Sharp feature preserving MLS surface reconstruction based on local feature line approximations

Christopher Weber, Stefanie Hahmann, Hans Hagen, Georges-Pierre Bonneau

Preprint (march 2012)

to appear in Graphical Models 2012

Sharp feature preserving MLS surface reconstruction based on local feature line approximations

Christopher Weber [a] Stefanie Hahmann [b] Hans Hagen [a] Georges-Pierre Bonneau [b]

[a] *Technische Universität Kaiserslautern*

[b] *Université de Grenoble, Laboratoire Jean Kuntzmann, INRIA*

Abstract

Sharp features in manufactured and designed objects require particular attention when reconstructing surfaces from unorganized scan point sets using moving least squares (MLS) fitting. It's an inherent property of MLS fitting that sharp features are smoothed out. Instead of searching for appropriate new fitting functions our approach computes a modified local point neighborhood so that a standard MLS fitting can be applied enhanced by sharp features reconstruction.

We present a two-stage algorithm. In a pre-processing step sharp feature points are marked first. This algorithm is robust to noise since it is based on Gauss map clustering. In the main phase, the selected feature points are used to locally approximate the feature curve and to segment and enhance the local point neighborhood. The MLS projection thus leads to a piecewise smooth surface preserving all sharp features. The method is simple to implement and able to preserve line-type features as well as corner-type features during reconstruction.

Key words: MLS, sharp feature, surface reconstruction, Gauss map, clustering, point set surfaces

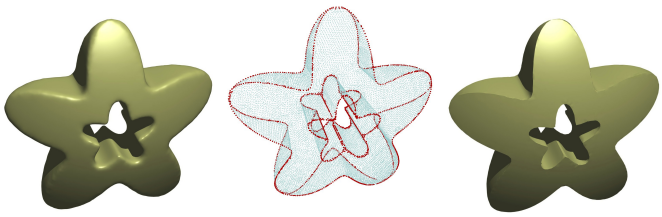

Fig. 1. Left: standard MLS surface. Middle: feature point detection in point cloud. Right: sharp feature preserving MLS

1. *Introduction*

Point based surfaces have become a very popular and attractive mesh less surface representation over the last decade. They define a smooth surface using local moving least squares (MLS) approximations of the data.
Since the initial approach [2] based on Levin's [18] projection operator, many improvements in terms of efficiency [3,28], rendering [12], stability [11], studies on properties and limitations [3,4] and variants using implicit surfaces [28,16,21] or specialized to non uniform, low dense sampling [8] have been developed.

The main strengths of MLS fitting include natural point denoising due to local least squares approximation, which can be seen as a local low pass filter. They generate smooth surfaces even in the presence of noise. It is thus assumed that the input data originates from a smooth surface.
Many scanned, manufactured or designed objects however exhibit sharp features. Sharp features are an important design element not only for mechanical parts. It is thus important to be capable to reconstruct those features not only for reverse engineering applications such as surface reconstruction, but also for quality control of a product, where the scanned object is compared to the CAD-prototype.
Even though point based surfaces were initially developed as an efficient visualization tool for point sampled data, they are nowadays used in a wide range of applications [10].
However, the requirements on point based surfaces to handle uniform noise and to generate a smooth surface is in contradiction to sharp feature preservation. It is thus not surprising that for example Amenta and Kil [4] attest instabilities near sharp features.
Although visual smoothness is one of the most required properties of surfaces or shapes in general, for the mentioned applications in product design, reverse engineering

or quality control it is however indispensable to preserve sharp features during reconstruction.

In this paper we address the problem of sharp feature preserving surface reconstruction from arbitrary point sets using moving least squares.
The output is a piecewise smooth point-based surface, meaning a refinable point cloud with normals. Feature points are first identified in the point cloud as part of a pre-processing using an adaptive Gauss-map clustering technique. Only neighborhoods containing some of the feature points are modified. Inside these neighborhoods, which are used by the projection operator, a local approximating feature curve is computed and serves to make a local decision in order to segment and enhance the point neighborhood. Applying standard MLS fitting [18] to the modified neighborhoods automatically generates piecewise smooth surfaces with sharp features since the points are projected to only one smooth surface part.
All advantages of standard MLS fitting (local smoothness, robustness to noise) are preserved and augmented with the ability to reconstruct sharp features.
Similar to RMLS [9] we identify individual neighborhoods within the regions of identified feature points, but we don't rely on computational expensive robust statistics. Instead we fit local curves through the identified feature points in order to slice the neighborhoods. Contributions and advantages of our technique in contrast to previous works [9,11,25,21] can be listed as follows:

- Sharp features are automatically reconstructed, no manual tagging [11,25] of feature lines is required.
- It is shown how local feature curve approximations can be used to partition the neighborhoods used for MLS projection.
- The method is much more simple to implement than statistical methods [9,6,7,21] where many parameters have to be fine-tuned.
- The quality of reconstructed sharp features is improved with respect to previous MLS methods, e.g. our local feature curve approximation and neighborhood modification reduces the appearance of jagged edges and produces smoother features.
- Feature point detection is a costly part here. MLS fitting with sharp features is then done in usual computation time. Performing the feature point detection in a pre-computation has the advantage that MLS reconstructions with different smoothness parameters can be performed multiple times. Our method speeds up such a process significantly.

The rest of the paper will be composed as follows: In Section 2, we review related work in the field of MLS reconstruction. Section 3 sets basic notations of MLS fitting. Section 4 describes the pipeline from feature point extraction, neighborhood modification and sharp feature surface fitting. Section 5 presents experimental results. Final remarks and future work close the paper.

2. *Related Works*

Moving least squares (MLS) surfaces are a very popular mesh less surface reconstruction tool [1,2]. In contrast to the common approach to generate a triangulated mesh, MLS approximate an unorganized dense, possibly noisy, set of points by an overall smooth point-based surface [18]. Many variants have been published [1,9,11,16,25] and several applications of point-based geometries can be found in [10]. Preservation of sharp features is not an inherent property of MLS fitting, any sharp feature will be smoothed and appears rounded after the fitting process. So let us focus now only on MLS approaches with sharp feature reconstruction in this section.
As one of the first, Fleishman et al. [9] presented a MLS approach that could reproduce sharp features. The so-called robust moving least squares fitting approach is based on methods used in statistics to search for outliers in the point set. It assumes, that the surface consists of several smooth patches connected by sharp features. The idea is, that a sample point lying on another smooth patch will be identified as outlier during the statistical analysis. An iterative refitting procedure is necessary to add points successively to the used neighborhood until an outlier is detected. Several iterations are necessary. It reconstructs the smooth surface parts, which in the end are connected through a sharp feature. Although the resulting surface is of a good quality, the outlier search and the refitting of the smooth surface parts are quite tricky to implement. The method also has problems with jagged edges as pointed out in [6,7] and needs a quite dense sampling. Our approach is inspired by the partitioning of the local neighborhood to reconstruct sharp features, although the way how we achieve this is completely different.
Guenebaud et al. [11] presented APSS, algebraic point set surfaces. They use moving least squares fitting of spheres instead of planes. This leads to a good stability in undersampled datasets. The sharp feature extraction itself is done manually by tagging of the point cloud, or automatic and based on a statistical analysis analogous to [9].
Öztireli et al. [21] use a kernel regression technique to reconstruct sharp features. They called it RIMLS, Robust Implicit MLS. Analogous to [9] they also use a statistics approach to find outliers belonging to different smooth patches on the surface. This technique has global parameters that can control the global visual sharpness of the reconstruction. However the resulting surface remains always C^2-continuous. So the reconstruction does not have a tangent plane discontinuous sharp feature, but only gives the visual effect of a sharp feature during rendering. Depending on the application, this can be seen either as an advantage or disadvantage.
An other interesting approach is the ERKPA by Reuter et al. [25]. ERKPA stands for Enriched Reproducing Kernel Particle Approximation. In this approach the user has to tag the sharp features manually and the second step of the

