Curvature-driven Modeling and Rendering of Point-Based Surfaces

João Paulo Gois[1], Eduardo Tejada[2], Tiago Etiene[1], Luis Gustavo Nonato[1],
Antonio Castelo[1] and Thomas Ertl[2]
[1]University of São Paulo [2]University of Stuttgart
{jpgois,etiene,gnonato,castelo}@icmc.usp.br {tejada,ertl}@vis.uni-stuttgart.de

Abstract

In this work we address the problem of computing point-based surface approximations from point clouds. Our approach is based on recently presented methods that define the approximated surface as the set of stationary points for an operator that projects points in the space onto the surface. We present a novel projection operator that differs from the defined in previous work in that it uses principal curvatures and directions approximation and an anisotropic diffusion equation to ensure an accurate approximation to the surface. We show how to estimate the principal curvatures and directions for point clouds and discuss the usefulness of the curvature information in the context of point-based surface modeling and rendering.

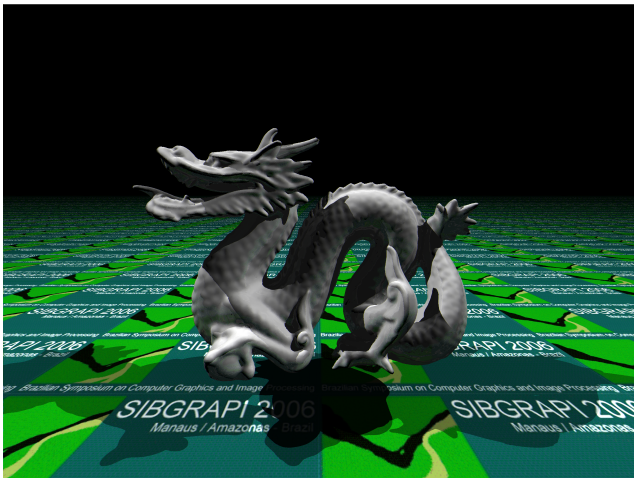

Figure 1. Rendering of the point-based surface approximation for the Dragon dataset obtained with the projection procedure proposed in this work.

1 Introduction

Point-based methods for modeling and rendering surfaces from point clouds have been addressed by various authors lately [15]. Alexa et al. [5, 6] and Zwicker et al. [24] presented point-based surface approximation methods that define the approximated surface as the set of stationary points for an operator that projects points in the space onto the surface. Both approaches approximate the surface locally by finding a polynomial approximation to the surface on a local orthonormal coordinate system near the point to be projected. Zwicker et al. made use of the *weighted least-squares* method both for finding the local coordinate system and the local polynomial approximation to the surface. On the other hand, Alexa et al. used *moving least-squares* to find the local coordinate system and weighted least-squares to calculate the polynomial approximation. These approaches present a number of interesting features, such as the fact that the approximated surfaces are meshless and low-frequency noise free.

We propose a novel projection operator for computing point-based surfaces based on the approximation of principal curvatures and directions and an anisotropic diffusion equation. We use the curvature information to define the local polynomial as a non-complete quadratic function without loosing accuracy in the local approximation. The anisotropic diffusion equation is used to define the point on the surface where such quadratic polynomial is calculated. The anisotropy in this equation helps us preserve the geometry of the original surface since this equation is based on directional curvatures. Therefore, we show how to estimate the curvature at any point on the surface, which is a further contribution of our approach since the curvature is useful for the analysis of both quantitative and qualitative characteristics of the surface [23]. For this, we modified the curvature estimation method by Huang and Menq [13] to introduce weights in order to improve the robustness of the method. This weights were carefully constructed and are specific to our problem.

The fact that the local approximation is defined as a non-complete quadratic polynomial can help decrease the pro-

cessing time of algorithms that perform intersection computations such as ray-tracing. For rendering the surfaces defined by our operator, we use the ray-tracing algorithm for point-based surface developed by Adamson and Alexa [2]. We describe how the intersection computation must be modified to fit our projection operator and use this ray-tracer to demonstrate the quality of the approximations obtained (see Figure 1 for an example). We compare the results obtained using the method presented with the results obtained using moving least-squares as in the work by Alexa et al.

2 Related work

As mentioned before, Alexa et al. [5, 6] and Zwicker et al. [24] developed approaches based on projection operators for generating point-based approximated surfaces from point clouds. Based on this work, Amenta and Kil performed an analysis of both the weighted and the moving least-squares strategies, presenting interesting results about their domain [8]. Such results were used by Tejada et al. [22] to define a 'predictor-corrector' projection scheme for extracting point-based surfaces from volumetric data. Amenta and Kil also proposed an alternative approach based on an integral formulation [8]. Adamson and Alexa [1] defined an implicit function for approximating surfaces from point clouds based on the work by Alexa et al. [5].

