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

On Normals and Projection Operators for Surfaces Defined by Point Sets

Marc Alexa[†] and Anders Adamson[‡]

Department of Computer Science, Darmstadt University of Technology, Germany

Abstract

Levin's MLS projection operator allows defining a surface from a set of points and represents a versatile procedure to generate points on this surface. Practical problems of MLS surfaces are a complicated non-linear optimization to compute a tangent frame and the (commonly overlooked) fact that the normal to this tangent frame is not the surface normal. An alternative definition of Point Set Surfaces, inspired by the MLS projection, is the implicit surface version of Adamson & Alexa. We use this surface definition to show how to compute exact surface normals and present simple, efficient projection operators. The exact normal computation also allows computing orthogonal projections.

Categories and Subject Descriptors (according to ACM CCS): G.1.2 [Numerical Analysis]: Approximation of surfaces and contours I.3.5 [Computer Graphics]: Curve, surface, solid, and object representations

1. Introduction

Point sets have become an increasingly popular shape representation, for modeling [ZPKG02, PKKG03] as well as rendering [PZvBG00, RL00, KV01, ZPvBG02]. Most shape processing and rendering tasks require the approximation of a continuous surface from the point data as well as efficient computational methods for generating points on the surface.

Levin's projection procedure [Lev03] has gained popularity as a tool for solving both problems: The projection operation takes points close to an anticipated surface approximation onto this surface and the set of fix points of the projection is conjectured to be a smooth surface. Thus, the projection operator defines the surface *and* provides the computational tool to generate points on that surface. These useful properties for surface modeling operations in general have been discussed in [ABCO*01]. Pauly et al. exploit the projection operator for a wide range of practically useful modeling operations [PKKG03].

An oftentimes overlooked problem of the MLS surface definition is that the approximating tangent frame is not tangent to the surface (this has also recently been discussed by Amenta & Yong [AK04]). Recall that the MLS projection is a two-step procedure, where the crucial properties follow from the first step. In the first step, a local approximating tangent frame for a point $\mathbf{p}$ close to the surface is computed. In the second step, a local polynomial is fitted to the points using the tangent frame as the parameter domain. Most follow-up works use only the first step (i.e. assuming a constant polynomial approximation in the second step) and assume that the normal of the approximating tangent frame is the surface normal. We show that this is not true. As a consequence, a projection operation based on only the first step is not orthognal (though this has been claimed, see e.g., [PKKG03]).

A related definition of a smooth surface from points, basically following the ideas of Levin [Lev03] and presented as an attempt to simplify the computation in [ABCO*03] is Adamson & Alexa's implicit version [AA03]. As implicit surfaces allow easy intersection computation with parametric curves, they have proposed ray intersection as the main tool to compute points on the surface. In [AA04] they explain how to define surface boundaries and demonstrated that ray surface intersection works even for unsigned im-

[†] alexa@informatik.tu-darmstadt.de
[‡] aadamson@gris.informatik.tu-darmstadt.de

plicits – yielding a way to define possibly bounded or non-orientable surfaces with point sets.

Here, we briefly re-introduce this surface definition and then show that the gradient of the implicit function can be computed explicitly. This leads to accurate surface normals. We compute these normals and compare them to normals resulting from a locally weighted co-variance analysis. It turns out that for densely sampled smooth surfaces the difference is very small, yet non-zero. We sketch a proof showing that the first step of the MLS projection procedure does not yield surface normals.

As the projection operator has turned out to be a useful primitive in many modeling situations, we introduce projection operations for the implicit surface. The properties of ray-intersection (i.e., the surface might have boundaries and need not be orientable) carry over to the projection. We show that the projection operation is easy to compute and is adaptable to have certain features, e.g. to be orthogonal.

From the fact that the projection is orthogonal, one can deduce that it is stationary on the medial axis. Thus the implicit version of the surface could as well include the medial axis. In future work, one might use this properties to relate this approach to the sampling criteria developed by Amenta et al. [ABE98, ABK98, ACK01].

2. Definition of the Implicit Surface

We assume that a set of points implicitly defines a smooth manifold surface, possibly with boundary. More specifically, let points $\mathcal{P} = \{p_i \in \mathbb{R}^3\}, i \in \{1,\dots,N\}$, be sampled from a surface $\mathcal{S}$ (possibly with a measurement noise).

We first define a neighborhood of $\mathcal{P}$ as the union of a set of balls centered in the $\mathbf{p}_i$:

$$\mathcal{B} = \{\mathbf{x} | d_{\mathcal{P}}(\mathbf{x}) < r_B\} = \bigcup_i B_i, B_i = \{\mathbf{x}, ||\mathbf{x}-\mathbf{p}_i|| < r_B\} \quad (1)$$

It is assumed that $\mathcal{B}$ contains the surface $\mathcal{S}$ as well as its approximation that we are going to define. For the definition we use two functions defined on the neighborhood: the weighted average and the normal direction. The weighted average $\mathbf{a} : \mathcal{B} \to \mathcal{B}$ maps each point $\mathbf{x}$ in the neighborhood of the points to the weighted average of the points, where the weights depend on the location of $\mathbf{x}$. The normal direction $n : \mathcal{B} \to \mathbb{S}^2$ assigns each point in the neighborhood of the point set a normal, thus, establishing an approximating tangent frame to the surface. For ease of notation, we identify the direction n with a unit vector $\mathbf{n} \in \mathbb{R}^3, \|\mathbf{n}\| = 1$.

