A new projection method for point set surfaces

Thomas Kalbe[1], Simon Fuhrmann[1], Stefan Uhrig[2], Frank Zeilfelder[3] and Arjan Kuijper[1]

[1]Technische Universität Darmstadt, Germany. firstname.lastname@gris.informatik.tu-darmstadt.de
[2] stefanuhrig@gmx.net [3] frank.zeilfelder@gmx.de

Abstract

A successful approach in triangulating point set surfaces is to apply operations, like a projection operator for advancing front algorithms, directly to Moving-Least Squares (MLS) surfaces. The MLS method naturally handles noisy input data and is especially useful for point clouds derived from real-world solids. Unfortunately, MLS is computationally extensive and complex. We present a novel projection method that does not require solving a non-linear optimization problem as MLS does. We create a polynomial approximation of the surface similar to MLS but our method adapts the degree of the polynomial with respect to the points to be approximated. The approximated points are iteratively collected compromising connectivity information. We enhance the orientation of the local coordinate system to further improve the method. The results confirm that our method is more robust and also accelerates triangulation due to a preprocessing step that needs to be done only once per data set.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Mesh Generation—.

1. Introduction

Various approaches that reconstruct a continuous surface from its discrete representation as point clouds are known. A common and versatile continuous surface representation are meshes. They are efficient in memory consumption and can be processed with high performance on current graphics hardware. Therefore, the construction of meshes has received a lot of attention and a substantial amount of literature, algorithms and techniques on this topic has been published. The generation of a mesh aims at finding a partition of the surface domain which is, in our case, given in form of a discrete representation as unstructured point cloud $\mathcal{P}$. The elements of the partition are typically triangles or quads. The notion of quality of the resulting mesh has several meanings and can refer to the sampling rate, regularity, size or shape of the elements. To obtain a triangulation from some surface representation one can choose between several approaches. We give a brief overview of this related work in Sect. 2.

A certain type of method depends on a projection which maps newly created vertices onto the surface. These projective methods are especially well suited for advancing front algorithms which allow for generation of high-quality meshes that adapt to surface curvature. A very popular projective method is the Moving-Least Squares (MLS) approach for surface reconstruction. The power of MLS surfaces is the ability to naturally cope with input noise in the data. Input noise is derived from the fact that most point clouds are obtained through a three dimensional scanning device to create an initial representation of the physical solid.

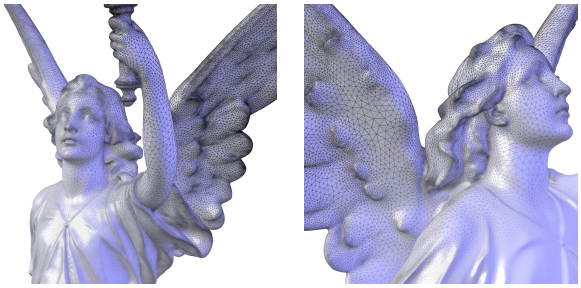

Figure 1: Triangulation of the *Lucy* model with our method.

However, we show that the MLS approach has some deficiencies and can lead to problems in the computation of both the tangent plane and the polynomial approximation of the surface. Additionally points that are beyond the vicinity of the surface are not guaranteed to be correctly projected with MLS. We propose a new projection method that does not

involve a non-linear optimization or similar problem. Furthermore, our method can project points which have an arbitrary distance to $\mathcal{P}$. In a first step we find a local parameter domain; this is closely related to the Cocone [ACDL00] algorithm. In a second step we approximate the surface by a polynomial. Unlike MLS, we do not use a fixed polynomial degree but adapt the degree depending on the location of the points to be approximated.

2. Related work

Some authors proposed to construct an implicit function from $\mathcal{P}$ that partitions the space into interior and exterior [HDD*92, Kol05]. The surface is then defined by the function's kernel [BW97]. As a result these functions nicely close unintended gaps in the point cloud but are not able to reconstruct surfaces with borders. The final surface is often extracted as a triangular mesh by applying marching cubes. However, this usually leads to aliasing or oversampling and badly shaped triangles. Computational geometry approaches such as Alpha Shapes [EM92, BB97], (Power) Crust [ABK98, ACK01] and Cocone [ACDL00] usually apply a Delaunay triangulation as a basic step. Although these methods often provide theoretical reconstruction guarantees, a common problem is the direct reflection of measurement noise. The number of generated vertices and triangles cannot be directly controlled but is related to the cardinality of $\mathcal{P}$.