MLS projection (the computation of the local polynomial approximation) is modified. Instead of the normal projection, they add an enrichment function with discontinuous derivatives to the approximation function. The enrichment functions are compactly supported with a user specific support size to control the influence of the sharp feature. For n features, n enrichment functions are needed. Our approach doesn't require any manual user interaction.
Ohtake et al. [23] introduce the multi-level partition of unity implicits. Piecewise quadratic functions are used to capture the local surface and then use weighting functions that blend together these local shapes of the surface. An octree subdivision that adapts to variations in the complexity of the shape is used. Sharp corners and edges can be reconstructed by selecting appropriate shape functions.
Daniels et al. [6,7] build up on Fleishmans approach [9] to project points onto the features of the dataset. A covariance analysis for smoothing and growing polylines of the detected feature points leads to a global approximation of feature lines. Those feature lines are then used as starting points for the construction of a triangle mesh via an advancing-front algorithm. This approach is related to ours only in the sense that they provide a pipeline for global feature curve extraction. But they don't apply it to MLS surface fitting.
Lipman et al. [19] also presented a MLS fitting method that is able to handle sharp features and can handle many cases like line-type features quite well. But as they mention themselves, their method needs careful fine tuning in presence of noise and has problems with structures like the corner of a cube. Both situations are handled by our approach as we will show in Sections 4.2.5 and 5.
In contrast to previous MLS methods for sharp feature reconstruction that use statistics methods (forward search combined with Levin's MLS [9], or local kernel regression combined with implicit MLS [21]), we derive our method by enhancing basic MLS with both, an automatic adaptive feature points identification procedure and a neighborhood segmentation in case a feature belongs to it.

There is also a large number of feature preserving surface reconstruction techniques not based on MLS but relying on surface meshes. To name a few we refer to [14,5,13,15,29,31,26,27].

3. *MLS basics and notations*

The MLS surface of Levin [18] is a point based surface representation approximating a given set of point samples. Let $P = \{\boldsymbol{p}_1, \cdots, \boldsymbol{p}_n\}$, $\boldsymbol{p}_i \in \mathbb{R}^3$ be a set of unorganized points sampled from a surface. Let N_p denote the set of k-nearest neighbors of $\boldsymbol{p}$ and I_p the index set of N_p. Note that $k << n$. The MLS surface S_P is defined implicitly by a projection operator Ψ, that projects points from a vicinity of the MLS surface onto the surface itself. S_P is then defined as the set of points projecting onto themselves, i.e. the fix points of the operator.
The projection $\boldsymbol{p} \mapsto \Psi(\boldsymbol{p})$ of a point $\boldsymbol{p}$ is motivated by the fact that the surface can be locally approximated by a function. Its computation following Alexa et al. [2] is split into three steps:

a) For a given point $\boldsymbol{p}$ and its local neighborhood N_p compute a local tangent reference plane $H = \{\boldsymbol{x} | \langle \boldsymbol{n}, \boldsymbol{x} \rangle - D = 0,\ \boldsymbol{x} \in \mathbb{R}^3\}, \boldsymbol{n} \in \mathbb{R}^3, \|\boldsymbol{n}\| = 1$ by minimizing

$$\sum_{i \in I_p} \left(\langle \boldsymbol{n}, \boldsymbol{p}_i - \boldsymbol{p} - t\boldsymbol{n} \rangle\right)^2 w\left(\|\boldsymbol{p}_i - \boldsymbol{p} - t\boldsymbol{n}\|\right) \tag{1}$$

among all normal directions $\boldsymbol{n}$ and offsets t. $\langle \cdot, \cdot \rangle$ denotes the dot product in $\mathbb{R}^3$. w is a smooth monotone decreasing weighting function

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

where h can be used as a parameter to adjust the smoothing and interpolation behavior.

b) Find a bivariate polynomial approximation $g : H \mapsto \mathbb{R}$ to the surface in a neighborhood N_p of $\boldsymbol{p}$ by a weighted least squares fit similar to (1). Let $\boldsymbol{q}_i$ be the orthogonal projection of $\boldsymbol{p}_i$ onto H, (x_i, y_i) its local coordinates, $h_i = \|\boldsymbol{p}_i - \boldsymbol{q}_i\|$ the height of $\boldsymbol{p}_i$ over H. The coefficients of g are computed by minimizing

$$\sum_{i \in I_p} (g(x_i, y_i) - h_i)^2 w\left(\|\boldsymbol{p}_i - \boldsymbol{p} - t\boldsymbol{n}\|\right). \tag{3}$$

c) The projection of $\boldsymbol{p}$ is finally given by

$$\Psi(\boldsymbol{p}) := \boldsymbol{p} + (t + g(0,0))\boldsymbol{n}. \tag{4}$$

Implementation details can be found in [2].

4. *Sharp feature preserving MLS*

Our method decomposes into 2 separate phases. First, sharp feature points are marked in the point set. This can be done as a pre-processing. In the second main phase, the selected feature points serve as indicator to modify the local neighborhood for a point being projected.

Performing the most difficult and most time consuming feature point detection part, in a separate (pre-processing) phase has some advantages. First, once the feature points are marked in a given point set, an important part of computation time is gone. The user has thus the possibility to recompute the MLS surface with different sets of parameters in order to fine-tune the smoothness of the surface and the sharpness of the features without having to recompute the feature points again.

Second, both parts of the method rely on local neighborhood computations. The k-nearest search is thus done only once, stored and reused for MLS projection and neighborhood modification.

4.1. *Marking the sharp feature points*

Detection of sharp features in point clouds is not a trivial task because of possible noise in the data set. Since we only need the points belonging to a sharp feature to be marked, any of the methods [30,6] seems appropriate. Several other existent techniques for point clouds are dedicated to detect points on any kind of characteristical feature, e.g. points with high curvature value being not necessary a sharp feature [22,24,20]. These methods are however not appropriate here, since they either produce a large band of feature points or they reconstruct global feature lines using splines which is quite costly. Let us just summarize the method we use. All details can be found in [30].