Let an implicit function f be defined as

$$f(\mathbf{x}) = \mathbf{n}(\mathbf{x})^{\mathsf{T}}(\mathbf{x} - \mathbf{a}(\mathbf{x})), \quad (2)$$

then the approximating surface is

$$\hat{\mathcal{S}} = \{\mathbf{x} \in \mathcal{B} | f(\mathbf{x}) = 0\} \quad (3)$$

If we assume that $\mathbf{a}$ and $\mathbf{n}$ are continuously differentiable functions (and that $\mathbf{n}$ is unique within $\mathcal{B}$) then $\hat{\mathcal{S}}$ is a two-dimensional surface.

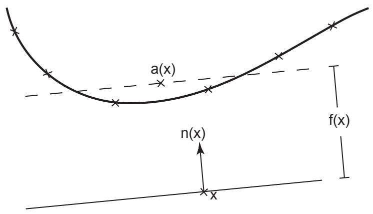

Figure 1: *The surface is defined implicitly as the zero set of a function $f(\mathbf{x})$. In each point $\mathbf{x}$ a local normal direction $\mathbf{n}(\mathbf{x})$ is estimated. The implicit function $f(\mathbf{x})$ describes the distance of a weighted average $\mathbf{a}(\mathbf{x})$ of the points along normal direction.*

For practical definitions of $\mathbf{a}$ and $\mathbf{n}$, a weight function $\theta : \mathbb{R} \to \mathbb{R}$ specifies the influence of a point. Weight functions are assumed to be smooth, positive, and monotonically decreasing (have negative first derivative).

Then, the weighted average of points at a location $\mathbf{x}$ in space could be described as

$$\mathbf{a}(\mathbf{x}) = \frac{\sum_{i=0}^{N-1} \theta(\|\mathbf{x}-\mathbf{p}_i\|)\mathbf{p}_i}{\sum_{i=0}^{N-1} \theta(\|\mathbf{x}-\mathbf{p}_i\|)}. \quad (4)$$

We describe two ways to define normal directions in each location $\mathbf{x}$:

1. Based on weighted covariance directions in $\mathbf{x}$ the normal could be defined as the direction of smallest weighted co-variance. This definition allows drawing a connection to Levin's MLS surfaces.
2. Assuming normals are supplied with the points, a normal in $\mathbf{x}$ could be computed using a weighted average of the given normals.

The direction of smallest covariance could be understand as a least squares fit of a plane with unit normal $\mathbf{n}$ through $\mathbf{x}$, i.e. the minimizer of

$$\min_{||\mathbf{n}||=1} \frac{\sum_i \left\| \mathbf{n}^{\mathsf{T}}(\mathbf{x}-\mathbf{p}_i) \right\|^2 \theta(\|\mathbf{x}-\mathbf{p}_i\|)}{\theta(\|\mathbf{x}-\mathbf{p}_i\|)}. \quad (5)$$

This constrained minimization problem is solved by one of the eigenvectors of the covariance matrix $\mathbf{W}(\mathbf{x}) = \{w_{jk}\}$, where

$$w_{jk} = \sum_i \left(\mathbf{e}_j^{\mathsf{T}}(\mathbf{x}-\mathbf{p}_i)\right)\left(\mathbf{e}_k^{\mathsf{T}}(\mathbf{x}-\mathbf{p}_i)\right)\theta(||\mathbf{p}_i-\mathbf{x}||) \quad (6)$$

and $\mathbf{e}_i, i \in \{0,1,2\}$ is a basis of $\mathbb{R}^3$. Let $\{\mathbf{v}_i\}$ be the eigenvectors of $W(\mathbf{x})$ corresponding to the eigenvalues $\lambda_0 \le \lambda_1 \le \lambda_2$, we set $\mathbf{n} = \mathbf{v}_0$ for any $\mathbf{x} \in \Omega$.

If each point $\mathbf{p}_i$ carries a normal $\mathbf{n}_i$ we can define the normal $\mathbf{n}(x)$ using weighted averages as for the points:

$$\mathbf{n}(\mathbf{x}) = \frac{\sum_{i=0}^{N-1} \theta(\|\mathbf{x}-\mathbf{p}_i\|)\mathbf{n}_i}{\left\|\sum_{i=0}^{N-1} \theta(\|\mathbf{x}-\mathbf{p}_i\|)\mathbf{n}_i\right\|}. \quad (7)$$

Note that the equation above should be understood in an abstract sense, i.e., a reasonable way to compute weighted averages of normal directions should be used. We have experienced no particular problem with using a vector representation, though.

3. Gradient & Normal

It has become standard practice to use the normal $\mathbf{n}(\mathbf{x}), \mathbf{x} \in \mathcal{S}$ of the approximating tangent plane as the surface normal [ABCO*01, ABCO*03, PKKG03]. It seems that some authors assume that this *is* the normal to $\mathcal{S}$ in $\mathbf{x}$, however, it is generally not.

In comparison to the MLS surface definition, the implicit description allows the exact evaluation of surface normals using the gradient of f. In the following, we describe how to compute this gradient in a point $\mathbf{x}$ explicitly, i.e. without taking finite differences. We feel this is an advantage over the MLS definition of the surface. It will also allow constructing an orthogonal projection operator.

In the following we first explain how to compute the gradient of f, which points in normal direction to the surface. We give a small example that, in contrast, the direction of smallest co-variance is not necessarily in the direction of the surface normal. Based on this observation we sketch a proof that the normals obtained in the first step of the MLS projection procedure are not surface normals.

3.1. Computing exact surface normals

We examine the gradient of f in the ortho-normal system $\{\mathbf{e}_k\}$, i.e.