Advancing front algorithms [Har98, SFS05] successively construct a triangulation from the border of the triangulated domain (the *front*), starting with an initial triangle. New points are predicted and then projected onto a locally reconstructed surface. This can be done with e.g. the MLS method [Lev98, ABCO*03]. In a first step the MLS method calculates the tangent plane of the surface near the point that is to be projected. Next, the surface is locally approximated with a polynomial in the parameter domain defined by the tangent plane. Unfortunately, finding this plane involves a non-linear optimization problem that can only be solved numerically. Scheidegger et al. [SFS05] propose a heuristic to find an appropriate initial value for the gradient descend solver. To be able to robustly solve the involved optimization problem the point to be projected must be within the vicinity of $\mathcal{P}$. Furthermore, as pointed out by [AK04], the correct solution of the optimization problem does not always lead to a tangent plane of the surface but can significantly differ from the correct plane. Calculating the polynomial can lead to similar problems if different regions of the surface approach each other. As a result, points whose Euclidean distance is small are taken into consideration even though their Riemannian distance is large.

We aim at curing the situation described above with an improved projection method that does not involve solving a non-linear optimization problem. The method makes use of connectivity information and uses a variable degree for the polynomials.

3. Our new projection

The problems with the MLS approach are caused by missing connectivity information on $\mathcal{P}$. With this information it is easier to create the tangent plane and an improved polynomial free of points with high Riemannian distances. Our projection is based on a precalculation step independent of the triangulation parameters which needs to be done only once per data set. This results in an efficient projection method that makes use of the precalculated data.

First, in our precalculation step, we reconstruct the surface locally at each point $\mathbf{p}_i \in \mathcal{P}$ with bivariate polynomials of variable degree. We do so by applying weighted least squares in a local coordinate system (LCS). To find a good approximation of the normal and to correctly weight the points by Riemannian distance, we establish inter-point connectivity using the Cocone algorithm. Roughly speaking, this algorithm creates a triangulation from double cones centered at the $\mathbf{p}_i$ and which are aligned orthogonally with the *poles* of each Voronoi cell (the Voronoi vertex with the farthest distance to $\mathbf{p}_i$). Triangulations obtained from the Cocone are not free from overlaps and in its basic version, it is only suitable for noise-free surfaces. To actually perform the projection of a point $\mathbf{r}$ onto the surface, we find the closest point $\mathbf{p}_i \in \mathcal{P}$ and project $\mathbf{r}$ onto the polynomial surface of $\mathbf{p}_i$. In order to compensate for single outliers in the projections, we also consider the polynomial surfaces of neighbours of $\mathbf{p}_i$ and use the component-wise median of the projections.

The Cocone algorithm guarantees that points are connected topologically correct if $\mathcal{P}$ is an ε-sample with $\varepsilon \leq 0.08$, which means that for each point $\mathbf{s}$ on the surface, there exists a point $\mathbf{p} \in \mathcal{P}$ which is closer to $\mathbf{s}$ than ε times the distance from the medial axis to $\mathbf{s}$. MLS guarantees correct reconstruction for $\varepsilon \leq 0.01$ if point normals are associated with $\mathcal{P}$. In practice the Cocone performs well with ε up to 0.5 [ABK98]. Point normals are approximated as a byproduct and can be used to initially align the LCS of the polynomial in the next step.

