TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik - Institut für Software und Multimediatechnik

TECHNICAL REPORT

ISSN 1430-211X

TUD-FI13-03 August 2013

Sören König, Stefan Gumhold

Robust Surface Reconstruction from Point Clouds

Technische Universität Dresden
Institut für Software und Multimediatechnik
Lehrstuhl für Computergraphik und Visualisierung
Fakultät Informatik, 01062 Dresden, Germany
http://www.inf.tu-dresden.de/cgv

Robust Surface Reconstruction from Point Clouds

Sören König
Faculty for Computer Science
TU Dresden
Email: Soeren.Koenig@tu-dresden.de

Stefan Gumhold
Faculty for Computer Science
TU Dresden
Email: Stefan.Gumhold@tu-dresdend.de

***Abstract*—Although a variety of algorithms have been proposed for surface reconstruction from point clouds a lot of real world data sets cannot be reconstructed satisfyingly. In this paper we present a new reconstruction pipeline that can handle noisy point clouds with insufficiently sampled features, surface boundaries, nearby surface sheets, non orientable surfaces and even self intersecting surfaces. We combine and improve techniques for normal estimation, denoising and surface tessellation into a robust reconstruction scheme. Wherever possible we locally estimate necessary process parameters to adapt to the point cloud and to avoid user input. We demonstrate the superiority of our approach on several difficult data sets.**

I. Introduction

Today we find more and more projects that acquire huge amounts of real world 3d data. Airborne laser scanners allow the reconstruction of landscape and the roofs of houses on a grand scale. Vans with mounted camera systems scan the street level of whole towns. Huge amounts of photographs can be collected automatically from the web allowing for very detailed 3d reconstructions of popular sites. In all these techniques the financial budget limits the quality of the acquired 3d data, which is full of noise, outliers, holes and other problems.

Although there is a huge amount of techniques that reconstruct surfaces from noisy point clouds, the generated surfaces of these real world data sets are typically not satisfying. In this work we proposed a new surface reconstruction scheme that can cope with much more difficult input data as known approaches. Our surface reconstruction pipeline can handle inhomogeneous samplings, noise, sharp features, nearby surface sheets, self-intersections, boundaries and even non-orientable surfaces. Minimal user input is necessary, making it very useful to the community.

An overview of our approach can be found in the next section, which is followed by the related work section. The following sections detail the three stages of our reconstruction pipeline. The result section shows that our approach can handle difficult input data much better than the state-of-the-art in surface reconstruction.

II. Overview

In this paper we propose the surface reconstruction pipeline illustrated in Figure 1. The input is a point cloud $\mathbf{P}$ given as a list of 3d points. We denote a single point by $\vec{p}$. The first stage (described in section IV) estimates the tangential space of each point for denoising and normal estimation. We use $\mathbf{N}$ to denote the list of point normals and $\vec{n}_{\vec{p}}$ for the point normal of $\vec{p}$.

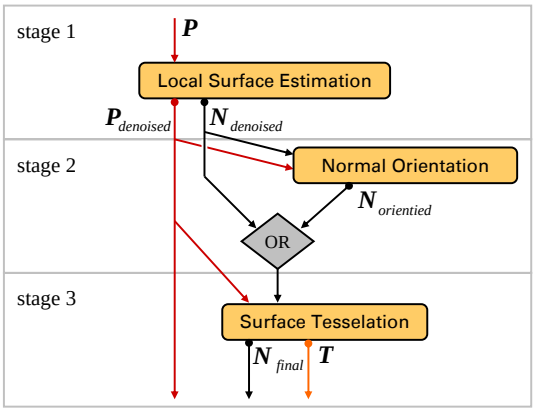


Fig. 1: Overview of the surface reconstruction pipeline.

The second stage (see section V) finds a globally consistent orientation of the estimated normals but does not touch the points themselves. The tessellation algorithm in the third stage (see section VI) constructs a triangular approximation of the surface. The denoised points serve as vertices in the tessellation. The tessellator can use the oriented normals or optionally ignore their orientation.