Although these methods are able to deal with low frequency noise, they can fail when high frequency noise and outliers are present in the data. Fleishman et al. [10] proposed a robust point-based strategy based on the statistical framework known as *forward-search*. Such strategy is able to deal with high frequency noise and to detect sharp corners. Reuter et al. [19] presented a method based on the *enriched reproducing kernel particle approximation* for preserving sharp corners with equally good results as the obtained by Fleishman et al. Although both methods work well even for noisy data, they have a high computational cost and require user interference.

Curvatures and principal directions estimation for regular grids and polygonal meshes has been extensively studied in both the qualitative and the quantitative cases. Maltred and Daniel [17] presented a survey on classical work on curvature estimation and a classification based on the requirements and constraints of the methods described. Although there are many methods for estimating principal curvatures and directions, almost all of them need a mesh. Only recently, effective methods to estimate curvature information directly from point sets were presented [4, 23, 16, 13].

Tong and Tang [23] presented a robust curvature estimation method based on adaptive curvature tensors by means of tensor voting. In addition, they presented an analytical comparison with classical and efficient methods with respect to their input (point clouds or mesh models), their requirements (geometrical measures) and their outputs (quantitative or qualitative estimations). Lange and Polthier [16] adapted the well known mesh-oriented method by Taubin [21] to the point cloud context. In their work, the authors used such adapted method together with an anisotropic diffusion equation for removing noise from point clouds without smoothing sharp corners.

Huang and Menq [13] proposed a curvature estimation method built upon the least-squares scheme and the Euler's theorem from differential geometry. Although the method was proposed to locally optimize unstructured surface meshes, it is also suitable for point clouds. We derive our method from this work, since it can be easily adapted to support weighting functions, which is an important condition for the quality of the models generated by point-based techniques.

3 Weighted and moving least-squares

Traditionally, authors describe point-based techniques by means of minimizations of squared sums [24, 6, 19]. We, on the other hand, opted for describing our method based on the normal equation approach [20]. To that end, we describe first the traditional formulation of weighted and moving least-squares for polynomial approximation for sake of understanding and continuity in the description.

Let us consider a finite set of n points $\mathcal{P} = \{p_i = (x_i, y_i, z_i) : p_i \in \mathbb{R}^3\}$. We aim at building a function f that best approximates $\mathcal{P}$. The standard least-squares for solving such task minimizes the following overdetermined system:

$$[\mathbf{B}][\mathbf{C}] = [\mathbf{Z}] \Longleftrightarrow \begin{pmatrix} b(\overline{p_1}) \\ b(\overline{p_2}) \\ \vdots \\ b(\overline{p_n}) \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_m \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix}, \quad (1)$$

where $b(\overline{p_i})$ is the application of $\overline{p_i} = (x_i, y_i)$ on a basis of functions b and c_i are the coefficients of the polynomial approximation determined in function of z_i. For instance, if we want to approximate $\mathcal{P}$ by a plane, we choose $b = [1\ x\ y]$. On the other hand, if we want to approximate $\mathcal{P}$ by a complete quadratic polynomial, we use $b = [1\ x\ x^2\ y\ y^2\ xy]$.

Both the weighted and the moving least-squares methods are formulated by multiplying a weighting matrix to both sides of System 1 in order to force some points to have more influence on the solution. Let us consider the weighting

matrix build upon the set $\mathcal{P}$:

$$[\mathbf{W}] = \begin{pmatrix} w(p,p_1) & 0 & \cdots & 0 \\ 0 & w(p,p_2) & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & w(p,p_n) \end{pmatrix}, \quad (2)$$

where we choose w as a monotonically decreasing function such that $w(p,p_i) \to 0$ when $||p - p_i|| \to \infty$. Thus, points near p have more influence than distant points. The overdetermined system with weights becomes:

$$[\mathbf{W}][\mathbf{B}][\mathbf{C}] = [\mathbf{W}][\mathbf{Z}]. \quad (3)$$

Therefore, the normal equation, which is the one that minimizes the sum of the squared differences between the left and right sides of System 3, is given by:

$$\begin{aligned} ([\mathbf{W}][\mathbf{B}])^t[\mathbf{W}][\mathbf{B}][\mathbf{C}] &= ([\mathbf{W}][\mathbf{B}])^t[\mathbf{W}][\mathbf{Z}] \iff \\ \iff [\mathbf{B}]^t[\mathbf{W}]^2[\mathbf{B}][\mathbf{C}] &= [\mathbf{B}]^t[\mathbf{W}]^2[\mathbf{Z}]. \end{aligned} \quad (4)$$

The main mathematical difference between weighted least-squares and moving least-squares is basically due to p. In the case of point-based surfaces approximated using least-squares methods, if p is an estimated fixed point, System 4 becomes linear and the problem turns into a weighted least-squares formulation where unicity of the solution can be ensured. On the other hand, if p is unknown (as well as the coefficients in $[\mathbf{C}]$), System 4 becomes non-linear turning into a moving least-squares formulation where unicity of the solution cannot be longer ensured.