The main idea here comes from the following observation for piecewise smooth curves and surfaces: The Gauss map of a particular point and its small neighborhood on a curve in 2D for example behaves different whether the point is a sharp feature or not. In fact there are 3 typical clustering behaviors to observe: one cluster when the point belongs to a flat region, a large fuzzy cluster for a point of a curved region or 2 distinct clusters in case of a sharp feature point. In 3D the clustering behavior is similar, except that more than 2 clusters may appear for a corner-type feature.
In the present setting however the input is not a smooth continuous surface, but an unstructured set of 3D points sampled from a surface without any normal information available. The expected output is a subset of points marked as points belonging to sharp features of the geometry, the so-called *feature points*. The algorithm first computes local neighborhoods for all data points using a kd-structure. For each data point inside its neighborhood, a so-called discrete Gauss-map clustering is computed to decide whether a data point $\boldsymbol{p}$ belongs to a sharp feature or not. Since no normals are available, all possible triangles $\Delta(\boldsymbol{p}, \boldsymbol{x}_i, \boldsymbol{x}_j)$, formed of $\boldsymbol{p}$ with two other points $\boldsymbol{x}_i, \boldsymbol{x}_j \in N_p$, $i, j = 1, \ldots, k$ are computed. The Gauss map of these triangles is then clustered using a geodesic distance criteria. The clustering behavior of the Gauss map of each point $\boldsymbol{p}$ can now be analyzed in order to decide if it's a sharp feature point or not. Note that for each triangle 2 opposite normal directions are mapped to the unit sphere. Each cluster has thus an opposite counter part on the sphere. In Figure 2 a 2D illustration is given. 3 typical clustering behaviors are shown from left to right corresponding to a point belonging to a flat region, to a curved region or to a sharp feature. In 3D the clustering behavior is similar, except that more clusters may appear for a corner-type feature. [30] presents finally an augmented adaptive version of this basic idea. It is adaptive in the sense that optimal local parameters are computed automatically so that very flat as well as very acute features can be recognized robustly even in the presence of noise.

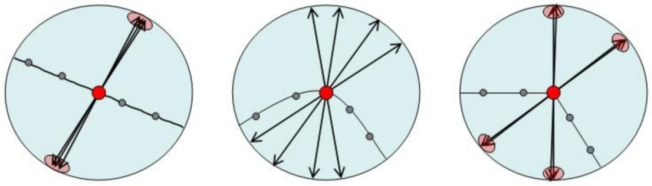

Fig. 2. 2D examples of a case resulting from Gauss-map clustering. Left: Clustering situation in flat area; middle: the curved case shows no clustering; right: in case of a sharp feature clearly distinguishable clusters are formed

4.2. *MLS surface computation*

4.2.1. *Overview*

After applying the feature detection method, we know for each point in the point set if it belongs to a sharp feature or not. This knowledge is exploited in the MLS fitting process in order to reconstruct sharp features as follows.
Let $\boldsymbol{p}$ be the point that is currently projected onto the surface. A neighborhood N_p of $\boldsymbol{p}$ is already available from the previous feature detection step.

Our strategy for sharp feature reconstruction is not to modify the mathematical formulation of the MLS method, but to modify the neighborhoods and thus the projection operator. A standard MLS projection is applied to all points, except if their neighborhood contains sharp feature points. Only then, the neighborhood is modified before projection applies.

Basically the neighborhood modification partitions N_p into two or three subsets, depending on the type of feature (line or corner) occurring inside. This partitioning is performed by clipping the neighborhood points close to $\boldsymbol{p}$ against a so-called local feature curve. This curve approximates the sharp feature locally inside N_p. Finally, the curve is up-sampled in order to smooth the sharp feature curve and to ensure equal quality everywhere. $\boldsymbol{p}$ is then projected on the surface part closest to $\boldsymbol{p}$.

Each neighborhood subset defines in fact a different smooth surface. Together these surfaces meet with discontinuous tangent planes creating thus the sharp feature.

The method is able to construct sharp features even in the case of corner-type features. Let us first treat the general case of a *line-type feature* traversing the neighborhood N_p, meaning a it curved feature line. The case of a corner and the way how to decide the type of feature will be handled later in Sections 4.2.5 and 4.2.6.

4.2.2. *Local feature curves*

Let us denote $\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_k \in N_p \subset P$ the k-nearest neighborhood of $\boldsymbol{p}$, and $\boldsymbol{f}_i$ the identified feature points among the set N_p. When a feature traverses N_p, the points $\boldsymbol{x}_i$ define two different surface parts.

We are interested only in the subset of points $\boldsymbol{x}_i$ which define the surface part closest to $\boldsymbol{p}$. To this end a feature

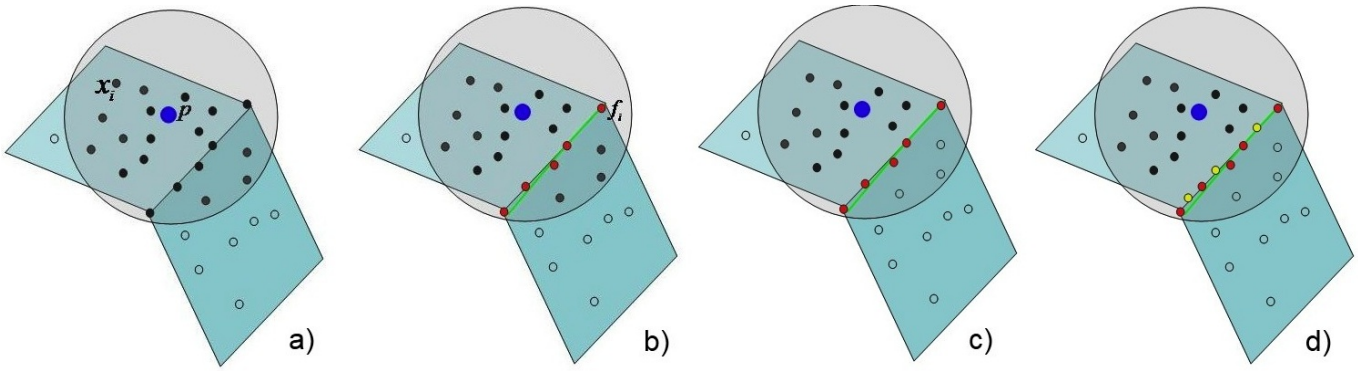

Fig. 3. *Local neighborhood modification:* a) the neighborhood of p (blue) is constructed. b) the feature points f_i (red) are identified and the local feature curve is constructed. c) the points $x_i \in N_p$ not belonging to the same surface part as p are removed from N_p. d) the removed points are replaced by samples from the feature curve.