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{e}_0}, \frac{\partial f(\mathbf{x})}{\partial \mathbf{e}_1}, \ldots\right). \quad (8)$$

The product rule for differentiating vector fields yields the directional derivatives of f:

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{e}_k} = \frac{\partial \mathbf{n}(\mathbf{x})^\mathsf{T}}{\partial \mathbf{e}_k}(\mathbf{x}-\mathbf{a}(\mathbf{x})) + \mathbf{n}(\mathbf{x})^\mathsf{T}\left(\mathbf{e}_k - \frac{\partial \mathbf{a}(\mathbf{x})}{\partial \mathbf{e}_k}\right), \quad (9)$$

We see that the difference between $\nabla f(\mathbf{x})$ and the normal direction $\mathbf{n}(\mathbf{x})$ is not necessarily in the direction of $\mathbf{n}(\mathbf{x})$ (see Figure 2 for an example where $\mathbf{n}(\mathbf{x})$ is certainly not in direction of the surface normal). Computing the gradient requires the evaluation of directional derivatives of $\mathbf{n}(\mathbf{x})$ and $\mathbf{a}(\mathbf{x})$.

Taking directional derivatives of $\mathbf{a}(\mathbf{x})$ along the basis directions $\mathbf{e}_k$ is straightforward and yields

$$\frac{\partial \mathbf{a}(\mathbf{x})}{\partial \mathbf{e}_k} = 2\frac{\sum \mathbf{p}_i \frac{\mathbf{e}_k^\mathsf{T}(\mathbf{x}-\mathbf{p}_i)\theta_i'}{\|\mathbf{x}-\mathbf{p}_i\|} \sum \theta_i - \sum \mathbf{p}_i\theta_i \sum \frac{\mathbf{e}_k^\mathsf{T}(\mathbf{x}-\mathbf{p}_i)\theta_i'}{\|\mathbf{x}-\mathbf{p}_i\|}}{(\sum \theta_i)^2}, \quad (10)$$

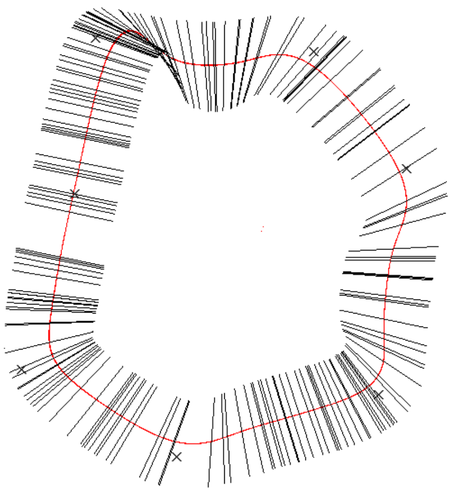

Figure 2: *An illustration of* $\mathbf{n}(\mathbf{x})$ *defined as the directions of smallest weighted co-variance. Random points on the zero-set contour have been chosen and the direction of smallest covariance is depicted as a line. Note that in the upper area* $\mathbf{n}(\mathbf{x})$ *is not normal to the contour.*

where $\theta_i = \theta(\|\mathbf{x}-\mathbf{p}_i\|)$ and $\theta_i' = \theta'(\|\mathbf{x}-\mathbf{p}_i\|)$. If the normal $\mathbf{n}(\mathbf{x})$ is defined as a weighted average as well, the computation can be performed in the same manner. Note that we assume the derivative of the weight functions can be computed analytically, which is certainly true for the typically used piecewise polynomial functions.

Computing the derivatives of the direction of smallest co-variance is slightly more complex but, nevertheless, can be performed explicitly. Let the covariance be $\mathbf{W}(\mathbf{x})$ and its smallest eigenvalue be $\lambda_0(|x)$ as before. The eigenvector of $\mathbf{W}(\mathbf{x})$ corresponding to $\lambda_0(\mathbf{x})$ is $\mathbf{n}(\mathbf{x})$, i.e.

$$\mathbf{n}(\mathbf{x})^\mathsf{T}\mathbf{W}(\mathbf{x}) = \lambda_0(\mathbf{x})\mathbf{n}(\mathbf{x})^\mathsf{T}. \quad (11)$$

Taking directional derivatives on both sides yields

$$\frac{\partial \mathbf{n}(\mathbf{x})^\mathsf{T}}{\partial \mathbf{e}_k}\mathbf{W}(\mathbf{x}) + \mathbf{n}(\mathbf{x})^\mathsf{T}\frac{\partial \mathbf{W}(\mathbf{x})}{\partial \mathbf{e}_k} = \frac{\partial \lambda_0(\mathbf{x})}{\partial \mathbf{e}_k}\mathbf{n}(\mathbf{x})^\mathsf{T} + \lambda_0\frac{\partial \mathbf{n}(\mathbf{x})^\mathsf{T}}{\partial \mathbf{e}_k}. \quad (12)$$