The least-square scheme to estimate principal directions and curvatures presented in the next section is developed using also the normal equation. However, our formulation is built upon Euler's theorem and the weighting function is no longer monotonically decreasing.

4 Principal directions and curvatures computation

The directional curvatures at a point p on a smooth surface $\mathcal{M}$ are defined in terms of the curvatures of smooth curves on $\mathcal{M}$ containing p. The minimum and maximum directional curvatures computed from the curves are called *principal curvatures* and their respective directions are called *principal directions*. One important result is that principal directions are orthogonal at p [9].

The Euler's theorem from differential geometry states that every directional curvature in $p \in \mathcal{M}$ can be described as a function of its principal directions and curvatures [9]. More formally, let us define the principal directions and curvatures at p as ν_1^p and ν_2^p, and κ_1^p and κ_2^p respectively. The Euler's theorem states that the curvature at p in the direction μ is given by:

$$\kappa^p(\mu) = \kappa_1^p \cos^2(\alpha) + \kappa_2^p \sin^2(\alpha), \quad (5)$$

where $\mu = \cos(\alpha)\nu_1^p + \sin(\alpha)\nu_2^p$.

Let us consider the approximated tangent plane $\mathcal{T}$ in p, where we define a local orthonormal coordinate system $\{t_1, t_2\}$ with origin in p (Figure 2). We compute the projections $\tilde{p_i}$ on $\mathcal{T}$ of the neighbors p_i of p, defining the directions ν^i. By sorting such directions counterclockwisely, we obtain the angles θ_i from t_1 to $\tilde{p_i}$. Let us also denote by β_i the counterclockwise angle from the principal direction ν_1^p to ν^i.

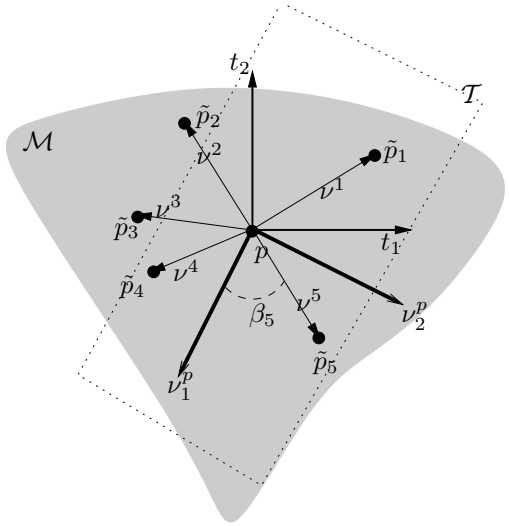

Figure 2. Estimating directional curvatures on the approximated tangent plane at p.

We can approximate the directional curvature by:

$$k^p(\nu^i) \approx \frac{2 < n_p, \varrho^i >}{< \varrho^i, \varrho^i >}, \quad (6)$$

where n_p is the normal vector at p and $\varrho^i = p_i - p$ for p_i in the neighborhood of p. This discrete directional curvature has been used in several other methods for curvature estimation [13, 16, 21, 23]. The following nonlinear overdetermined system is obtained from the Euler's theorem and the directional curvatures approximation (Equation 6):

$$\begin{pmatrix} \cos^2(\beta_1) & \sin^2(\beta_1) \\ \cos^2(\lambda_2) & \sin^2(\lambda_2) \\ \vdots & \vdots \\ \cos^2(\lambda_m) & \sin^2(\lambda_m) \end{pmatrix} \begin{pmatrix} \kappa_1^p \\ \kappa_2^p \end{pmatrix} = \begin{pmatrix} \kappa^p(\nu^1) \\ \kappa^p(\nu^2) \\ \vdots \\ \kappa^p(\nu^m) \end{pmatrix}, \quad (7)$$

where $\lambda_i = \beta_1 + \delta\beta_i$ and $\delta\beta_i = \beta_i - \beta_1 = \theta_i - \theta_1$.