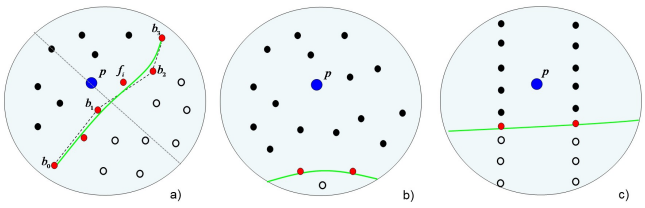

Fig. 4. a) Feature curve computation: red points are the feature points f_i. A cubic Bézier curve $\mathcal{F}$ (green) approximates the feature points. The Bézier points b_0 and b_3 are set to the extremal feature points. A simple heuristic determines b_1, b_2. Together they form the control polygon on $\mathcal{F}$ (gray dotted); b) Situation with only a small number of feature points f_i in N_p c) Situation of a poor sampled dataset in the extreme case of non uniform sampling

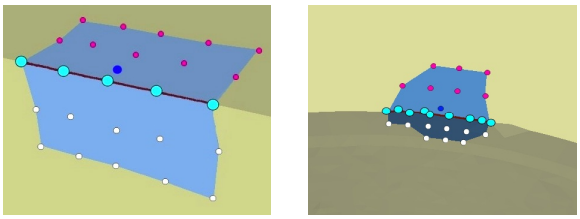

Fig. 5. Examples of computed local feature curves (black curve) approximating the cyan feature points detected in the neighborhood of p (blue point). Examples depicted from real surfaces in Figs.11 and 13.

curve is locally reconstructed as a cubic Bézier curve $\mathcal{F}$ defined by

$$\mathcal{F}(t) = \sum_{i=0}^{3} \boldsymbol{b}_i B_i^3(t), \quad \boldsymbol{b}_i \in \mathbb{R}^3 \tag{5}$$

and using the following heuristic to determine control points $\boldsymbol{b}_i$.

Since the neighborhood N_p represents only a very small part of the surface ($|N_p| = k$ with $k << n$), we can assume the traversing feature having a simple shape. Furthermore $\boldsymbol{f}_i$ being unsorted, we take the couple of feature points $\boldsymbol{f}_i$ and $\boldsymbol{f}_j$ with largest distance to define the first and last Bézier control points $\boldsymbol{b}_0 := \boldsymbol{f}_i$ and $\boldsymbol{b}_3 := \boldsymbol{f}_j$. The remaining $\boldsymbol{f}_k$ are now sorted with respect to $\boldsymbol{b}_0$ and $\boldsymbol{b}_3$. If there are exactly four feature points, $\boldsymbol{b}_1, \boldsymbol{b}_2$ are set accordingly. If there are more than four feature points, the set is divided in two parts and an arbitrary representative of each subset is chosen and set to be $\boldsymbol{b}_1, \boldsymbol{b}_2$. In Figure 4(a) an example of feature curve computation from 5 feature points is shown. This is a rough approximation, but sufficient when assuming that N_p represents a very small portion of the initial point set. As one alternative, a more sophisticated curve approximation would be costly without any extra benefit. Taking $\mathcal{F}$ as a simple straight line as another alternative however would result in a poor quality feature curve.

Implementation issues:
(1) With a neighborhood size of $k = 20$ we generally get 4 or 5 feature points in N_p. This corresponds in mean to the number of points which can be aligned to a straight line traversing a neighborhood in the middle of 20 points which are placed regularly in a circle or quad. See Fig. 5 for some real data examples.
(2) In case the number of feature points is < 4 in N_p we don't perform any neighborhood modification since not enough data is available for a reliable approximation of a local feature. The situation may correspond to a feature placed near the border of N_p, so that the number of points which would be eliminated by the feature curve is so small that their influence during the projection is also small, see Figure 4(b). Or it may correspond to errors in the feature detection.
(3) In the case of very special non-uniform sampling of P the method may fail, see Figure 4(c) for an example.

4.2.3. *Modification of neighborhood: Clipping N_p*

The locally generated feature curve $\mathcal{F}$ will now serve to divide the points $\boldsymbol{x}_i \in N_p$ in two subsets and to keep the subset which is 'close' to $\boldsymbol{p}$. Since the neighborhood N_p corresponds to a very small part of the point set, one can assume the two surfaces being almost planar near the feature line (a similar assumption is done for the MLS surface anyway). Two methods are possible:

(1) A Gauss-map clustering can be used again. For each

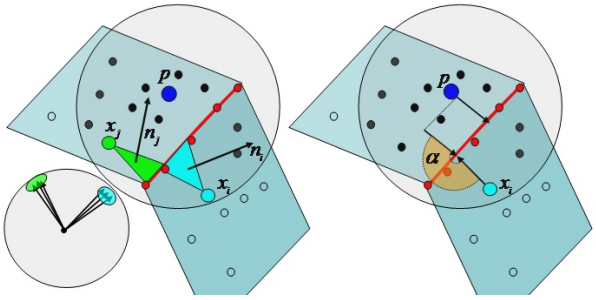

Fig. 6. Left: Gauss-map clustering. Right: Angle criterion. Both approaches can be used for neighborhood clipping in case of a line-type feature.

point $\boldsymbol{x}_i$, $i = 1, \ldots, k$ which is not a feature point a triangle $T_i = \delta(\boldsymbol{x}_i, \boldsymbol{f}_j, \boldsymbol{f}_k)$ is formed together with two feature points closest to $\boldsymbol{x}_i$. The corresponding normal vectors define the Gauss-map to be clustered. The result are two clusters, each cluster represents the points $\boldsymbol{x}_i$ defining one of the two surface parts, Fig. 6-left.

(2) An alternative method for separating the neighborhood points into two sets, is to compare the angles between the projection vectors of each point $\boldsymbol{x}_i$ onto the feature curve $\mathcal{F}$ with the projection of $\boldsymbol{p}$ onto $\mathcal{F}$. An angle greater than a fixed threshold value, would indicate that $\boldsymbol{x}_i$ and $\boldsymbol{p}$ belong to different surface parts, see Fig. 6-right.

Discussion:

Note that the existence of two surface parts seperated by a sharp feature is already acquired in this neighborhood. We don't need to decide whether there are two surface parts, but only whether the points belong to one or to the other part. Both methods can make profit from the preprocessing feature detection method and the angle thresholds used there. In fact, to decide if a point is a sharp feature point or not, a range of angles is computed there. Outside this range, the point is assumed to belong to a smooth surface part. So, in practice a range of $[40°, 140°]$ is used to define a sharp feature. The same range is used here to separate the points into two parts on either side of the feature curve. It is thus guaranteed that the present sharp feature reconstruction method works with the same precision than the previous feature detection in particular in the presence of noise. For the examples in this paper the first method has been used.

4.2.4. *Up-sampling of the feature points*

After clipping N_p, the modified neighborhood is now reduced in size. In order to guarantee the modified MLS projection to work with the same precision everywhere, the modified neighborhood has to be of same cardinal k as N_p. To this end additional points are sampled along the local feature curve, see yellow points in Figure 3(d). This particular choice is motivated and justified by the fact that increasing the number of local feature points

- strengthens the sharp feature reconstruction, and
- is smoothing the curve of the edge that is defined by the projection operator. A possible undersampling of the data points along sharp features can thus be resolved.

Note that this local sampling step solves a problem that has previously been raised as a limitation in [9] (end of Sect.5) and left for future work there. It is indeed a drawback of statistic-based feature reconstruction methods that the projection operator may not be smooth along sharp features.