Once inter-point connectivity is calculated we transform the points of $\mathcal{P}$ into the LCS centered at $\mathbf{p}_i$ and the z-axis is aligned with the normal approximated by the Cocone algorithm. We approximate the surface locally at each point $\mathbf{p}_i \in \mathcal{P}$ with a bivariate polynomial $p(\mathbf{x}) \in \prod_d^2$ of variable degree d which also provides a surface normal and curvature. We use this normal to improve the orientation of the LCS if the angle between the z-axis and the surface normal exceeds a specified threshold. We refer to the transformation into the LCS of a point $\mathbf{p}_j \in \mathcal{P}$ as $(\mathbf{x}_j, f_j)$ with $\mathbf{x}_j = (x_j, y_j) \in \mathbb{R}^2$ and $f_j \in \mathbb{R}$. To find the local surface approximating polynomial we minimize the sum of weighted error squares

$$e = \sum_{j=1}^{N} \big(p(\mathbf{x}_j) - f_j\big)^2 \cdot \theta\big(R(\mathbf{p}_i, \mathbf{p}_j)\big), \tag{1}$$

with the Riemannian distance R and the weighting function

$\theta(d)$. As in similar applications we suggest to set $\theta(d)$ to

$$\theta(d) = e^{-\frac{d^2}{h^2}} \tag{2}$$

whereas h can be interpreted as smoothing factor. By default h should be set to the local sample spacing at $\mathbf{p}_i$ but can be increased to smooth out measurement noise [ABCO*03].

We refer to the polynomial base of $p(\mathbf{x})$ depending on the degree d as $\mathbf{b}(\mathbf{x}) = (1, x, y, x^2, xy, y^2, \ldots)$ and to the polynomial coefficients as $\mathbf{c} = (c_1, c_2, c_3, c_4, c_5, c_6, \ldots)$. Therefore $p(\mathbf{x}) = \mathbf{c} \cdot \mathbf{b}(\mathbf{x})^T$ and we can rewrite (1) as

$$e = \sum_{j=1}^{N} \left(\mathbf{c} \cdot \mathbf{b}(\mathbf{x})^T - f_j \right)^2 \cdot \theta\left(R(\mathbf{p}_i, \mathbf{p}_j)\right). \tag{3}$$

To find the coefficients minimizing the error square sum we set the partial derivatives $\left(\frac{\partial e}{\partial c_1}, \frac{\partial e}{\partial c_2}, \ldots\right)$ to zero which leads to the linear system of equations

$$\mathbf{A}\mathbf{c} = \mathbf{d} \tag{4}$$

with

$$\mathbf{A} = \sum_{j=1}^{N} \mathbf{b}(\mathbf{x}_j)^T \mathbf{b}(\mathbf{x}_j) \cdot \theta\left(R(\mathbf{p}_i, \mathbf{p}_j)\right), \tag{5}$$

$$\mathbf{d} = \sum_{j=1}^{N} \mathbf{b}(\mathbf{x}_j)^T f_j \cdot \theta\left(R(\mathbf{p}_i, \mathbf{p}_j)\right). \tag{6}$$

Theoretically, every point is taken into account in order to find the minimum error sum. However, since equation (2) decays fast, in practice it is sufficient to only consider points in a certain distance to $\mathbf{p}_i$. The error in a distance of $4h$ is almost negible and for $6h$ it is around double precision.

We approximate the Riemannian distance with the Euclidean distance but only consider candidate points $\mathbf{p}_j$ if there exists a path (given through inter-point connectivity) from $\mathbf{p}_i$ to $\mathbf{p}_j$ and there is no point $\mathbf{p}_k$ on the path with $\|\mathbf{p}_k - \mathbf{p}_i\| > r_c$, where r_c is the maximum point distance. To ensure a good polynomial approximation and to prevent the triangulation from overlapping, points are iterated in a breadth-first search, projected onto the xy-plane and only collected if the projection does not lie within the convex hull of already collected points (see Fig. 2 for an 2D-example).

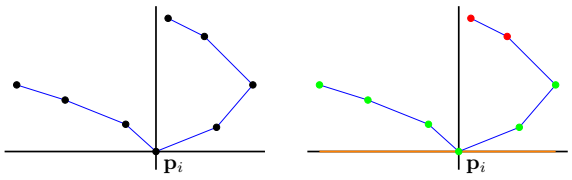

Figure 2: *Left*: overlapping points in the coordinate system prevent a good approximation. *Right*: (green) points not leading to overlaps are collected; (red) points intersecting the convex hull (orange) are rejected.