In order to obtain a linear system, we follow the work by Huang and Menq [13] and set $\gamma_1 = \frac{\kappa_1^p + \kappa_2^p}{2}$, $\gamma_2 = -\cos(2\beta_1)(k_2^p - k_1^p)$ and $\gamma_3 = \sin(2\beta_1)(k_2^p - k_1^p)$, producing the following overdetermined linear system:

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & \cos(2\delta\beta_2) & \sin(2\delta\beta_2) \\ \vdots & \vdots & \vdots \\ 1 & \cos(2\delta\beta_m) & \sin(2\delta\beta_m) \end{pmatrix} \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{pmatrix} = \begin{pmatrix} \kappa^p(\nu^1) \\ \kappa^p(\nu^2) \\ \vdots \\ \kappa^p(\nu^m) \end{pmatrix}. \tag{8}$$

Therefore, the normal equation for System 8 becomes:

$$\begin{pmatrix} m & \sum_{i=1}^{m} c_i & \sum_{i=1}^{m} s_i \\ \sum_{i=1}^{m} c_i & \sum_{i=1}^{m} (c_i)^2 & \sum_{i=1}^{m} c_i s_i \\ \sum_{i=1}^{m} s_i & \sum_{i=1}^{m} c_i s_i & \sum_{i=1}^{m} (s_i)^2 \end{pmatrix} \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{m} \hat{\kappa}_i \\ \sum_{i=1}^{m} \hat{\kappa}_i c_i \\ \sum_{i=1}^{m} \hat{\kappa}_i s_i \end{pmatrix}, \tag{9}$$

where $c_i = \cos(2\delta\beta_i)$, $s_i = \sin(2\delta\beta_i)$ and $\hat{\kappa}_i = \kappa^p(\nu^i)$. Thus, the principal directions and curvatures are straightforwardly obtained from γ_i, $1 \leq i \leq 3$.

Although Huang and Menq state that the method is robust for noisy data, we only achieved robustness for our problem by adding suitable weights. To that end, we defined the weights $\mathbf{w}_i = (\mathbf{w}(||p_i - p||))^2$, where $\mathbf{w}$ is an "M"-like function. For our problem we used:

$$\mathbf{w}(x) = x^{2\varsigma} e^{-\frac{x^2}{h^2}}, \tag{10}$$

where $\varsigma \in \mathbb{N}^*$ and h is a smoothing parameter [6]. That way, we modified the normal equation as in the following expression:

$$\begin{pmatrix} \sum_{i=1}^{m} \mathbf{w}_i & \sum_{i=1}^{m} \mathbf{w}_i c_i & \sum_{i=1}^{m} \mathbf{w}_i s_i \\ \sum_{i=1}^{m} \mathbf{w}_i c_i & \sum_{i=1}^{m} \mathbf{w}_i (c_i)^2 & \sum_{i=1}^{m} \mathbf{w}_i c_i s_i \\ \sum_{i=1}^{m} \mathbf{w}_i s_i & \sum_{i=1}^{m} \mathbf{w}_i c_i s_i & \sum_{i=1}^{m} \mathbf{w}_i (s_i)^2 \end{pmatrix} \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{m} \mathbf{w}_i \hat{\kappa}_i \\ \sum_{i=1}^{m} \mathbf{w}_i \hat{\kappa}_i c_i \\ \sum_{i=1}^{m} \mathbf{w}_i \hat{\kappa}_i s_i \end{pmatrix}. \tag{11}$$

The use of an "M"-like function is very important to obtain good results with our method. This is due to the fact that the directional curvatures obtained by Equation 6 are dependent on the position of the points and their normal vectors [13]. Figure 3 depicts an example showing how a small perturbation in the position of point p, represented by a square, can produce a quite different solution with the original method by Huang.

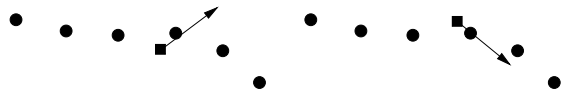

Figure 3. Results from the method by Huang and Menq [13]. A small perturbation in the position of the point can produce considerably different curvature and direction approximations (represented by the arrows).

In the "M"-like function, $x^{2\varsigma}$ controls the influence of the points close to p on the solution. The larger ς is, the smaller the region around p which will have significant influence in the solution is. The exponential member of the function maintains the same behavior of the traditional Gaussian function [7]. We present examples of graphs of this function in Figure 4 with different parameter values.

Unlike a Gaussian weight, usually employed in previous work [5, 6, 24, 1, 7], the "M"-like function is able to solve the problem depicted in Figure 3. This will be shown in the next section after the description of the surface approximation procedure.

5 Projection and rendering procedures

As mentioned before, traditional point-based surface approximation techniques define the approximated surface for a given point cloud as the set of stationary points for a carefully designed projection operator. This way, the input point cloud can be re-sampled by projecting a sufficiently large number of points from its neighborhood onto the approximated surface. With this procedure a dense sampling that covers the image space consistently can be obtained.