Although new techniques are developed for each stage, we see an important contribution in the proposed combination of the stages. Wrongly estimated normals or wrongly propagated normal orientation can be corrected during the surface tessellation by exploiting topological consistency.

All stages build on point neighborhoods, which we compute with the approximate nearest neighbor library [AMN*98]. The neighborhood of a point $\vec{p}$ is denoted by $N_{\vec{p}}$ and the neighbors $\vec{q}_i$.

III. Related Work

We can only touch the large amount of previous work related to the three stages of our scheme.

Local Surface Estimation Existing methods can be classified in numerical methods and methods based on Voronoi diagrams. Numerical methods are preferred due to easier implementation and faster running times. Hoppe [HDD*92] proposed to estimate the local tangent plane by applying a PCA to the k-neighborhood around a given point $\vec{p}$. In the discussion section he mentioned that one strategy to adjust the parameter k controlling the neighborhood size would be to analyze the behavior of the eigenvalues when the value of k is changed. If

k is chosen to small the eigenvalues start to become instable, if k is chosen to large the third eigenvalue corresponding to the direction of lowest variation (normal direction or thickness) starts growing. He also proposed a method for solving the normal orientation problem by consistent propagation of normal orientation. We improve this method in section V.

Pauly et al.[PKKG03] use the normal of a fitted plane or alternatively the gradient of an additionally fitted bivariate polynomial. The main difference to Hoppe's approach is the use of a distance decaying exponential weighting function inspired by Levin's work [Lev03]. In [OBA*03] general 3D quadrics, bivariate quadratic polynomials in local coordinates and piecewise quadric surfaces are used as local shape functions. Similar to our approach they propose to separate points from different surface sheets at sharp features by clustering. Mitra et al.[MNG04] propose to use the points within a ball of radius r instead of a fixed number of neighboring points. They also propose an iterative method to estimate an optimal radius assuming zero mean noise with given standard deviation.

The Voronoi based technique in [AB99] shows that the line from the pole of point $\vec{p}$ to the point $\vec{p}$ itself can be used as an approximation to the surface normal at this position. Later the big Delaunay ball method [DG04],[DS05] was proposed as an improvement to handle noisy point cloud data. One large drawback of these methods is the computation of Voronoi cells which can be quite difficult and slow especially in the context of extremely large data sets produced by state-of-the-art scanning technologies.

Denoising and Filtering Filtering techniques for surfaces can be divided into isotropic and anisotropic techniques. One of the first fast isotropic method was described by Taubin and works on triangulated surfaces. This work was later improved by Desbrun et al. [DMSB99] to handle irregular meshes by using a geometric flow analogy allowing for significant smoothing in reasonable time even. The basic drawback of isotropic filters is that no differentiation between correct data and noise is made during smoothing. To preserve sharp features the idea of anisotropic methods[DMSB00] is to modify the diffusion equation in such a way that the local diffusion is driven by the curvature tensor. Typically these kind of methods are computationally expensive and rely on iterative numerical solving techniques. Another very popular non-iterative edge-preserving smoothing approach is bilateral filtering. Originally described as a combined domain and range filtering technique for images[TM98]. It was later adapted from Fleishman et al.[FDCO03] and from Jones et al. [JDD03] for meshes and then to point clouds and normals [JDZ04]. The basic idea of a bilateral filter is that smoothing should take place over spatial nearby values which also have similar functional values.

Surface Reconstruction There are three main approaches to surface reconstruction: surface fitting, surface evolution and methods based on combinatorial structures like the Delaunay tetrahedralization. Recently also statistical methods have been proposed [JWB*06].

Surface fitting can be split into local [HDD*92], [ABCO*01], [FCOS05], [DS05], hierarchical [OBA*03] and global fitting. Two types of surfaces are commonly used for fitting: MLS surfaces [ABCO*01], [FCOS05], [DS05] and implicit surfaces [HDD*92], [OBA*03]. The latter have the problem that only closed surfaces can be reconstructed.