In the Appendix we show that for our definition of $\mathbf{W}$ (including the assumption that $\mathbf{n}(\mathbf{x})$ is unique and, thus, λ_0 is a single eigenvalue) the directional derivative of the eigenvalue is

$$\frac{\partial \lambda_0(\mathbf{x})}{\partial \mathbf{e}_k} = \mathbf{n}(\mathbf{x})^\mathsf{T}\frac{\partial \mathbf{W}(\mathbf{x})}{\partial \mathbf{e}_k}\mathbf{n}(\mathbf{x}) \quad (13)$$

which can be used to compute the directional derivative of

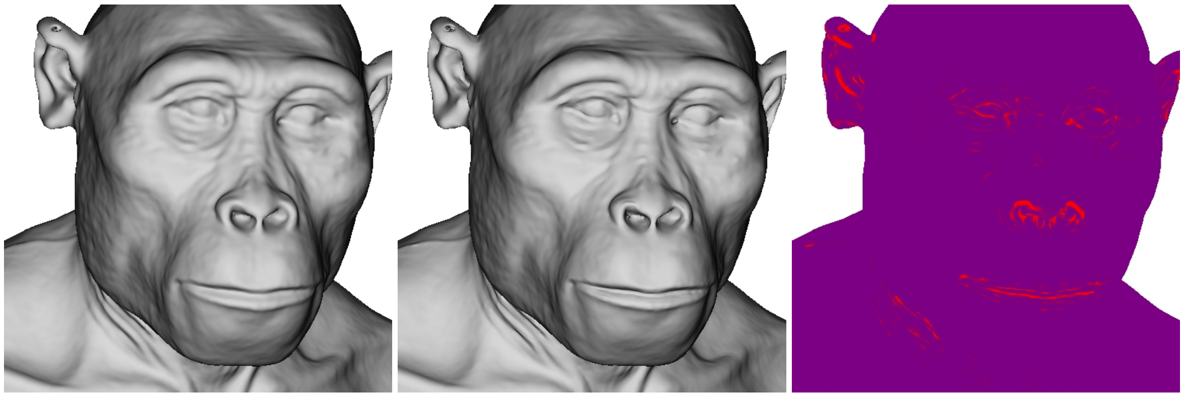

Figure 3: *A comparison of using the direction of smallest co-variance* $\mathbf{n}$ *and* ∇f *for image synthesis. The images have been generated by ray tracing the surface. The left image is shaded using* $\mathbf{n}$ *and appears slightly smoothed in areas of high complexity as compared to the center image, which has been generated using* ∇f *for shading. The right images illustrates the difference by color coding the scalar product* $\mathbf{n}^{\mathsf{T}}\nabla f/\|\nabla f\|$.

$\mathbf{n}(\mathbf{x})$ as

$$\frac{\partial \mathbf{n}(\mathbf{x})^{\mathsf{T}}}{\partial \mathbf{e}_k} = \mathbf{n}(\mathbf{x})^{\mathsf{T}} \left(\frac{\partial \lambda_0(\mathbf{x})}{\partial \mathbf{e}_k}\mathbf{I} - \frac{\partial \mathbf{W}(\mathbf{x})}{\partial \mathbf{e}_k} \right) (\mathbf{W}(\mathbf{x}) - \lambda_0 \mathbf{I})^{-1} \tag{14}$$

For easy re-implementation we give the coefficients of the directional derivative of $\mathbf{W}$ explicitly:

$$\begin{aligned}\frac{\partial w_{jm}}{\partial \mathbf{e}_k} = \sum_i & \left(\mathbf{e}_j^{\mathsf{T}}\mathbf{e}_k\right)\left(\mathbf{e}_m^{\mathsf{T}}(\mathbf{x}-\mathbf{p}_i)\right)\theta_i \\ & + \left(\mathbf{e}_j^{\mathsf{T}}(\mathbf{x}-\mathbf{p}_i)\right)\left(\mathbf{e}_m^{\mathsf{T}}\mathbf{e}_k\right)\theta_i \\ & + \left(\mathbf{e}_j^{\mathsf{T}}(\mathbf{x}-\mathbf{p}_i)\right)\left(\mathbf{e}_m^{\mathsf{T}}(\mathbf{x}-\mathbf{p}_i)\right)\frac{\mathbf{e}_k^{\mathsf{T}}(\mathbf{x}-\mathbf{p}_i)\theta_i'}{\|\mathbf{x}-\mathbf{p}_i\|}.\end{aligned} \tag{15}$$

Figure 3 illustrates the difference of using the gradient or directions of smallest co-variance for shading a ray traced image.

3.2. MLS surface normals

We show that the normals of approximating tangent frames in the MLS projection are not surface normals. Recall that the approximating tangent frame for a point $\mathbf{x}$ is computed as the local minimum of

$$e_{MLS} = \sum_{i=1}^{N} \left(\mathbf{n}^{\mathsf{T}}(\mathbf{p}_i - \mathbf{x} - t\mathbf{n})\right)^2 \theta\left(\|\mathbf{p}_i - \mathbf{x} - t\mathbf{n}\|\right) \tag{16}$$

with smallest t. The projection of $\mathbf{x}$ is $\mathbf{q} = \mathbf{x} + t\mathbf{n}$ is a point on the surface and typically serves as the origin of the tangent frame.

Let $\mathbf{x} = \mathbf{q}$ project onto itself (i.e. $t = 0$) and define a local frame by $\mathbf{q}$ and $\mathbf{n}$. Note that in contrast to the minimization $\mathbf{n}$ is now fixed and it points into the direction of smallest weighted co-variance in $\mathbf{x}$. We construct a locally weighted constant approximation a to the points using in the fixed frame (i.e. a denotes the (constant) height of the fit over the tangent plane) by minimizing

$$\sum_{i=1}^{N} \left(\mathbf{n}^{\mathsf{T}}(\mathbf{p}_i - \mathbf{q} - a\mathbf{n})\right)^2 \theta\left(\|\mathbf{p}_i - \mathbf{q}\|\right), \tag{17}$$

which is solved by a satisfying

$$0 = \sum_{i=1}^{N} 2\left(\mathbf{n}^{\mathsf{T}}(\mathbf{p}_i - \mathbf{q} - a\mathbf{n})\right)\left(\mathbf{n}^{\mathsf{T}}\mathbf{n}\right)\theta\left(\|\mathbf{p}_i - \mathbf{q}\|\right). \tag{18}$$