In this section we describe a new projection operator derived from the theory discussed in the previous section on the estimation of principal directions and curvatures by means of weighted least-squares. For this, we make use of a result from differential geometry which states that a surface can be defined locally (in the neighborhood of p) by:

$$g_\kappa(\xi, \eta) = \frac{1}{2}\left(\kappa_1^p \xi^2 + \kappa_2^p \eta^2\right), \tag{12}$$

where (ξ, η) is in the local coordinate system defined by the principal directions at p. However, this local approximation is valid only for points on the surface. This fact does not

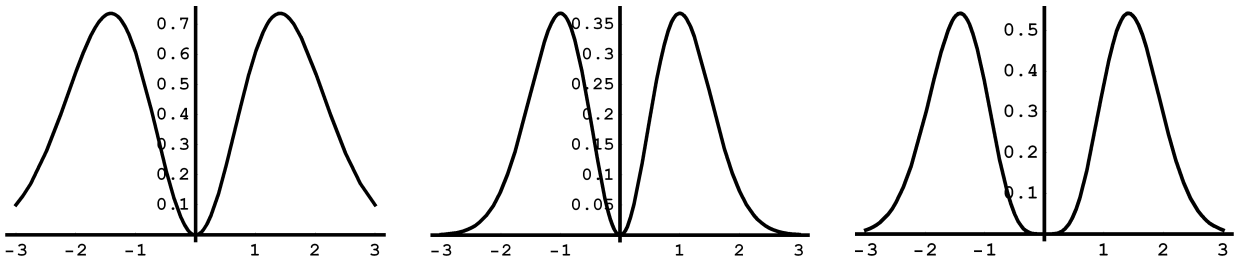

Figure 4. Examples of "M"-like function graphs. From left to right: $h = 2$ and $\varsigma = 1$, $h = 1$ and $\varsigma = 1$, and $h = 1$ and $\varsigma = 2$.

allow us to define a polynomial at a point near (but not on) the surface as done in previous work [6, 24]. In order to avoid this problem, we use a scheme to move points near the surface onto it. The scheme we use for this was proposed by Lange and Polthier [16] for removing noise by moving the points in a point cloud. The authors make use of an anisotropic diffusion equation, which is useful to preserve sharp corners.

Let us consider the diffusion equation:

$$\frac{\partial \mathbf{p}}{\partial t} = \lambda \Delta \mathbf{p}, \tag{13}$$

where $\Delta \mathbf{p} = \left(\frac{\partial^2 \mathbf{p}}{\partial x^2} + \frac{\partial^2 \mathbf{p}}{\partial y^2} + \frac{\partial^2 \mathbf{p}}{\partial z^2} \right)$ is the Laplacian of $\mathbf{p}$, λ is the diffusive term and $\mathbf{p}$ is the position of p at time t [12]. We approximate the Laplacian by the umbrella operator:

$$\tilde{\Delta}\mathbf{p} = \frac{1}{\Omega} \sum_{i}^{m} \omega_i \cdot (p_i - \mathbf{p}), \tag{14}$$

where $\omega_i = \frac{1}{||p_i - \mathbf{p}||^2}$ and $\Omega = \sum \omega_i$.

The explicit forward Euler method for discretizing Equation 13 leads to the traditional iterative Gaussian formula for smoothing data:

$$\mathbf{p}^{n+1} = \mathbf{p}^n + \lambda \delta t \tilde{\Delta} \mathbf{p}^n. \tag{15}$$

It must be observed that $\lambda \delta t$ must satisfy the time step conditions [12]. Lange and Polthier modified the traditional umbrella operator to obtain an anisotropic operator by introducing a suitable real function which offers information related to the shape of the object. This operator is able to move a point onto the surface fairly. The anisotropic Laplacian becomes:

$$\tilde{\Delta}_{\Lambda} \mathbf{p}^n = \frac{1}{\Omega} \sum_{i=1}^{m} \Lambda_i \cdot (p_i - \mathbf{p}^n), \tag{16}$$

where Λ_i is a real function which depends on the directional curvatures (Equation 6) at $\mathbf{p}^n$. Lange and Polthier argued for the use of one of the following functions for a given threshold ε:

$$\Lambda_i = \begin{cases} 1, & \text{if } |\kappa^{\mathbf{p}^n}(\nu^i)| < \varepsilon \\ 0, & \text{if } |\kappa^{\mathbf{p}^n}(\nu^i)| \geq \varepsilon; \end{cases} \tag{17}$$

$$\Lambda_i = \begin{cases} 1, & \text{if } |\kappa^{\mathbf{p}^n}(\nu^i)| < \varepsilon \\ \frac{\lambda^2}{\lambda^2 + 10(|\kappa^{\mathbf{p}^n}(\nu^i)| - \lambda)^2}, & \text{if } |\kappa^{\mathbf{p}^n}(\nu^i)| \geq \varepsilon. \end{cases} \tag{18}$$

Note that the computational cost increase is not significant since the point is already close to the surface, which makes the convergence to the surface fast. Although this step seems to be negligible, it is very important to ensure accurate solutions.

