO L., DE FIGUEIREDO L. H.: Moving least squares multiresolution surface approximation. In *Proceedings of SIBGRAPI 2003 - XVI Brazilian Symposium on Computer Graphics and Image Processing* (2003).

[Pau03] PAULY M.: *Point Primitives for Interactive Modeling and Processing of 3D Geometry*. PhD thesis, ETH Zurich, 2003.

[PKKG03] PAULY M., KEISER R., KOBBELT L., GROSS M.: Shape modeling with point-sampled geometry. *ACM SIGGRAPH 2003* (2003).

[ZPKG02] ZWICKER M., PAULY M., KNOLL O., GROSS M.: Pointshop 3d: An interactive system for point-based surface editing. *ACM SIGGRAPH 2002* (2002).

Appendix A: Details

One might think that integrating the function δ, instead of δ^2, would suffice for selecting the optimal projection direction in our proposed integral energy function e_I. We believe that it would work well near a cloud of points scattered around a line. But far from P, especially in regions of high curvature, the weight of the nearest point $p \in P$ in the point cloud dominates the squared distance function δ. In two dimensions, we can model this situation as follows. We assume that for all y, $\theta_N(y,p) = 1$ and hence $\theta_N(y,p_j) = 0$ for all other $p_j \in P$. We place p at the origin, x at $(0,k)$, we let (u,v) be a unit vector in the direction a of line $P(x,a)$ and we use the parameter α to express a point $y \in P(x,a)$ as $(0,k)+\alpha(u,v)$. We have

$$\tilde{e}(x,a) = \int_{\alpha=-\infty}^{\infty} \theta(y,x)\,(\alpha^2u^2+\alpha^2v^2+2\alpha kv+k^2)$$

$$= \int_{\alpha=-\infty}^{\infty} \theta(y,x)\,(\alpha^2+2k\alpha v+k^2)$$

How does this function change as a function of the tangent direction (u,v)? It is constant! Only the linear term depends on v, and since we integrate from $-\infty$ to ∞, every positive value of $2\alpha kv$ is canceled out by a corresponding negative value. So $f(x,a)$ is the same for any choice of (u,v).

We therefore increase the penalty for aligning (u,v) with the gradient of δ by squaring δ in the integral. Now in the simplified model of the situation in which δ is determined by a single point p we have $e_I(x,a) =$

$$\int_{\alpha=-\infty}^{\infty} \theta(y,x)\,(\alpha^4+4k^2\alpha^2v^2+k^4+4k\alpha^3v+2k^2\alpha^2+4k^3\alpha v)$$

Here again the terms linear in v cancel out in the integral, but the term $2k^2\alpha^2v^2$ grows with the absolute value of v; that is, $e_I(x,a)$ is minimized when $(u,v) = (1,0)$ is perpendicular to the gradient of the distance function.