4.2.5. *Special case: corner feature*

Up to now, we have assumed that only a line-type feature traverses the local neighborhood. It can happen that several sharp features meet at a common vertex. The corner of a cube is a typical example. If this situation happens inside a neighborhood the approach presented for line-type features in the previous sections would not work well, since a single curve cannot fit three or more feature curves meeting at a common vertex.

One way to solve this problem would be to construct several feature curves inside the neighborhood. It remain the questions how to ensure that the independently constructed curves meet at a common corner point? and how to chose this point? These questions are partially answered in Daniels et al. [7]. However, the critical mass necessary for [7] is not given here.

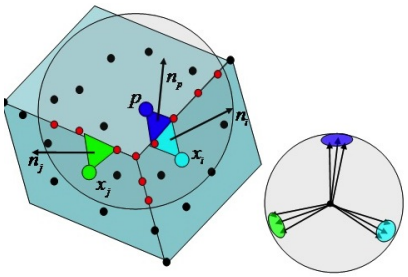

Fig. 7. Gauss-map clustering used for neighborhood clipping in case of a corner-type feature.

The solution we propose is consistent to the rest of the paper, since using again the Gauss-map clustering technique. It is composes of two steps:

Step 1: neighborhood clipping

What we need is a set of points belonging to the same smooth surface part as $\boldsymbol{p}$. Note that the feature points separating the surfaces parts are already available. It's therefore sufficient to use the clipping technique (1) proposed in Sect. 4.2.3: a Gauss-map for all neighborhood points $\boldsymbol{x}_i \in N_p$ is constructed by computing the normals of all triangles $\Delta(\boldsymbol{p}, \boldsymbol{f}_j, \boldsymbol{f}_k)$ and $\Delta(\boldsymbol{x}_i, \boldsymbol{f}_j, \boldsymbol{f}_k)$ $(\boldsymbol{x}_i \neq \boldsymbol{f}_j, \boldsymbol{f}_k)$, $i \in I_p$, where $\boldsymbol{f}_j$ and $\boldsymbol{f}_k$ can be chosen to be the two feature points, e.g. closest to $\boldsymbol{x}_i$. Clustering of these normals results in three (cube's corner) or more clusters. The cluster representing $\boldsymbol{p}$ thus designates the subset of $\{\boldsymbol{x}_i\}$ defining the requested surface part closest to $\boldsymbol{p}$. All other points are removed from N_p, see Fig. 7.

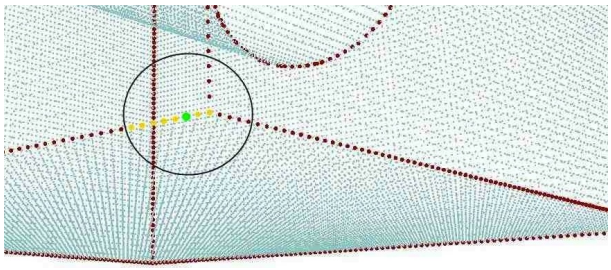

Fig. 8. Result of the feature clustering near the corner of a cube

Step 2: neighborhood up-sampling
In order to increase the number of points inside N_p and simultaneously increase the precision of the sharp feature, the set of feature points in N_p is increased by up-sampling some local feature curve. The computation of this feature curve in the case of a corner-type feature requires to divide the marked feature points into different sets, each set containing only the feature points belonging to one edge, see the colored points in Fig.9a) illustrating the case of a cube's corner. For efficiency reasons we don't compute these groups of feature points for each $\boldsymbol{p}$ since there may be many points in the point set close to $\boldsymbol{p}$ having the same feature points inside their neighborhood. Instead, we mark during the pre-processing (feature point detection) step for each detected feature point $\boldsymbol{f}$ all feature points inside N_f which belong to the same feature line as $\boldsymbol{f}$. Figure 8 illustrates the result of this computation for a feature point $\boldsymbol{f}$ (green dot) near a cube corner. The red dots in the figure are all the features detected in the data set. Its neighborhood is presented by the black circle. The yellow dots are thus the marked neighbor features points inside N_f and belonging to the same edge.

With these neighbor features marked for each feature point in the data set, it is now an easy task to determine for each point $\boldsymbol{p}$ the set of feature points closest to $\boldsymbol{p}$ (i.e. containing the closest feature point). A cubic Bézier feature curve $\mathcal{F}$ is computed from this set and used for up-sampling the modified neighborhood before projecting $\boldsymbol{p}$, see Fig.9b).

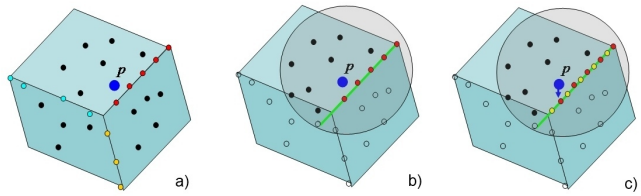

Fig. 9. *Local neighborhood modification near corners:* a) marking sets of feature points belonging to different features. b) selection of the closest feature set closest to $\boldsymbol{p}$, generation of the local feature curve, clipping of local Neighborhood N_p. c) up-sampling the neighborhood with points on feature curve.

4.2.6. *Characterization of neighborhoods*
One last issue remains: how to determine whether a point neighborhood N_p contains a line-type feature or a corner-type feature? The number of Gauss-map clusters computed for the neighborhood clipping (Fig. 6 and 7) clearly indicates if the points belong to two different surface parts (line-type) or to more surface parts (corner-type).

5. *Results*

We implemented and tested the sharp feature surface reconstruction method developed in the previous sections. Let us report here on 4 types of input data
· academic models without any noise (cube with hole, bilinear surfaces)
· academic models with random noise added (cube)
· complex models (fan disk, trimstar, octflower)
· raw data output of a scanner (drill).
We focus in particular on the sharp edges, since classic MLS projection is applied everywhere else on the model. Our method is in total slower than classic MLS due to the feature point pre-processing and neighborhood modification. It offers however the possibility to separate feature point detection from the projection step. All feature point computations including detection and the list of *nearest connected* neighbors can be pre-processed. Timings are given in [30]. Without these preprocessing steps, timings close to the classical MLS can be reached, since only a few number of the points (8% in the cube example of Fig.11) undergo a neighborhood modification with $O(k)$ operations for feature curve computation, clipping and up-sampling ($k = |N_p|$). For that reason we don't add timing tables here.

Figure 1 shows our sharp feature preserving reconstruction in comparison to Levin's classical MLS.

Robustness to feature detection

The present technique depends on a prior feature points detection method. The Gauss-map clustering method [30] has been proven to be reliably enough.
It can be observed, for example that perfect (exact) input data without noise leads to perfect recognition of feature points and our resulting MLS surface is perfect as well, see first row of Figure 11. Any other feature detection method can be used here instead.

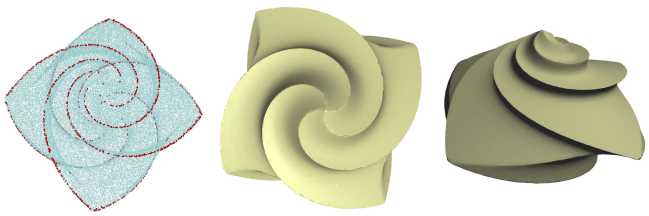

Fig. 10. Noisy octaflower model. Left: feature points detected. Middle and right: sharp reconstructions.