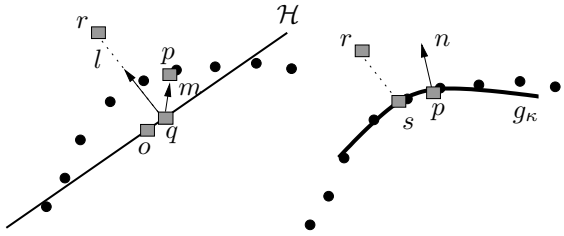

Figure 5. Projection scheme based on principal directions and curvatures.

With this framework, the projection s of a point r onto the surface can be calculated using the following process (see Figure 5):

1. Find a quasi-tangent plane $\mathcal{H}$ with origin o and normal l, where o is the weighted mean of the neighbors of r and l is the weighted mean of the normals at the neighbors of r.
2. Find the projection q of r onto $\mathcal{H}$.
3. Calculate the normal m at q.
4. Find a point p on the surface by moving q in the direction of m using the anisotropic diffusion equation.

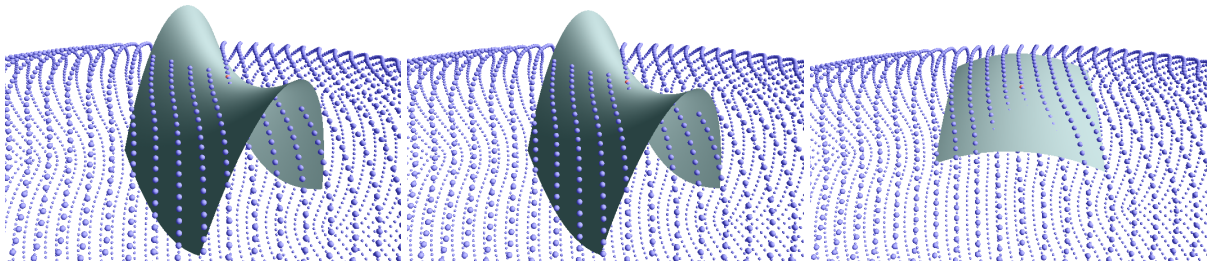

Figure 6. Polynomial approximations obtained (from left to right) without weights, with Gaussian weights and with the "M"-like function.

5. Calculate the normal n and the principal directions ν_1^p, ν_2^p and curvatures κ_1^p, κ_2^p at p.

6. Define (ν_1^p, ν_2^p, n) as a local coordinate system and $g_\kappa(\xi, \eta) = \frac{1}{2}(\kappa_1^p \xi^2 + \kappa_2^p \eta^2)$ as the local polynomial approximation to the surface.

7. Transform r to the local coordinate system and find its projection s_t onto g_κ.

8. The projected point s is the point s_t in the global coordinate system.

The approximated surface is defined as the set of stationary points for the projection process described above. To render the approximated surface, we use the ray-tracing algorithm proposed by Adamson and Alexa [2], changing only the projection procedure. Adamson and Alexa defined a set of enclosing spheres centered at the sample points. The union of these spheres encloses the approximated surface completely. The spheres are used to find a first approximation to the intersection between the surface and the ray. This approximated intersection is then used in an iterative process that projects the intersection onto the surface and calculates the new approximated intersection as the intersection between the ray and the local polynomial approximation. If this new intersection lies outside the current enclosing sphere, the next nearest sphere is tested. The process stops when the distance between the current approximated intersection and its projection is smaller than a pre-defined threshold.

A good estimative of the principal curvatures (κ_1^p, κ_2^p) and directions (ν_1^p, ν_2^p) is of major importance to obtain a good local approximation to the surface (Equation 12). As claimed in the previous section, only by introducing suitable weights into the curvature estimation we can obtain a robust approximation. Figure 6 shows the effect of introducing weights into System 11. As can be seen, a Gaussian weight does not solve the problem. Therefore, we use the "M"-like function presented in the previous section. Note, however, that the "M"-like function is used only for the curvature estimation, whilst a Gaussian weight is used for the rest of the process.

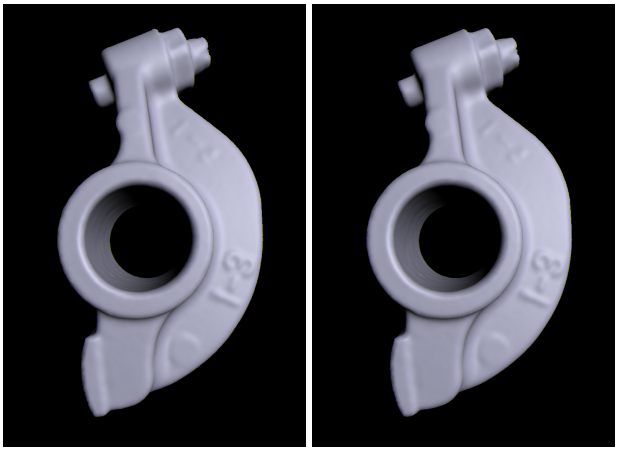

Figure 7. Ray-tracing of the approximated surface for the Rocker Arm dataset obtained with the moving least-squares-based (left) and the curvature-driven (right) methods.