It seems most authors assume that $a = 0$ has to be the solution because q has been defined so that e_{MLS} is minimized for $t = 0$. Note that this is not necessary because the two minimization functionals differ in the argument to the weight function θ. The partial derivative of e_{MLS} w.r.t. t is

$$\begin{aligned}\frac{\partial e_{MLS}}{\partial t} = -\sum_{i=1}^{N} 2\left(\mathbf{n}^{\mathsf{T}}(\mathbf{p}_i - \mathbf{x} - t\mathbf{n})\right)\mathbf{n}^{\mathsf{T}}\mathbf{n}\theta\left(\|\mathbf{p}_i - \mathbf{x} - t\mathbf{n}\|\right) + \\ \left(\mathbf{n}^{\mathsf{T}}(\mathbf{p}_i - \mathbf{x} - t\mathbf{n})\right)^2 \theta'\left(\|\mathbf{p}_i - \mathbf{x} - t\mathbf{n}\|\right)\frac{\mathbf{n}^{\mathsf{T}}(\mathbf{p}_i - \mathbf{x} - t\mathbf{n})}{\|\mathbf{p}_i - \mathbf{x} - t\mathbf{n}\|}.\end{aligned} \tag{19}$$

This is identical zero for $t = 0$ by our assumptions and $a = 0$ would lead to

$$0 = \sum_{i=1}^{N} \left(\mathbf{n}^{\mathsf{T}}(\mathbf{p}_i - \mathbf{q})\right)^2 \theta'\left(\|\mathbf{p}_i - \mathbf{q}\|\right)\frac{\mathbf{n}^{\mathsf{T}}(\mathbf{p}_i - \mathbf{q})}{\|\mathbf{p}_i - \mathbf{q}\|}, \tag{20}$$

which is not necessary.

For this reason we consider the following two cases for the comparison of MLS surface normals with the gradient of the implicit version:

1. For every $\mathbf{x} = \mathbf{q}$ we find $a = 0$: The weighted average of points is contained in the tangent frame, which is the

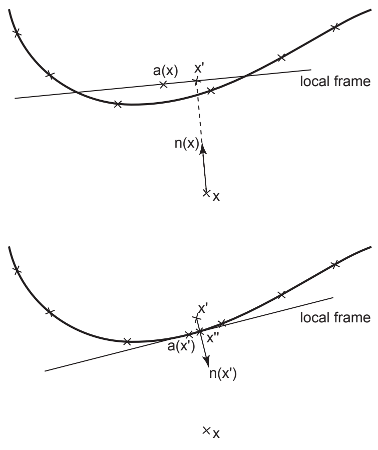

Figure 4: *The basic projection procedure. In each step the current approximation* $\mathbf{x}'$ *is updated by moving in direction* $\mathbf{n}(x')$ *so that* $f(\mathbf{x}') = 0$.

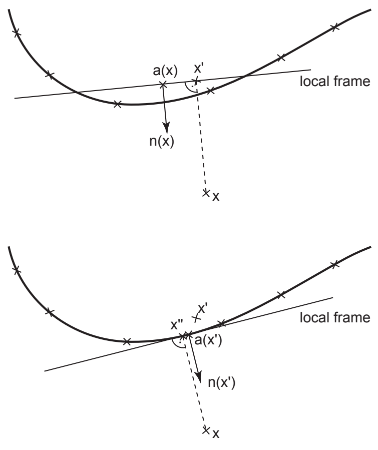

Figure 5: *The first two steps of an 'almost' orthogonal projection of a point* $\mathbf{x}$ *onto the surface. In each step the current approximation* $\mathbf{x}'$ *is used to build an orthogonal tangent frame using* $\mathbf{n}(\mathbf{x}')$ *and* $\mathbf{a}(\mathbf{x}')$*, onto which* $\mathbf{x}$ *is projected to get an new approximation.*

necessary condition for points to be part of the implicit surface as defined above. Thus, the MLS surface is contained in the implicit surface. As we have already shown, the direction of smallest co-variance is not normal to the surface, so this would also be true for the MLS surface.

2. There exist $\mathbf{x} = \mathbf{q}$ for which $a \neq 0$: By definition of the MLS surface, the points $\mathbf{x} = \mathbf{q}$ as well as the points $\mathbf{x}' = \mathbf{q} + a\mathbf{n}$ form a surface. These two surfaces share the same gradient field ($\mathbf{n}$), which cannot be: Assuming $\nabla\mathbf{x} = \nabla\mathbf{q} = \mathbf{n}$ leads to $\nabla x' = \mathbf{n} + a\nabla\mathbf{n} \neq \mathbf{n}$ because (as shown before) $\nabla\mathbf{n}$ is non-zero and a is not everywhere zero by assumption.

4. Projection operators

Let $\mathbf{n}(\mathbf{x})$ and $\mathbf{a}(\mathbf{x})$ define a tangent frame with origin in $\mathbf{a}(\mathbf{x})$ and let the projection of $\mathbf{x}$ onto the tangent be

$$Q(\mathbf{x}) = \mathbf{x} - \mathbf{n}(\mathbf{x})^{\mathsf{T}}(\mathbf{a}(\mathbf{x}) - \mathbf{x})(\mathbf{a}(\mathbf{x}) - \mathbf{x}). \tag{21}$$

The definition of projection operators follows from the following

Observation $Q(\mathbf{x}) = \mathbf{x} \Longleftrightarrow \mathbf{x} \in \mathcal{S}$.