Among the evolution based methods are level set methods [Whi98], [ZOF01] and Poisson surface reconstruction [KBH06] which solves a partial differential equation with a multi-grid solver. Also the Bayesian surface reconstruction [JWB*06] is solved in an evolutionary fashion. All these methods discard the original points and perform a resampling and or remeshing of the estimated surface.

Our scheme is more related to the combinatorial methods as it triangulates the original points. This has the advantage that original surface features can be reconstructed better as shown in the results section. Most existing methods are based on one or two Delaunay tetrahedralizations of the point cloud [AB99], [ACK01], [AGJ02], [DG03], [DG04], [KSO04]. The surface triangles of the reconstruction are filtered from the triangles in the tetrahedralizations, such that the original points are tessellated. But the Delaunay tetrahedralization turns out to be too time consuming for large data sets. The results section shows that publicly available implementations have problems at boundaries and nearby surface sheets. They can neither handle self-intersections. The advantage of most combinatorial approaches is that they can guarantee a topologically correct and geometrically close reconstruction as long as the sampling is dense enough.

Our approach builds on an approximate knn-graph that can be computed efficiently. The tangential space at each point is constructed from normals that are estimated in a preprocessing stage. A local 2D Delaunay triangulation is used as filter. Then we grow a tessellation that is maximally consistent with the local Delaunay triangulations. A final hole closing step sews the surface together at sharp features. This is similar in spirit to the zippering approach proposed in [TL94]. Overall our approach is very fast and can deal with extremely difficult input data.

IV. Local Surface Estimation

Techniques to estimate local surface properties of a point sampled surface S typically approximate S by fitting a simple surface model M – for example a plane or a paraboloid – to the neighborhood $N_{\vec{p}}$ of each point $\vec{p}$ in the point cloud $\mathbf{P}$. Assuming that the local approximation of S by M is sufficient allows to estimate local surface properties of S directly from M.

Crucial are the selection of the model M, the criterion to choose a proper neighborhood and the numerical method used to fit M to the neighbors $\vec{q}_i$.

The model M should be able to capture all relevant surface properties of the local surface without having too much degrees of freedom to avoid over-fitting especially if sample points are perturbed by noise.

Intuitively, a good neighborhood criterion should select a sufficient number of points within a small Euclidean distance to the point $\vec{p}$. To preserve sharp feature like edges and corners it is important that neighboring points are selected only from the surface sheet on which $\vec{p}$ lies. In the presence of noise standard least squares fitting techniques are not recommended because single outliers can have strong (quadratic) influence on the achieved fitting results. A very popular modification

to least squares is the weighted least squares approach. This method limits the influence of outliers by minimizing the sum of weighted squared distances between the model and data points. The weights are chosen in such a way that nearby points are given larger weights than points which are far apart.

A. Normal estimation method

To produce very accurate normals our proposed estimation procedure handles all the described issues with special care. Before describing the whole algorithm, we first want to explain the basic concept used in our method.
Figure 2 shows a difficult situation where standard methods will fail to produce good results. The depicted example contains sharp feature and several nearby surface sheets falling into the neighborhood of the point selected for normal estimation. Local surface models like planes or second order surfaces are not able to approximate such a complex neighborhood. Selecting a more complex surface model which is able to deal with sharp edges or corners is extremely difficult to fit. Choosing a smaller k or radius to define the neighborhood typically doesn't help near corners and is limited by the minimal number of points needed for fitting. Moreover decreasing k too much can have a negative impact on the robustness in case of noise. Our method tries to identify a better neighborhood for fitting by applying a RANSAC approach [FB81].