It can be observed, furthermore that our fitting method is able to auto-correct the input data. In case where only sparsely distributed feature points are detectable due to

noise in the input data for example, our method is able to correct this failure up to a certain limit. Figure 10 shows the octaflower data set. It is a noisy data set and it is not clean in the sense that data points are not exactly sampled on the sharp features of the octaflower. Even though many feature points are missing in the center and along the sharp curves (Fig.10-left), the resulting surface only looses sharpness at the center point, all other curves are sharply reconstructed. Let us note also the smoothness of these curves. The reason for this auto-correction property lies in the facts that on one hand local and continuous smooth feature curves are available and that on the other hand they are up-sampled inside the local neighborhoods.

In Figure 12 another academic example is shown. It represents two bilinear surfaces with a common sharp edge. The angle of the tangent planes varies along the edge from acute (40°) to obtuse (140°). This case of varying angles of a sharp feature is important in real life data sets and could cause problems for feature detecting and reconstructing systems using global sets of parameters. The adaptive feature detection method here ensures these cases to be covered by our reconstruction method.

Robustness to noise

The benefits of using local feature curves are also visible in the presence of noise. Figure 11 shows an academic example which has been perturbed. The noise is obtained by adding to each point a random vector chosen in a ball whose radius is 0.2% resp. 0.5% of the bounding box size of the model. In Figure 11 we show the clean model without noise in the top row, the 0.2% perturbed model in the middle row, and the 0.5% perturbed model in the bottom row. In each row the performance of the feature detection, standard MLS and our method are compared. As one can see, our sharp MLS reconstruction uses the benefit of the smoothing capabilities of MLS in the flat areas of the dataset while the sharp edges are still reproduced.

Complex data sets

Further data sets have been tested. The fan disk model in Figure 13, and the trim star model in Figure 1, are all available through the AIM@SHAPE Shape repository [1]. With the fan disk model, we would like to focus onto the region where the sharp feature looses its sharpness. The transition behaves smoothly and natural.

The raw data set of a drill output from a Cyberware^{TM} scanner in Figure 14 shows from left to right the feature points detection, shading of the original data, the classic MLS reconstruction, the sharp feature preserving reconstruction. This is a kind of worst case example. In the closeup in Figure 15 the difference between the smooth and the sharp reconstruction is clearly visible. The smooth reconstruction not only smooth the jagged edges of the original data but also the whole cutting edge of the drill. The sharp reconstruction is able to preserve the sharp cutting edge.

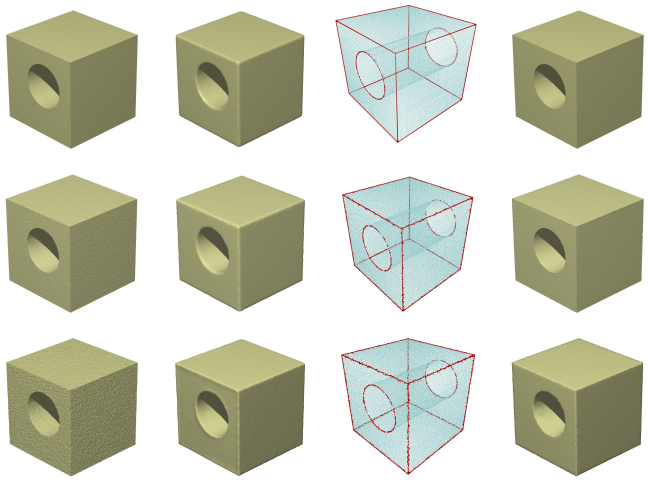

Fig. 11. Cube with hole. From left to right: original data set, smooth MLS reconstruction, feature points detected, our method. First row: clean data set without noise. Second row: data set perturbed with 0.2% random noise. Third row: data set perturbed with 0.5% random noise.

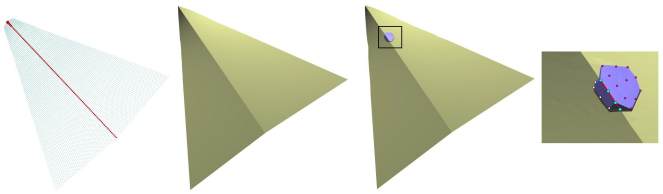

Fig. 12. Two bilinear surfaces with a common sharp edge. The angle of the tangent planes vary along the feature from acute (40°) to obtuse (140°).

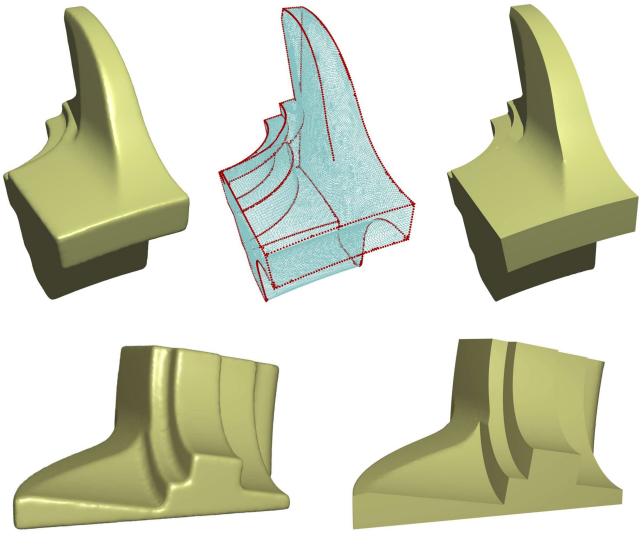

Fig. 13. Fandisk. Left: smooth standard MLS reconstruction. Middle: feature points detected. Right: our method. Bottom: front view of a smooth and sharp MLS reconstruction of the fandisk