6 Results and discussion

We implemented the moving least-squares-based surface approximation method proposed by Alexa et al. [6] to compare the results obtained with their method and the method we present. Our goal was to be able to obtain a comparable approximation to the surface using a reduced polynomial obtained from the curvature information. Therefore, we used complete quadratic polynomials in our implementation of Alexa's method. In Figure 7 we show the results of both the moving least-squares-based approximation and the curvature-driven approximation for the Rocker Arm dataset.

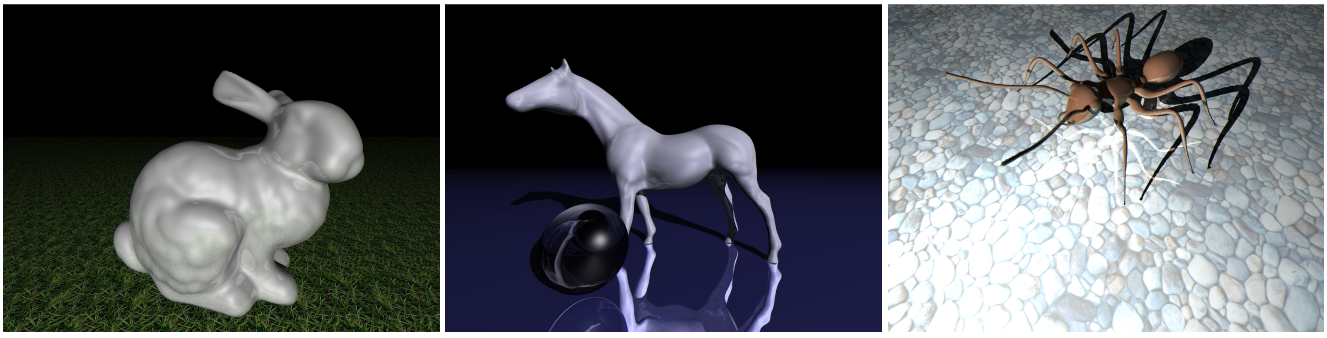

Figure 8. Renderings of the Stanford Bunny, Horse and Ant datasets.

In both cases, we used the ray-tracing algorithm by Adamson and Alexa [2] to render the approximated surface modified to use a kd-tree to accelerate the rendering. As can be seen in the figure, the results are equally good, however our method is between 1.5 to 2.5 times faster. For these performance tests, the models were rendered with no reflection or refraction effects to a 533×400 viewport. More complex scenes were also rendered and are shown in Figures 1 and 8.

This difference in the processing time is due to the fact that, although our method makes use of trigonometric functions to construct the normal equation, it only needs to solve a linear system with three unknowns. This linear system can be solved using closed formulations. Although our methods estimates the normals at three different positions during point projection, this does not represent a bottleneck since we used weighted normals instead of estimating them using covariance analysis.

Although a complete quadratic polynomial can better approximate a larger neighborhood than the non-complete quadratic polynomial we use, for the local computations involved in our method this latter quadratic polynomial approximate the surface accurately. Another important advantage of our method is the availability of an explicit characterization of the point-based surface by means of the curvature information. Also, in practical terms, when computing the local approximation, we discard points for which the "M"-function has values lower than a threshold. This minimizes the computational cost since we perform less trigonometrical operations. Additionally, it reduces the possibility of numerical instability during the local approximation computation.

7 Conclusion

As mentioned before, the use of a non-complete quadratic polynomial for the polynomial approximations simplified the ray-surface intersection computation. However, it would be desirable to analyze the performance impact of the principal directions and curvature estimation process and of the anisotropic diffuse equation (although we found that this latter process needs few iterations to converge). We plan to introduce acceleration techniques into the implementation of the ray-tracing algorithm in order to be able to exploit the simplicity of the intersection computation.

Besides the potential gain in computation time, the availability of curvature information is important for a number of applications [23]. This is a clear advantage of our method over previous point-based surface approximation techniques. There are few works in the literature for curvature estimation from cloud of points [13, 23, 16] and so far (to the extent of our knowledge) there is no work presenting comparisons among these methods. A mathematical and computational efficiency study of such methods must be performed.