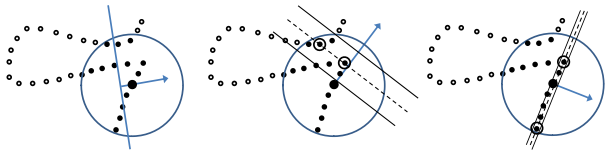


Fig. 2: A difficult situation for normal estimation due to three surface sheets falling into the neighborhood (circle) queried by a knn-request with $k = 15$ for the large center point. Left: result achieved by weighted least squares fitting of a plane. Middle: example for a bad RANSAC sample for $\tau = \frac{6}{15}$(indicate by two small circles). This sample produces a large error and would be rejected. Right: example for a good sample producing a small error, resulting in a good normal.

The main idea of using RANSAC for fitting a model is to randomly sample the smallest possible subset of points needed to define a candidate of the model in a first step (e.g. two in case of a line or three in case of a plane). In a second step the quality of the candidate is evaluated by computing a cost function involving the complete neighborhood of points. In the basic version of RANSAC an apriori known threshold t is used to divide the points into a set of inlier points (consensus set) with a model distance smaller than t and a set of outliers. The algorithm succeeds if a sufficiently large set of inliers is found. Otherwise the two steps are repeated.
In our case two different types of outliers can occur: points from different surface sheets and outliers in the classical sense not representing a valid surface sample often produced due to errors in the acquisition process. Many proposals are made to improve the basic version of RANSAC considering automatic parameter adjustments and improved cost functions, see [TZ00] for more details.

Our implementation is based on three parameters k, τ and $d_{\max}$ which are automatically adjusted as described in subsection IV-B. The first parameter k indicates the number of points $\vec{q}_i$ in a neighborhood N_p around a selected point $\vec{p}$. τ defines the percentage of expected inliers within a given neighborhood N_p. The minimal number of necessary RANSAC iterations can be directly computed from k and τ.

First we define the probability p_{succ} to sample three inliers which define a valid plane from a set of k points with an inlier probability of τ.

$$p_{succ} = \frac{\tau \cdot (\tau \cdot k - 1)(\tau \cdot k - 2)}{(k-1)(k-2)} \tag{1}$$

A lower bound M of necessary trials to sample at least one valid plane within a fixed surety p_{surety} of e.g. 99 percent can be computed by the following formula

$$M \geq \frac{log(1 - p_{surety})}{log(1 - p_{succ})} \tag{2}$$

as illustrated by the tree of all possible combination depicted in Figure 3.

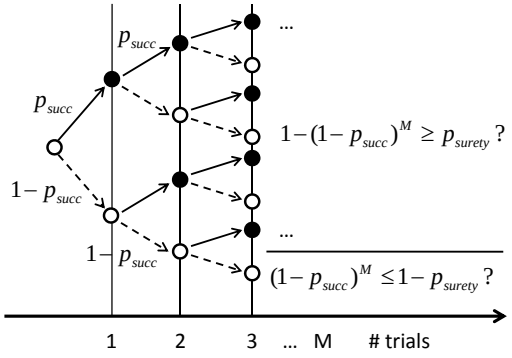


Fig. 3: Filled circles represent a successful trial, empty circles a fail. Arrows indicate the transition probabilities. The probability not to hit a valid plane at all after M trials is $(1-p_{succ})^M$ and corresponds to the lowest path in the tree. All other paths consist of one or more successful trials. The probability of the lowest path should be less or equal to the required inverse surety $1 - p_{surety}$. Taking the logarithm on both sides of this inequality and resolving for M yields the inequality for M given in 2

To find the best candidate plane pl_{best} we use a modified version of the original cost function. Because we expect $s = \tau \cdot k$ inliers we compute the error of a candidate as the sum of squared distances between the nearest s points and the surface of the candidate.

Although the normal of pl_{best} is very close to the true underlying surface for objects with low curvature and without sharp features the quality of the estimation can be further improved, especially in case of high curvature.

Combining the ideas of robust weighted least squares and bilateral filtering, we use the plane $pl_{best} : \vec{n}_{best}^T \vec{x} + d_{best} = 0$ to assign a weight w_i for each neighboring point $\vec{q}_i$ in N_p.