Solely solving equation (4) does not automatically lead to a good local surface approximation. As pointed out in [HZDS01, DZ04], a surface approximation can fail if all $\mathbf{x}_j$ in the local coordinate system are near an algebraic curve of same or lower degree than d. An example is given in Fig. 3. We can use the matrix condition $\kappa(\mathbf{A})$ as criterion to detect if the collected points are near an algebraic curve. However, high matrix conditions are not only caused by points in the vicinity of an algebraic curve but also from inappropriate scaling. To avoid high κ we scale the points of $\mathcal{P}$ when transforming them to the LCS We observed that κ is lowest if the average point spacing is approximately 1. The matrix condition is then a reliable criterion to check if the points lie near an algebraic curve.

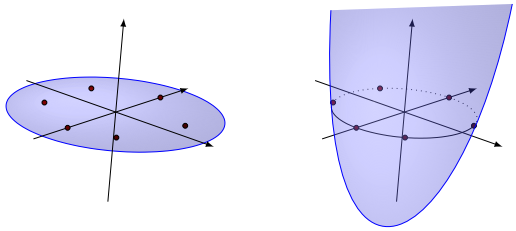

Figure 3: Points are near an algebraic curve. They might lie on a plane (*left*) as well as on a paraboloid (*right*).

We build matrix $\mathbf{A}$ according to equation (4) starting with an initial degree $d_{\max}$ and check if $\kappa(\mathbf{A})$ exceeds a threshold (which is independent of $\mathcal{P}$). In this case, we reduce the polynomial degree by one and rebuild $\mathbf{A}$. We repeat this step until the condition is below the threshold or we reach a degree of 1. We note that reasonable matrix conditions lie between 200 and 400 and we can set $d_{\max}$ to around 5. The above strategy allows us to use polynomials of high degree that nicely adapt to local points without the side-effects various methods not considering matrix conditions suffer from.

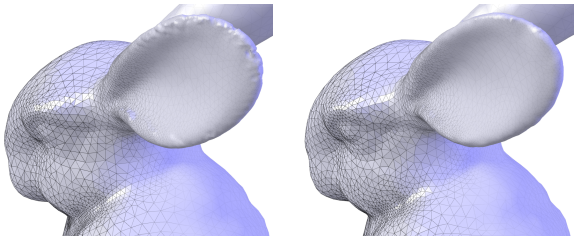

Figure 4: The *Bunny* ear. *Left*: MLS. *Right*: our method.

4. Results

We implemented a mesh generation tool *CloudMesh* [Uhr08] with an advancing front algorithm based on our projection method. Triangulations using MLS were performed with *Afront* [SSFS07]. We provide results of our experiments

with a collection of standard data sets, namely the Stanford models *Lucy*, *Bunny* and *Dragon* (see Figures 1, 4 and 5).

Establishing connectivity and polynomial surface reconstruction form up the preprocessing step. Here the computation time linearly depends on the number of points in $\mathcal{P}$. Triangulation time depends on the number of points as well as generated triangles. As a result, objects with many highly curved areas have increased triangulation time, see Table 1. All timings were done on a single core 2.4 GHz AMD CPU. We note that the preprocessing step is easily parallelizable. The given triangulation times are not optimal, since we visualize intermediate results during the triangulation process.

Data Set	*Bunny*	*Lucy*	*Dragon*
Points	34,834	262,909	435,545
Connectivity	0:32	4:15	7:26
Reconstruction	7:08	50:14	84:52
Triangulation	1:55	29:55	18:00
Vertices	24,802	442,422	198,157
Triangles	49,421	881,725	389,709

Table 1: Timings and results for various data sets.

Screenshots in Fig. 4 show that our approach is more robust and improves the triangulation results especially in regions of high curvature. A close-up of the dragon head (see Fig. 5) confirms that graduation in triangle size is more plausible due to polynomial reconstruction with variable degree.