Although, with our method local characterization of the surface offered by the principal directions and curvatures is available, it could be of interest to define global properties, e.g. the Euler characteristic. For such tasks, we are currently studying means for incorporating Morse Theory [18] and Computational Homology [14] into our method. This information can be useful for elastic simulations and crack propagation problems [11]. We also intend to use the anisotropic point-based method by Adamson and Alexa [3] due to the fact that weights can be defined for each sample independently with their approach. For this, we are working on defining such weights based on the principal curvatures. Also, since the curvatures produce local information about the object, they can be used for identifying sharp corners. We intend to exploit this fact to propose a new data structure that will support the modeling of sharp corners.

Acknowledgments

This work was partially supported by the German International Exchange Service (DAAD), with Grants A/04/08711 and A/06/04969, and the State of São Paulo Research Foundation (FAPESP), with Grant 04/10947-6.

References

[1] A. Adamson and M. Alexa. Approximating and intersecting surfaces from points. In *Proc. of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 230–239. Eurographics Assoc., 2003.

[2] A. Adamson and M. Alexa. Ray tracing point set surface. In *Proc. of Shape Modeling International*, pages 272–282, 299. IEEE Computer Society, 2003.

[3] A. Adamson and M. Alexa. Anisotropic point set surfaces. In *Proc. of the International Conference on Virtual Reality, Computer Graphics, Visualisation and Interaction in Africa*, pages 7–13. ACM Press, 2006.

[4] G. Agam and X. Tang. A sampling framework for accurate curvature estimation in discrete surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 11(5):573–583, 2005.

[5] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, 2003.

[6] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. In *Proc. of IEEE Visualization*, pages 21–28. IEEE Computer Society, 2001.

[7] N. Amenta and Y. J. Kil. Defining point-set surfaces. *ACM Transactions on Graphics*, 23(3):264–270, 2004.

[8] N. Amenta and Y. J. Kil. The domain of a point set surfaces. In *Eurographics Symposium on Point-based Graphics*, pages 139–147. Eurographics Association, 2004.

[9] M. P. D. Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976.

[10] S. Fleishman, D. Cohen-Or, and C. T. Silva. Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics*, 24(3):544–552, 2005.

[11] X. Guo, X. Li, Y. Bao, X. Gu, and H. Qin. Meshless thin-shell simulation based on global conformal parameterization. *IEEE Transactions on Visualization and Computer Graphics*, 12(3):375–385, 2006.

[12] C. Hirsch. *Numerical Computational of Internal and External Flows*, volume 1. A Wiley-Interscience Publication, 1989.

[13] J. Huang and C. H. Menq. Combinatorial manifold reconstruction and optimization from unorganized point cloud with arbitrary topology. *Computer-Aided Design*, 1(34):149–165, 2002.

[14] T. Kaczyrski, K. Mischaikow, and M. Mrozek. *Computational Homology*. Springer, 2004.

[15] L. Kobbelt and M. Botsch. A survey of point-based techniques in computer graphics. *Computers & Graphics*, 28(6):801–814, 2004.

[16] C. Lange and K. Polthier. Anisotropic smoothing of point sets. *Comput. Aided Geom. Des.*, 22(7):680–692, 2005.

[17] J.-L. Maltret and M. Daniel. Discrete curvatures and applications : a survey. Tech. Report LSIS.RR.2002.002, Laboratoire des Sciences de l'Information et des Systèmes, 2002.

[18] Y. Matsumoto. *An Introduction to Morse Theory*. Amer. Math. Soc., 2002.

[19] P. Reuter, P. Joyot, J. Trunzler, T. Boubekeur, and C. Schlick. Surface reconstruction with enriched reproducing kernel particle approximation. In *Eurographics Symposium on Point-Based Graphics*, pages 79–87. Eurographics Assoc., 2005.

[20] C. Shen, J. F. O'Brien, and J. R. Shewchuk. Interpolating and approximating implicit surfaces from polygon soup. *ACM Transactions on Graphics*, 23(3):896–904, 2004.

[21] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision*, pages 902–907. IEEE Computer Society, 1995.

[22] E. Tejada, J. Gois, L. Nonato, A. Castelo, and T. Ertl. Hardware-accelerated Extraction and Rendering of Point Set Surfaces. In *Proceedings of EUROGRAPHICS - IEEE VGTC Symposium on Visualization*, pages 21–28, 2006.

[23] W.-S. Tong and C.-K. Tang. Robust estimation of adaptive tensors of curvature by tensor voting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(3):434–449, 2005.

[24] M. Zwicker, M. Pauly, O. Knoll, and M. Gross. Pointshop 3D: an interactive system for point-based surface editing. In *SIGGRAPH : Proc. of Computer Graphics and Interactive Techniques*, pages 322–329. ACM Press, 2002.