[1] shapes.aimatshape.net

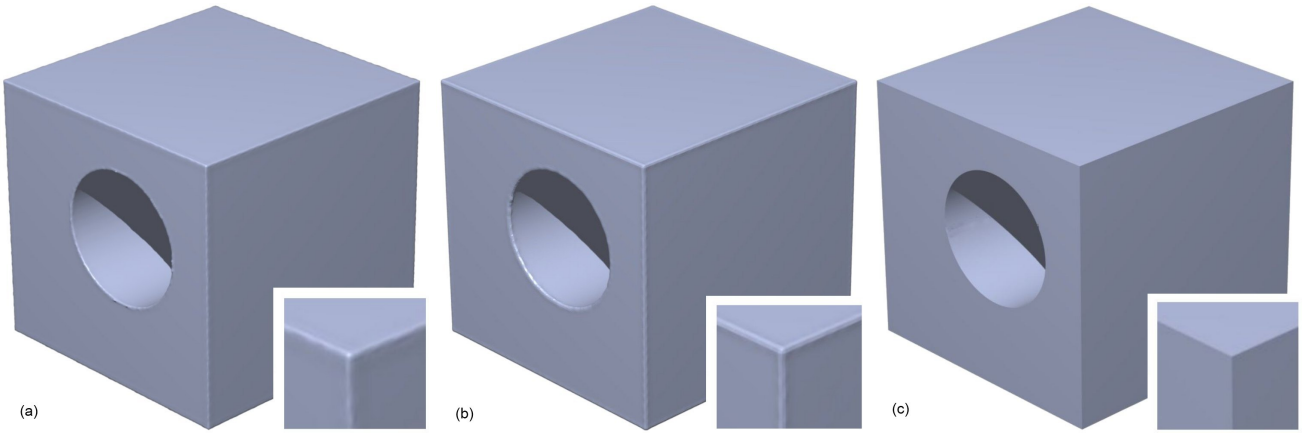

Fig. 16. MLS reconstructions using RIMLS — APSS — our method

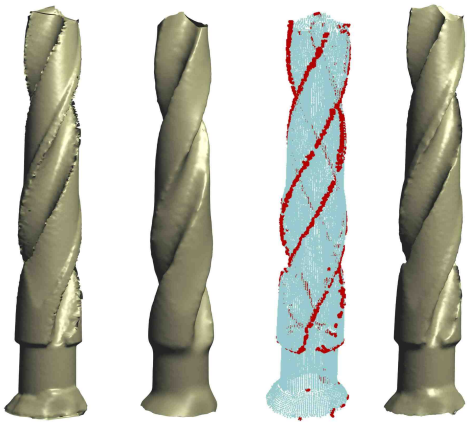

Fig. 14. Drill raw data by Cyberware^{TM} scanner. From left to right: shading of the original data, the classic MLS reconstruction, feature points detection, the sharp feature preserving reconstruction.

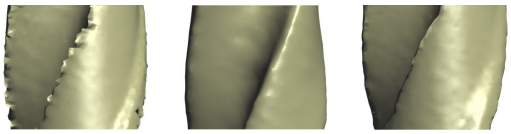

Fig. 15. Close up of the drill data; From left to right: shading of the original data, the classic MLS reconstruction, the sharp feature preserving reconstruction.

Comparison to known methods

We compared our method to other available feature sensitive reconstruction methods. The cube with hole data set was reconstructed with RIMLS [21] and APSS [11] using the Meshlab [2] implementations. In Figure 16 the original data set was taken. The same data set perturbed with 0.2% random noise served in Figure 17. Figures 16(a),(b) show two reconstructed surfaces by RIMLS and APSS. Naturally, in these reconstructions the edges are smoothed out. In Figure 16(c), we show the surface that is reconstructed using the new neighborhood modification approach. The reason is inherent to RIMLS characteristical behavior to produce C^2-continuous surfaces. In sharp regions curvature values increase, but the surface remains curvature continuous. The RIMLS reconstruction for the cube example took 170s, our method took 190s. Hereby we used the parameter settings proposed in Meshlab for sharp feature reconstruction. The APSS implementation in Meshlab unfortunately lacks the ability to reconstruct sharp features. But it can be used as an example to show the difference between a usual MLS reconstruction method and reconstructions with sharp feature preserving.

Another issue is sensitivity to noise. Figure 17 compares our method to others in the presence of noise. The noise is much more visible in Figures 17(a) and (b). The reason for that in RIMLS is that both effects together (sharp feature and smooth surface) can not be obtained. When setting the parameters, one has to choose. We privileged sharp features in Figure 17(a), but then the sensitivity to noise increased and leads to this bumpy surface. A smoother surface can be obtained, but then the features loose their sharpness significantly. Both Figures 16 and 17 clearly show the advantages of a sharp feature preserving method. The other sharp feature reconstruction in [9] was not tested. An implementation was not available and seems to be tricky.

Figure 18 is a further comparison. Here the angle of the tangent planes along the sharp feature is continuously varying from acute to obtuse. The closeups give better insight.

Limitations

In the presence of noise it is challenging to reconstruct sharp features as well as smooth surfaces in general. The present method is however able to enhance the sharpness in comparison to other works, but to a limited extend.

Even though our method reduces the dependency from parameter fine tuning (most parameters for feature point detection are adaptively computed) it still depends on the appropriate choice of k. Similar to all MLS methods we acknowledged this problem without being able to propose a better solution.

[2] `meshlab.sourceforge.net`

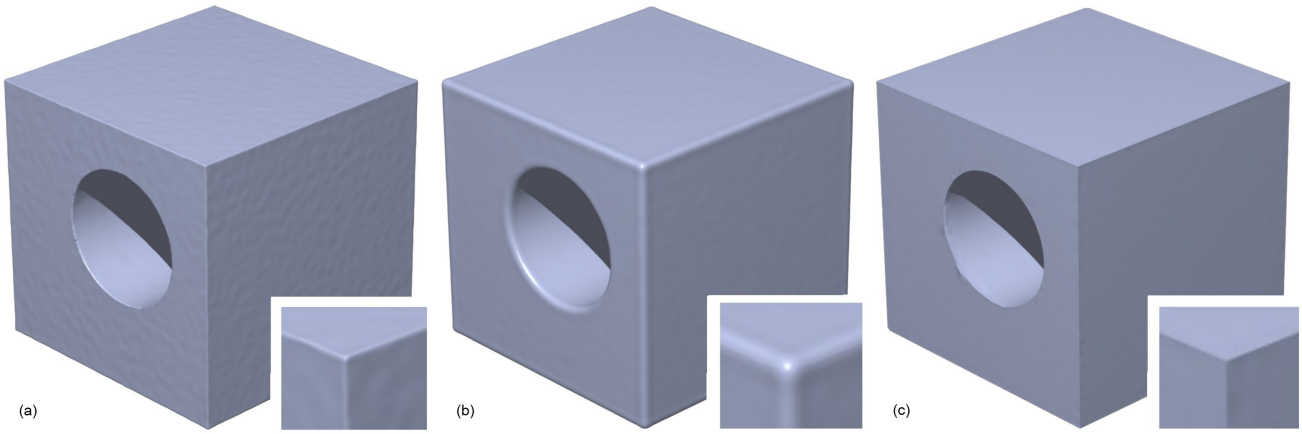

Fig. 17. MLS reconstructions with low noise using RIMLS — APSS — our method

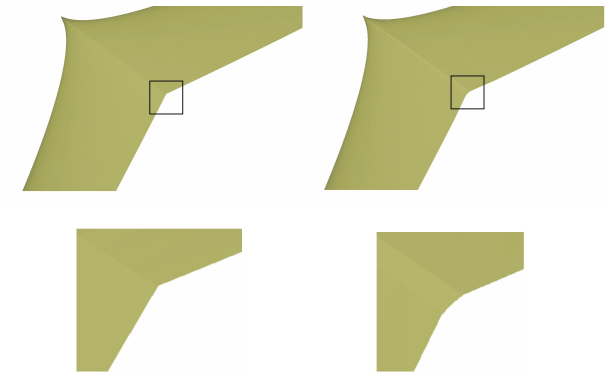

Fig. 18. MLS reconstructions using RIMLS and our method, the close up shows the different behavior along sharp features

6. *Conclusion and future work*

Our motivation was to show that it is possible to improve MLS sharp feature reconstruction with an algorithm which is not too tricky to implement. Simplicity and efficiency characterizes our method. And what is more important, it improves the smoothness of features curves of statistical methods such as [9]. We presented a new augmented version of the classic Moving Least Squares method to reconstruct a surface from point cloud data with sharp features. We not only segment the local neighborhood but modify it before the moving least squares projection step to achieve the sharp feature preservation. The method is capable of handling sharp line-type features as well as corner features. The resulting surface can be used in many applications such as reverse engineering or quality control in a design process.