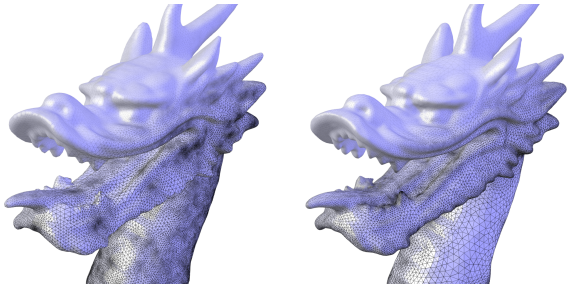

Figure 5: The *Dragon* head. *Left*: MLS. *Right*: our method.

5. Conclusion

We showed that there are problems with current MLS projection methods typically used for advancing front algorithms. We argued that these problems are related to missing connectivity information and presented a technique that creates inter-point connectivity in a first step which is used to create robust polynomial approximations of variable degree of the surface. In contrast to MLS, generation of the surface approximation does not involve a non-linear (or similar) optimization problem. Calculations are done in a preprocessing step only once per point set. Projections can then be performed efficiently and robustly with user-controlled granularity of the generated mesh.

References

[ABCO*03] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C.: Computing and rendering point set surfaces. *IEEE Trans. on Visualization and Computer Graphics 9*, 1 (2003), 3–15.

[ABK98] AMENTA N., BERN M., KAMVYSSELIS M.: A new voronoi-based surface reconstruction algorithm. In *SIGGRAPH '98: Proc. of the 25th annual conf. on Computer Graphics and interactive techniques* (1998), ACM Press, pp. 415–421.

[ACDL00] AMENTA N., CHOI S., DEY T. K., LEEKHA N.: A simple algorithm for homeomorphic surface reconstruction. In *SCG '00: Proc. of the 16th annual symposium on Comp. Geom.* (2000), ACM Press, pp. 213–222.

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

[AK04] AMENTA N., KIL Y.: Defining point-set surfaces. In *SIGGRAPH '04: ACM SIGGRAPH Papers* (2004), pp. 264–270.

[BB97] BERNARDINI F., BAJAJ C.: Sampling and reconstructing manifolds using alpha-shapes. In *Proc. 9th Canadian Conf. on Comp. Geom.* (1997), pp. 193–198.

[BW97] BLOOMENTHAL J., WYVILL B. (Eds.): *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[DZ04] DAVYDOV O., ZEILFELDER F.: Scattered data fitting by direct extension of local polynomials to bivariate splines. *Adv. Comput. Math. 21*, 3-4 (2004), 223–271.

[EM92] EDELSBRUNNER H., MÜCKE E.: Three-dimensional alpha shapes. In *VVS '92: Proc. of the 1992 workshop on Volume Visualization* (1992), ACM Press, pp. 75–82.

[Har98] HARTMANN E.: A marching method for the triangulation of surfaces. *The Visual Computer 14*, 3 (1998), 95–108.

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. In *SIGGRAPH '92: Proc. of the 19th annual conf. on Computer Graphics and interactive techniques* (1992), ACM Press, pp. 71–78.

[HZDS01] HABER J., ZEILFELDER F., DAVYDOV O., SEIDEL H.: Smooth approximation and rendering of large scattered data sets. In *VIS '01: Proc. of the conf. on Visualization* (2001), IEEE Computer Society, pp. 341–348.

[Kol05] KOLLURI R.: Provably good moving least squares. In *SIGGRAPH '05: ACM SIGGRAPH Courses* (2005), p. 213.

[Lev98] LEVIN D.: The approximation power of moving least-squares. *Math. of Computation 67*, 224 (1998), 1517–1531.

[SFS05] SCHEIDEGGER C., FLEISHMAN S., SILVA C.: Triangulating point-set surfaces with bounded error. In *Proc. of the 3rd Eurographics/ACM Symposium on Geometry Processing* (2005), Desbrun M., Pottman H., (Eds.), pp. 63–72.

[SSFS07] SCHREINER J., SCHEIDEGGER C., FLEISHMAN S., SILVA C.: Afront. http://afront.sourceforge.net/, 2007.

[Uhr08] UHRIG S.: Cloudmesh. http://sourceforge.net/projects/cloudmesh, 2008.