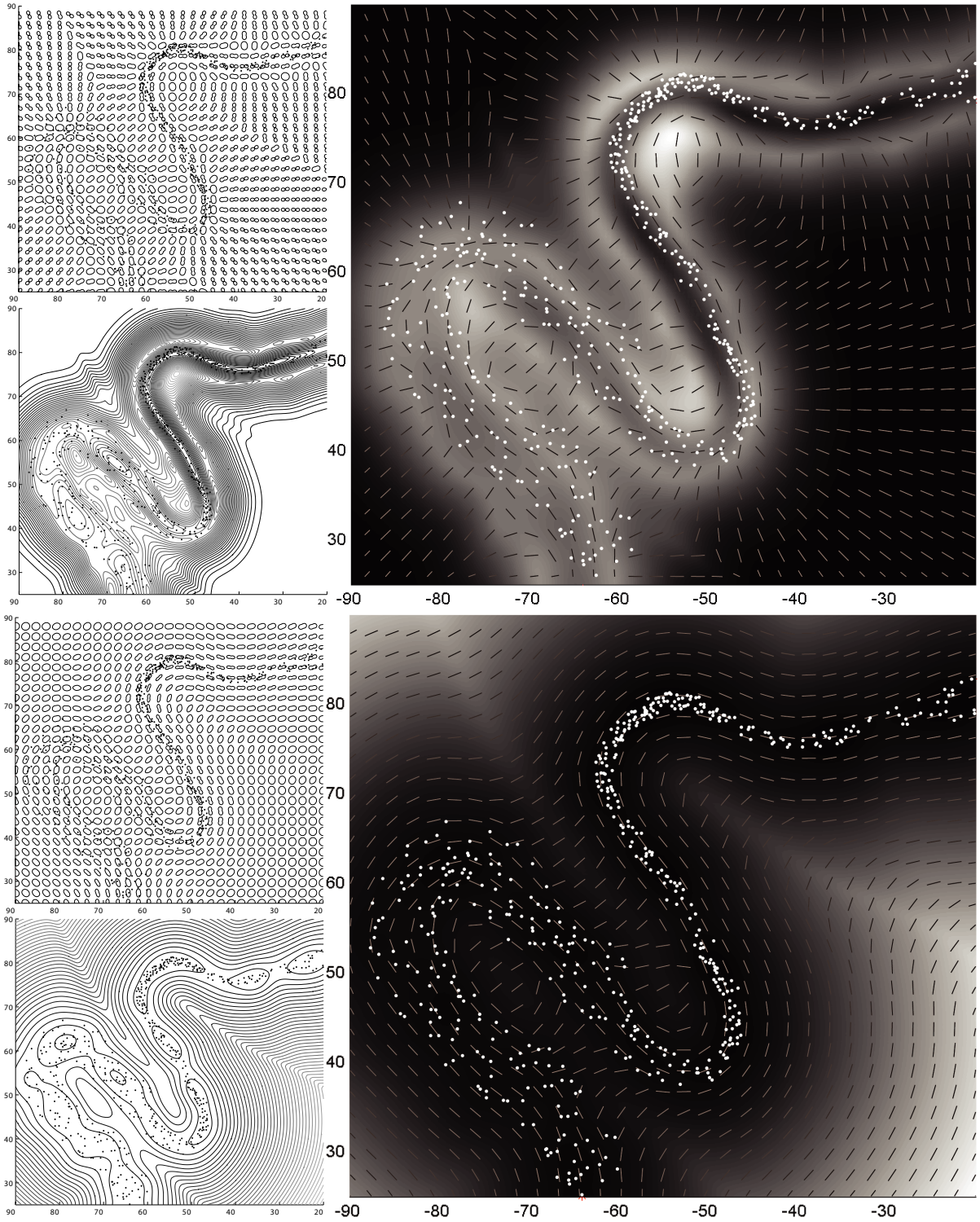

Figure 8: *On the top, the MLS energy function on the noisy example. In the upper left, the circular plots, as defined by Figure 4, show that in the noisy region the minimum-energy direction n_{MLS} switches from perpendicular to the surface in the middle of the cloud to parallel to the surface near the outside. In the large image, the gray lines represent the best-fitting planes, and so are* perpendicular *to n_{MLS}; the background intensity represents $e_{MLS}(x, n_{MLS}(x))$. On the bottom, similar visualizations for our integral energy function; notice that n_I changes much more smoothly.*