Proof: If $\mathbf{x} \in \mathcal{S}$ then $\mathbf{n}(\mathbf{x})^{\mathsf{T}}(\mathbf{a}(\mathbf{x}) - \mathbf{x}) = 0$ and, thus, $Q(\mathbf{x}) = \mathbf{x}$. On the other hand, if $Q(\mathbf{x}) = \mathbf{x}$, then $n(\mathbf{x})^{\mathsf{T}}(\mathbf{a}(\mathbf{x}) - \mathbf{x})(\mathbf{a}(\mathbf{x}) - \mathbf{x}) = 0$, so that also $n(\mathbf{x})^{\mathsf{T}}(\mathbf{a}(\mathbf{x}) - \mathbf{x}) = 0$ and, thus, $\mathbf{x} \in \mathcal{S}$.

4.1. The basic projection procedure

Consequently, the idea for a projection operation is to repeatedly apply $Q(\mathbf{x})$ to a position in space until $\|Q(\mathbf{x}) - \mathbf{x}\| < \varepsilon$ for a given ε. This idea is illustrated in Figure 4. More specifically, for a given point $\mathbf{x} \in \mathcal{B}$ the following simple procedure yields a projected point $\mathbf{x}'$ on the surface:

1. Compute $\mathbf{a}(\mathbf{x})$ and set $\mathbf{x}' = \mathbf{a}(\mathbf{x})$.
2. Compute $\mathbf{n}(\mathbf{x}')$, $\mathbf{a}(\mathbf{x}')$, and set $\mathbf{x}' = \mathbf{x}' - \mathbf{n}(\mathbf{x}')^{\mathsf{T}}(\mathbf{a}(\mathbf{x}') - \mathbf{x}')(\mathbf{a}(\mathbf{x}') - \mathbf{x}')$
3. If $\|\mathbf{n}(\mathbf{x}')^{\mathsf{T}}(\mathbf{a}(\mathbf{x}') - \mathbf{x}')\| > \varepsilon$ go back to 2.

It is clear that if this iteration converges it yields a point on $\mathcal{S}$. It is interesting to note, however, that even though this iterative procedure minimizes the number of projection steps, the Euclidean distance between $\mathbf{x}$ and its final projection $\mathbf{x}' = Q(\mathbf{x}')$ is not minimized. In other words: the projection is not orthogonal.

4.2. An 'almost' orthogonal projection

We can adapt the projection procedure to make it 'almost' orthogonal. By almost orthogonal we mean that the projection is in direction of $\mathbf{n}(\mathbf{x}')$, if $\mathbf{x}'$ is the projection. There is only one change to the basic procedure: The projection always considers the original point $\mathbf{x}$ and not the intermediate points $\mathbf{x}'$. More specifically, the following procedure computes an orthogonal projection of $\mathbf{x}$:

1. Set $\mathbf{x}' = \mathbf{x}$.
2. Compute $\mathbf{a}(\mathbf{x}')$ and $\mathbf{n}(\mathbf{x}')$.
3. Set $\mathbf{x}' = \mathbf{x} - \mathbf{n}(\mathbf{x}')^{\mathsf{T}}(\mathbf{a}(\mathbf{x}') - \mathbf{x})(\mathbf{a}(\mathbf{x}') - \mathbf{x})$.
4. If $\|\mathbf{n}(\mathbf{x}')^{\mathsf{T}}(\mathbf{a}(\mathbf{x}') - \mathbf{x}')\| > \varepsilon$ go back to 2.

The procedure is illustrated in Figure 5. When terminated $\mathbf{x}'$ is a point on the surface, because $f(\mathbf{x}') = \mathbf{n}(\mathbf{x}')^{\mathsf{T}}(\mathbf{a}(\mathbf{x}') -$

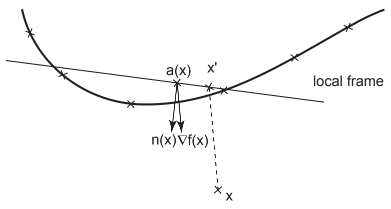

Figure 6: *One step of the orthogonal projection procedure. Orthogonality is achieved in the limit because* $\mathbf{x}$ *is projected in the direction of* $\nabla(f\mathbf{x}')$*. Note that the difference of* $\mathbf{n}(\mathbf{x}')$ *and* $\nabla f(\mathbf{x}')$ *has been exaggerated.*

$\mathbf{x}') \approx 0$. Moreover, since $\mathbf{x}' = \mathbf{x} - \mathbf{n}(\mathbf{x}')^\top(\mathbf{a}(\mathbf{x}') - \mathbf{x})(\mathbf{a}(\mathbf{x}') - \mathbf{x})$ is the orthogonal projection of $\mathbf{x}$ onto the frame defined by $\mathbf{n}(\mathbf{x}')$ and $\mathbf{a}(\mathbf{x}')$, $\mathbf{x}' - \mathbf{x}$ is in direction of $\mathbf{n}(\mathbf{x}')$.