This weight is used to do a weighted least squares fit of a higher order surface which can better deal with high curvatures.

$$w_i = e^{-\frac{(\vec{n}_{best}^T \vec{q}_i + d_{best})^2}{h_{pl}}} \cdot e^{-\frac{||(\vec{q}_i - \vec{p}||_2)}{h_p}} \quad (3)$$

The first term penalizes the squared distance to the best plane pl_{best}, the second one penalizes the squared distance to the point $\vec{p}$ which was used to define the neighborhood N_p. The parameters h_p and h_{pl} control the decay of the two exponential functions. If no better motivated choice from the data is available we propose to set $h_{pl} = \frac{1}{3} \cdot (\vec{n_{best}}^T \vec{q}_m + d_{best})^2 + \epsilon$ and $h_p = \frac{1}{3}||(\vec{q}_m - \vec{p}||_2$ where $\vec{q}_m$ is the inlier with the largest distance to p_{best}. The ϵ is only added to avoid a division by zero in the weight computation if the worst inlier has a zero distance to the plane. This choice for h_{pl} and h_p ensures that the influence of points outside of the inlier set is almost zero.

In our implementation we use a paraboloid in the form $pa : z = ax^2 + bxy + cy^2 + dx + ey + f$. Therefore we first need to find a local reference frame defined by an origin $\vec{o}_p$ and a rotation matrix R_p. Using the weights w_i the origin $\vec{o}_p$ can be computed as the weighted mean of all $\vec{q}_i \in N_p$. The orientation of the local frame can be found by computing the matrix of eigenvectors E of the weighted covariance matrix of all points $\vec{q}_i$ with $i \in N_p$. The eigenvectors in the columns of E should be sorted in descending order by their corresponding eigenvalues to get R_p. This ensures that the third column which represents the z axis of the local coordinate frame is the direction of the smallest variance around the weighted mean. After transforming the points $\vec{q}_i$ into the local frame using $\vec{q}_i' = R_p \cdot (\vec{q}_i - \vec{o}_p)$, weighted least squares regression can be used again to fit the paraboloid into the points $\vec{q}_i'$ by minimizing the weighted squared z distances.

To measure the overall success of our fitting process which is used in our automatic parameter adjustment procedure we define the success rate sr as the ratio between the number of $\vec{q}_i'$ with a z distance to the paraboloid smaller than a given threshold $d_{\max}$ and the number k of points in the neighborhood.

$$sr = \frac{\text{number inliers defined by } d_{\max}}{k} \quad (4)$$

The normal direction of the paraboloid is given analytically by the following vector $(ax + by + d, bx + cy + e, -1)^T$ divided by its length and can be transformed back into global space by multiplying with R_p^T. The described steps of the algorithm are summarized in Figure 4.

B. Adaptive Parameter Adjustment

The three parameters which are automatically adjusted to control the normal estimation are $d_{\max}$, k and τ. Before describing the adaptation in detail, we first want to give some intuitions on how the RANSAC estimation process is influenced by these values. The parameter $d_{\max}$ decides whether a given point is close enough to a computed local surface approximation to derive normal information. Choosing

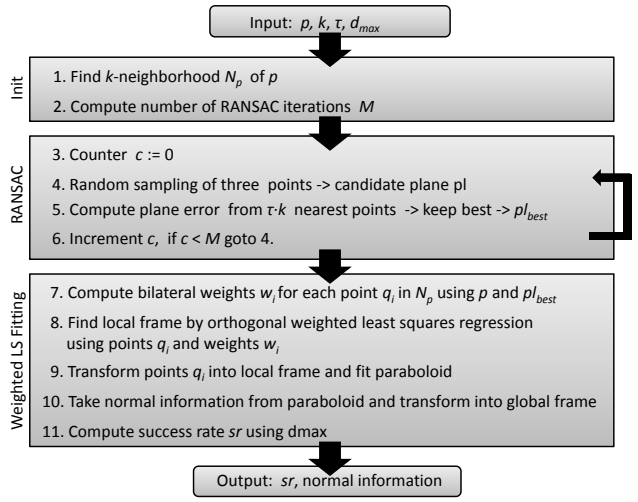


Fig. 4: Summary of normal estimation steps.