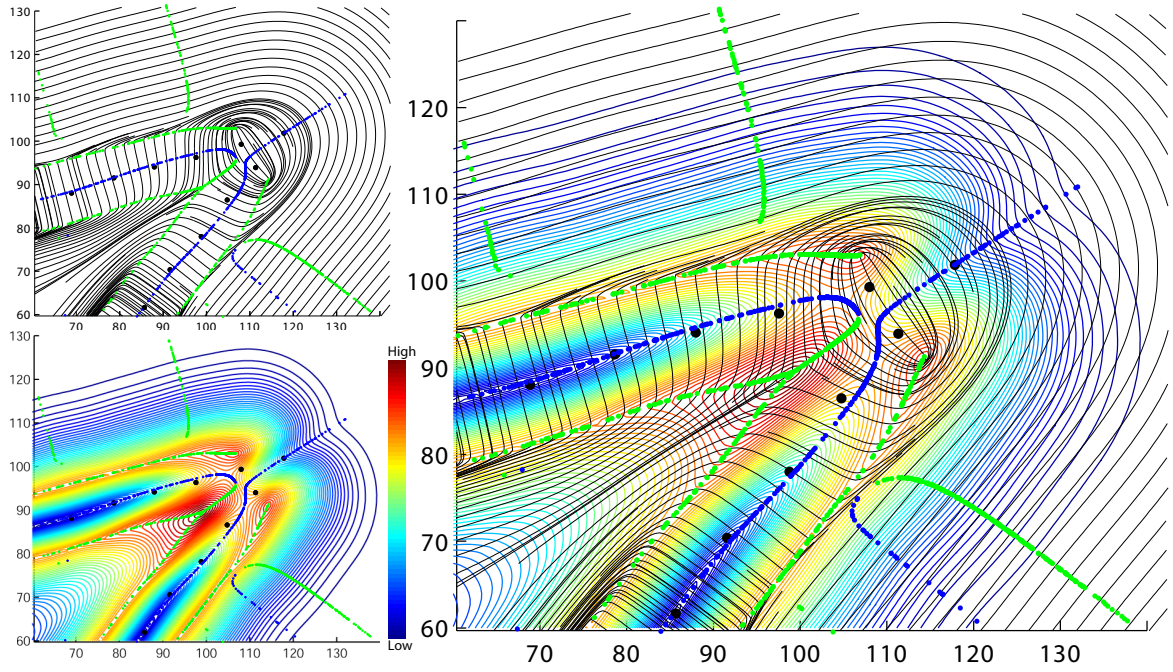

Figure 9: *Top left, the stream lines of the vector field generated by $n_{MLS}(x)$. Bottom left, the energy function $e_{MLS}(x, n_{MLS}(x))$. Right, both together. The scale factor on the weights in the MLS energy function $h = 10$. Points x of the MLS surface, shown in blue, are local minima of e_{MLS} on the stream lines. The local* maxima*, which also satisfy Equation 2, are shown in green. The endpoints and junctions in the union of the blue and green curves are singularities of n_{MLS}.*

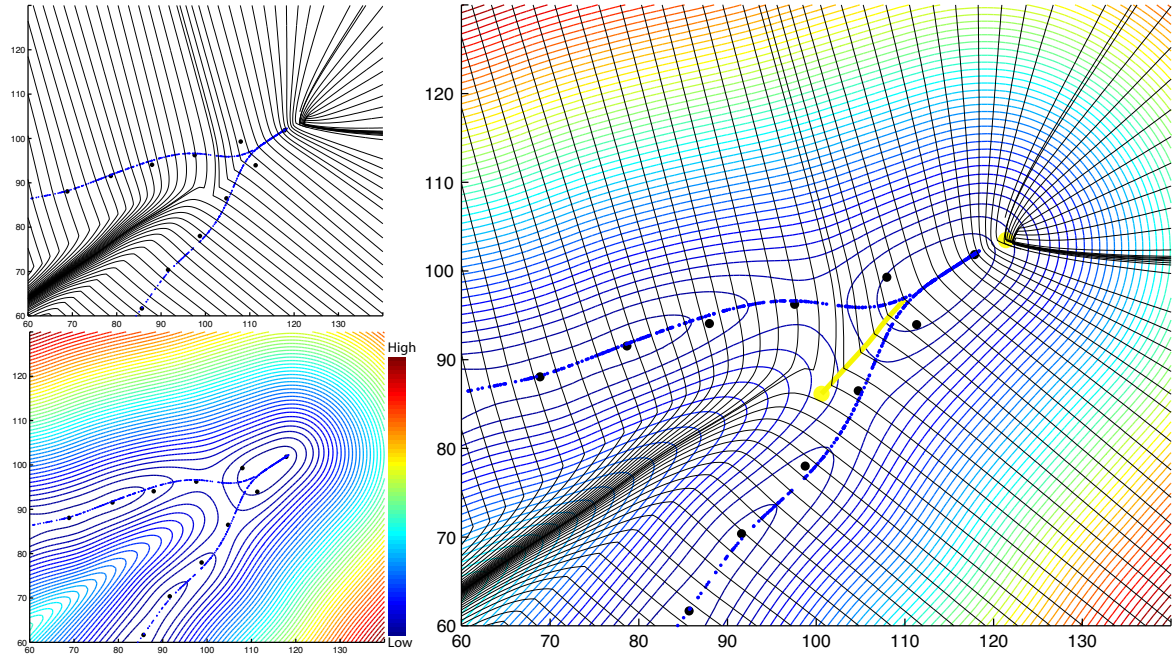

Figure 10: *Top left, stream lines of n_l, bottom left iso-contours of $e_l(x, n_l(x))$. As above, $h = 10$. We believe that the apparent junction point on the point-set surface again is actually two components near each other but not meeting. We believe the upper curve should continue as (green) local maxima, similar to MLS, above, but our procedure failed to locate the maxima at the sharp corners in the stream lines, hilighted in yellow. The yellow dots are drawn near the singularities of n_l, where the two curve components should end.*