4.3. Computing orthogonal projections

Making the projection orthogonal, i.e. projecting into direction of $\nabla f(\mathbf{x}')$, is slightly more complex. For the above procedure we have used the fact that any point $\mathbf{x}''$ projected onto the frame defined by $\mathbf{n}(\mathbf{x}')$ and $\mathbf{a}(\mathbf{x}')$ yields $\mathbf{n}(\mathbf{x}')^\top(\mathbf{a}(x') - \mathbf{x}'') = 0$ so that in the case of convergence the point is part of the surface. Thus, simply replacing $\mathbf{n}(\mathbf{x}')$ with $\nabla f(\mathbf{x}')$ wouldn't work because then the result would satisfy $\nabla f(\mathbf{x}')(\mathbf{a}(\mathbf{x}') - \mathbf{x}') = 0$, which is not identical to $f(\mathbf{x}') = 0$.

We keep the idea of projecting onto a tangent plane defined by $\mathbf{n}(\mathbf{x}')$ and $\mathbf{a}(\mathbf{x}')$, however, the projection has to be in direction of the gradient of f. Strictly, an intermediate projection $\mathbf{x}''$ should satisfy $\mathbf{n}(\mathbf{x}')^\top(\mathbf{a}(\mathbf{x}') - \mathbf{x}'') = 0$ (i.e. $\mathbf{x}''$ is on the tangent plane) and $\mathbf{x}'' + g\nabla f(\mathbf{x}'') = \mathbf{x}$ (i.e. $\mathbf{x}$ is projected in the direction of the gradient in $\mathbf{x}''$) simultaneously. Thus, we would have to solve

$$\mathbf{n}(\mathbf{x}')^\top \left(\mathbf{a}(\mathbf{x}') - \mathbf{x} + g\nabla f(\mathbf{x}'')\right) = 0 \tag{22}$$

to find the next point in the iteration. This is equation is difficult to solve and might not have a unique solution. We use the same approach as before and assume the situation in the limit, i.e.

$$\mathbf{n}(\mathbf{x}')^\top \left(\mathbf{a}(\mathbf{x}') - \mathbf{x} + g\nabla f(\mathbf{x}')\right) = 0, \tag{23}$$

which means projecting $\mathbf{x}$ onto the tangent frame defined by $\mathbf{n}(\mathbf{x}')$ and $\mathbf{a}(\mathbf{x}')$ in the direction of $\nabla f(\mathbf{x}')$. The step-by-step procedure looks as follows:

1. Set $\mathbf{x}' = \mathbf{x}$.
2. Compute $\mathbf{a}(\mathbf{x}')$, $\mathbf{n}(\mathbf{x}')$, and $\nabla f(\mathbf{x}')$.
3. Compute $g = \frac{\mathbf{n}(\mathbf{x}')^\top(\mathbf{a}(\mathbf{x}') - \mathbf{x})}{\mathbf{n}(\mathbf{x}')^\top \nabla f(\mathbf{x}')}$ to set $\mathbf{x}' = \mathbf{x} - g(\mathbf{a}(\mathbf{x}') - \mathbf{x})$.
4. If $\|\mathbf{n}(\mathbf{x}')^\top(\mathbf{a}(\mathbf{x}') - \mathbf{x}')\| > \varepsilon$ go back to 2.

The illustration in Figure 6 shows the concept by exaggerating the typical deviation between $\mathbf{n}(\mathbf{x})$ and $\nabla f(\mathbf{x})$.

5. Remark on a manifold surface definition

Note that the gradient of f for points on the medial axis is not defined properly. Furthermore, points on the surface, for which $\nabla f = 0$ are non-manifold. With the possibility to compute the gradient exactly one might define the surface simply as:

$$\hat{\mathcal{S}} = \{\mathbf{x} | f(\mathbf{x}) = 0 \wedge \nabla f(\mathbf{x}) \neq \mathbf{0}\} \tag{24}$$

Then, $\hat{\mathcal{S}}$ is necessarily a (collection of) smooth manifolds, possibly with boundary (provided that $\mathbf{n}(\mathbf{x})$ and $\mathbf{a}(\mathbf{x})$ are smooth, as before).

6. Conclusions

We explicitly consider the normals of Point Set Surfaces and its variants. We demonstrate that the normal to the approximating tangent frame is not the surface normal. Based on an implicit version of the surface description we show how to compute exact surface normals. These exact surface normals allow computing orthogonal projections. We feel that these tools help to solidify the computational framework of Point Set Surfaces.

Is is well known that the stationary points of orthogonal projections on surfaces include points on the surface as well as points on the medial axis. This gives rise to the hope that a sampling theory in the spirit of Amenta and co-workers' can be established in the near future.

Acknowledgements

We appreciate the valuable comments by the anonymous reviewers. The early human models in Figure 3 were 3d-digitized by Peter Neugebauer of Polygon Technology Ltd, Darmstadt, Germany using a structured light scanner and the QTSculptor system.

References

[AA03] ADAMSON A., ALEXA M.: Approximating and intersecting surfaces from points. In *Proceedings of the Eurographics Symposium on Geometry Processing* (June 2003), Kobbelt L., Schröder P.,, Hoppe H., (Eds.), pp. 245–254.

[AA04] ADAMSON A., ALEXA M.: Approximating bounded, non-orientable surfaces from points. In *Proceedings of Shape Modeling International 2004* (2004), IEEE Computer Society. accepted for publication.

[ABCO*01] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Point set surfaces. In *IEEE Visualization 2001* (October 2001), pp. 21–28. ISBN 0-7803-7200-x.

[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 Computer Graphics and Visualization 9*, 1 (2003), 3–15.

[ABE98] AMENTA N., BERN M., EPPSTEIN D.: The crust and the beta-skeleton: Combinatorial curve reconstruction. *Graphical Models and Image Processing 60*, 2 (March 1998), 125–135.