$d_{\max} = 0$ will only allow to compute normals for points which are lying exactly on the local surface. To be able to handle noisy point clouds the value of $d_{\max}$ must be increased to accommodate for the amount of noise present in a given data set. So we propose to start with a small value for $d_{\max}$ consistent with the smallest amount of noise present in the data and keep this value constant until the number of new normals which can be identified becomes too small. This number can be observed by computing the success rate (see equation 4). The initial value of $d_{\max}$ is identified by applying a bisection search to find the smallest value for that parameter which can achieve a success rate $sr > 0.5$ by random sampling. The other two parameters are fixed at $\tau = 0.5$ and $k = \sqrt{n}$ during this search where n is the number of points in the input point cloud $\mathbf{P}$.

The parameter k defines the neighborhood size and should be adaptive to varying sampling densities. Typically a smaller value is needed for homogeneously sampled regions. τ, which defines the expected percentage of inliers, restricts the shape of the neighborhood; large values of τ corresponds to round patches. Both parameters together control the size of the support set of points used within the fitting process. A proper strategy is to start with large values for k and τ to identify large and round patches first. The full picture on how the different adjustment mechanisms act together is given as pseudo code in Figure 5.

The first outer loop is used to slowly increase the parameter $d_{\max}$ after each iteration until all points are processed. The next inner loop varies the expected number of points per facet n_p from $k_{max} = \sqrt{n}$ down to k_{min}. This number is used in the inner most loop to control the selection of k which is randomly sampled between n_p and k_{max}. Then the parameter τ is set to $\frac{n_p}{k}$.

Now the normal estimation begins by randomly selecting an unprocessed point p from the point cloud and applying the estimation procedure as described in IV-A. For each point in the neighborhood of p the distance to the local surface is computed. If this distance is smaller than $d_{\max}$ and also

smaller than all distances between $\vec{p}$ to previously found surface approximations then the normal information for $\vec{p}$ is computed and stored. These steps are repeated until all points are processed or the success rate sr drops below 50%.

```
k_min :=12, k_max := sqrt(num points in point cloud)
find initial d_max
counter c := 0
do {
   for( n_p: k_max -> k_min in ten steps) {
        do {
            random select k between n_p and k_max
            τ := n_p / k
            random select an unprocessed point p
            ransac_estimation (p, τ, k, d_max) -> success rate, local surface

            for all q_i in neighborhood of p
            {
                 d_i := distance to local surface
                 if(d_i < d_max &&  local surface closer to q_i than all
                    previously found local surfaces )
                 {
                     derive new normal information for q_i
                 }
            }
        } while (num unprocessed points > 0 && success rate > 0.5)
    }
     d_max := initial d_max * c;
  while (num unprocessed point > 0)
```

Fig. 5: Pseudo code describing the parameter adjustment procedure.

C. Denoising

If the data points are corrupted by noise we propose to apply a feature preserving filtering technique to the points before meshing. Because providing high quality normal information is the most import step, the main work here is already done by our estimator. The simplest option to correct noisy point positions is the projection of the points onto their locally estimated surfaces. We also implemented the bilateral position filtering described in [JDD03] and [JDZ04]. Here the new position of a point $\vec{p}$ is computed as the weighted mean of all projections of point $\vec{p}$ onto the local surface approximations in the neighborhood of $\vec{p}$. The bilateral weights have a similar structure as our weights (compare 3) used in the normal estimation procedure. They penalize the distance between $\vec{p}$ and the neighboring point on the one hand and the distance between $\vec{p}$ and its projection onto the local surface approximation corresponding to the neighboring point $\vec{q}$ on the other hand (see Figure 6 for a small example). The latter can be different from the local surface of $\vec{p}$ if $\vec{q}_i$ lies on a different surface sheet e.g. on another side of a sharp edge.