The parameters available for fine-tuning of the resulting surface are the same as in the standard MLS approach. The smoothness parameter h from the weight function in (3) allows to adapt between interpolation and approximation. The size k of the local neighborhood governs about the quality of the sharp feature. But the size of the neighborhood is related to the computation time. In practice, $k > 20$ is not reasonable.

It would finally be interesting to investigate the trade-off between quality, simplicity and speed by comparing the present method to methods using global optimization [9] which are more robust but computationally more expensive.

The use of global feature curves instead of local ones is another interesting issue. We assume that computation time will increase, but it would be interesting to pursue in the future.

Another possible improvement of this present method could be to guarantee C^0-continuity. For this one has to guarantee uniqueness of the local feature curve or to use the more costly pre-computation of global feature curves. Here again the question remains if the quality of the resulting surface could be improved.

ACKNOWLEDGMENTS

The fandisk and trim-star are provided courtesy of MPII, all by the AIM@SHAPE Shape Repository. The drill model is courtesy of Sergei Azernikov, Siemens Corporate Research, Princeton. This work was partially supported by the IRTG 1131 of the DFG (German Research Foundation), and by the Deutsche Forschungsgemeinschaft INST 248/72-1.

References

[1] Alexa M., Behr J., Cohen-Or D., Fleishman S., Levin D., Silva T.: Point set surfaces. *IEEE Visualization 2001* (2001).

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

[3] Amenta, N. and Kil, Y.J. The Domain of a Point Set Surface. *Eurographics Symposium on Point-Based Graphics* (2004).

[4] Amenta, N. and Kil, Y.J. Defining Point-Set Surfaces. In *In Proceedings SIGGRAPH 04* (2004).

[5] Azernikov S., Miropolsky A., and Fischer A. Surface reconstruction of freeform objects based on multiresolution volumetric method In *ACM symposium on Solid modeling and applications (SM '03)* (2003).

[6] Daniels, J., Ha, L., Ochotta, T. and Silva, T. Robust Smooth Feature Extraction from Point Clouds. In *SMI* (2005).

[7] Daniels, J., Ha, L., Ochotta, T. and Silva, T. Spline-based feature curves from point-sampled geometry. The Visual Computer 24(6), (2008).

[8] Dey T. K. and Sun J. An Adaptive MLS Surface for Reconstruction with Guarantees. Symposium on Geometry Processing (SGP 2005).

[9] Fleishman S., Cohen-Or D., Silva C.: Robust moving least-squares fitting with sharp features. *ACM Trans. Graph.* (2005), 37–49.

[10] Gross M., Pfister H-P. *Point-Based Graphics*. The Morgan Kaufmann Series in Computer Graphics, (2007).

[11] Guennebaud G., Gross M.: Algebraic point set surfaces. In *In Proceedings SIGGRAPH 07* (2007).

[12] Guennebaud G., Germann M. and Gross M.: Dynamic Sampling and Rendering of Algebraic Point Set Surfaces *Computer Graphics Forum 27*, 2 (2008), 653–662.

[13] Hildebrand K., Polthier K., and Wardetzky K. Smooth feature lines on surface meshes. *Proceedings of Symposium on Geometric Processing*, 2005.

[14] Hoppe H., DeRose T., Duchamp T., Halstead M., Jin H., McDonald J., Schweitzer J., Stuetzle W.: Piecewise Smooth Surface Reconstruction *ACM SIGGRAPH 1994 Proceedings* , 1994, 295-302.

[15] Hubeli A. and Gross M. Multiresolution feature extraction for unstructured meshes. *Proceedings of IEEE Visualization*, pages 287–294, 2001.

[16] Kolluri R.: Provably good moving least squares. *ACM SIAM Symposium on Discrete Algorithms* (2005).

[17] Levin D.: The approximation power of moving least-squares. *Mathematics of Computation 67* (1998), 1517–1531.

[18] Levin D.: Mesh-independent surface interpolation. *Geometric Modeling for Scientific Visualization* (2003), 37–49.

[19] Lipman Y., Cohen-Or D., Levin D.: Data-dependent MLS for faithful surface approximation. *Symposium on Geometry Processing* (2008), 59–67.

[20] Mérigot, Q., Ovsjanikov, M., and Guibas, L. J. Robust voronoi-based curvature and feature estimation. In *Symposium on Solid and Physical Modeling* (2009), W. F. Bronsvoort, D. Gonsor, W. C. Regli, T. A. Grandine, J. H. Vandenbrande, J. Gravesen, and J. Keyser, Eds., ACM, pp. 1–12.

[21] Oztireli C., Guennebaud G., Gross M.: Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum 28*, 2 (2009).

[22] Gumhold, S., Wang, X., and McLeod, R. Feature extraction from point clouds. *Proceedings of 10th International Meshing Roundtable V* (2001).

[23] Ohtake Y., Belyaev A., Alexa M., Turk G. and Seidel H-P. Multi-level partition of unity implicits In *In Proceedings SIGGRAPH 2003* (2003).

[24] Pauly, M., Keiser, R., and Gross, M. Multi-scale feature extraction on point-sampled surfaces. *Computer Graphics Forum* (2003).

[25] Reuter P., Joyot P., Trunzler J., Boubekeur T., Schlick C.: Surface reconstruction with enriched reproducing kernel particle approximation. *Eurographics Symposium on Point-Based Graphics* (2005).

[26] Scheidegger, C.E., Fleishman, S., Silva, C.T. Triangulating point set surfaces with bounded error. *Symposium on Geometry Processing* (2005), 63-72.

[27] Schreiner, J., Scheidegger, C., Fleishman, S., Silva, C. Direct (re)meshing for efficient surface processing. *Comput.Graph. Forum 25* (2006), 527536.

[28] Shen, C., O'Brien, J. F., and Shewchuk, J. R. Interpolating and approximating implicit surfaces from polygon soup. *ACM Trans. Graph. 23* (August 2004), 896–904.

[29] Watanabe K. and Belyaev A. Detection of salient curvature features on polygonal surfaces. *Computer Graphics Forum*, pages 385–392, 2001.

[30] Weber C., Hahmann S., Hagen H.: Sharp feature detection in point clouds. In *In Proceedings SMI 10* (2010).

[31] Weinkauf T. and Günther D. Separatrix Persistence: Extraction of Salient Edges on Surfaces Using Topological Methods. *Computer Graphics Forum (Proc. SGP '09)*, 28(5):1519–1528, July 2009.