[ABK98] AMENTA N., BERN M., KAMVYSSELIS M.: A new voronoi-based surface reconstruction algorithm. *Proceedings of SIGGRAPH 98* (July 1998), 415–422. ISBN 0-89791-999-8. Held in Orlando, Florida.

[ACK01] AMENTA N., CHOI S., KOLLURI R.: The power crust, unions of balls, and the medial axis transform. *CGTA: Computational Geometry: Theory and Applications 19*, 2–3 (2001), 127–153.

[AK04] AMENTA N., KIL Y. J.: Defining point set surfaces. *ACM Transactions on Graphics (SIGGRAPH 2004 issue) 23*, 3 (2004). to appear.

[HJ91] HORN R. A., JOHNSON C. A.: *Topics in Matrix Analysis*. Cambridge University press, Cambridge, 1991.

[KV01] KALAIAH A., VARSHNEY A.: Differential point rendering. In *Rendering Techniques 2001: 12th Eurographics Workshop on Rendering* (June 2001), Eurographics, pp. 139–150. ISBN 3-211-83709-4.

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

[PKKG03] PAULY M., KAISER R., KOBBELT L., GROSS M.: Shape modeling with point-sampled geometry. *ACM Transactions on Graphics (SIGGRAPH 2003 issue) 22*, 3 (2003). to appear.

[PZvBG00] PFISTER H., ZWICKER M., VAN BAAR J., GROSS M.: Surfels: Surface elements as rendering primitives. *Proceedings of SIGGRAPH 2000* (July 2000), 335–342. ISBN 1-58113-208-5.

[RL00] RUSINKIEWICZ S., LEVOY M.: Qsplat: A multiresolution point rendering system for large meshes. *Proceedings of SIGGRAPH 2000* (July 2000), 343–352. ISBN 1-58113-208-5.

[ZPKG02] ZWICKER M., PAULY M., KNOLL O., GROSS M.: Pointshop 3d: An interactive system for point-based surface editing. *ACM Transactions on Graphics 21*, 3 (July 2002), 322–329. ISSN 0730-0301 (Proceedings of ACM SIGGRAPH 2002).

[ZPvBG02] ZWICKER M., PFISTER H., VAN BAAR J., GROSS M.: Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics 8*, 3 (July - September 2002), 223–238. ISSN 1077-2626.

Appendix A: Derivative of the eigenvalue of $\mathbf{W}$

The derivative of the characteristic polynomial $p_{\mathbf{W}}$ of a matrix $\mathbf{W}$ is [HJ91](6.5.10) is

$$\frac{dp_{\mathbf{W}}(\lambda)}{d\lambda} = Tr(Adj(\lambda\mathbf{I} - \mathbf{W})). \tag{25}$$

Now, in our case we take directional derivatives and λ and $\mathbf{W}$ are dependent variables, which means we have apply the chain rule to get

$$\frac{\partial p_{\mathbf{W}}(\lambda)}{\partial \mathbf{e}_k} = Tr\left(Adj(\lambda\mathbf{I} - \mathbf{W})\left(\frac{\partial\lambda}{\partial \mathbf{e}_k}I - \frac{\partial \mathbf{W}}{\partial \mathbf{e}_k}\right)\right). \tag{26}$$

Note that $\mathbf{n}^{\mathsf{T}}$ and $\mathbf{n}$ are the left and right eigenvalues of $\lambda_0\mathbf{I} - \mathbf{W}$ corresponding to eigenvalue 0. Therefore, $(\mathbf{n}\mathbf{n}^{\mathsf{T}})(\lambda_0\mathbf{I} - \mathbf{W}) = 0$ and because $\mathbf{n}\mathbf{n}^{\mathsf{T}}$ is rank one (assuming that λ is single eigenvalue, which we may in our setting) it is the adjugate of $\lambda_0\mathbf{I} - \mathbf{W}$. Further, note that $\mathbf{n}^{\mathsf{T}}\mathbf{n} = 1 \neq 0$.

Taking derivatives on both sides of $p_{\mathbf{W}}(\lambda_0)$ and using the mentioned identities leads to the following derivation

$$\begin{aligned}
0 &= Tr\left(Adj(\lambda\mathbf{I} - \mathbf{W})\left(\frac{\partial\lambda_0}{\partial \mathbf{e}_k}I - \frac{\partial \mathbf{W}}{\partial \mathbf{e}_k}\right)\right)\\
&= Tr\left(\mathbf{n}\mathbf{n}^{\mathsf{T}}\left(\frac{\partial\lambda_0}{\partial \mathbf{e}_k}I - \frac{\partial \mathbf{W}}{\partial \mathbf{e}_k}\right)\right)\\
&= Tr\left(\mathbf{n}\mathbf{n}^{\mathsf{T}}\frac{\partial\lambda_0}{\partial \mathbf{e}_k}I - \mathbf{n}\mathbf{n}^{\mathsf{T}}\frac{\partial \mathbf{W}}{\partial \mathbf{e}_k}\right)\\
&= \mathbf{n}^{\mathsf{T}}\mathbf{n}\frac{\partial\lambda_0}{\partial \mathbf{e}_k} - \mathbf{n}^{\mathsf{T}}\frac{\partial \mathbf{W}}{\partial \mathbf{e}_k}\mathbf{n}\\
&= \frac{\partial\lambda_0}{\partial \mathbf{e}_k} - \mathbf{n}^{\mathsf{T}}\frac{\partial \mathbf{W}}{\partial \mathbf{e}_k}\mathbf{n}.
\end{aligned} \tag{27}$$