V. Surface Orientation

To provide a consistent normal orientation we use an extended version of Hoppe's normal orientation algorithm[HDD*92], because this algorithm is not restricted to closed surfaces, like other proposed volumetric approaches. Hoppe's algorithm starts by creating a weighted symmetrized knn-graph over all points in the point cloud $\mathbf{P}$. The idea is to start from a node in the graph with known normal orientation and to propagate this knowledge along the edges of the graph. In the original version each edge e_{ij} connecting the neighboring points $\vec{p}_i$, $\vec{p}_j$ with normals $\vec{n}_i$, $\vec{n}_j$ is assigned a weight w_{ij} with a value of $1 - | < \vec{n}_i, \vec{n}_j > |$. The weighting function is designed to produce a large value for edges connecting two points with large angular normal difference

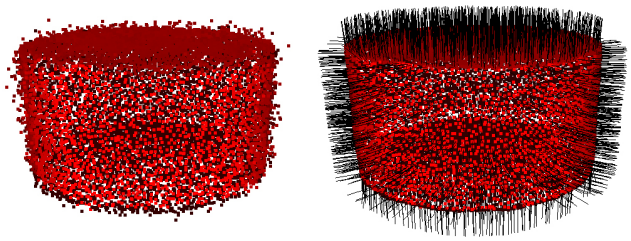


Fig. 6: Left: point cloud of a sampled cylinder with radius of 0.5. 40 % of points are corrupted by Gaussian noise with $\sigma = 0.02$ in normal direction. Right: Same point cloud after applying the bilateral filter using the normals of our RANSAC estimator.

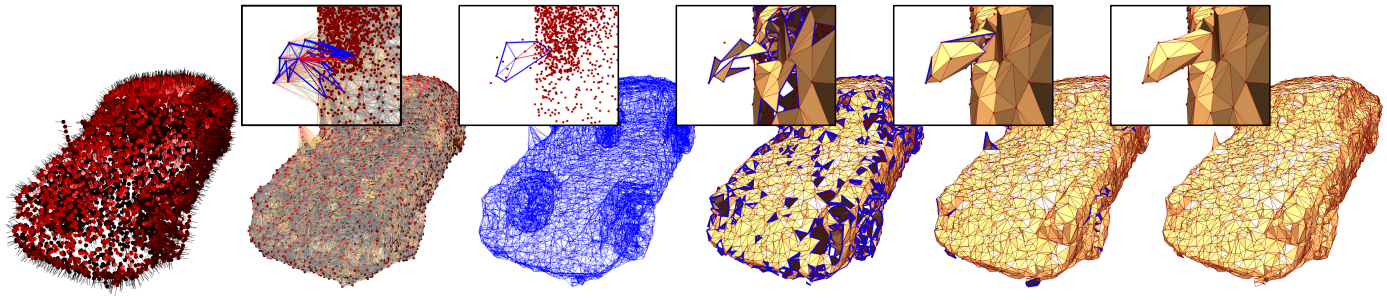
and a low value for edges connecting points with nearly parallel normal directions. Then the propagation is done along the edges of the minimal spanning tree (MST) of the original graph. While propagating normal orientation along low curvature is in principle a good idea, this algorithm often fails on meshes consisting of several sheets which are only touching each other on sharp edges e.g. a tetrahedron.
The first improvement we propose is to start the propagation from multiple points of known orientation instead of just one. For closed surfaces often many known initial normal orientations can be found automatically by computing the convex hull of all points. The normals of each point on the convex hull is initially orientated to point outside of the hull. Another way to provide initial known orientation is by user selection which typically should be the last option to simplify manual correction in case of failure.
The second improvement we propose is a modification concerning the edge weights and the way of how the correct flip of a normal is found during the propagation along an edge in the MST. Our idea is based on the observation that in case of high curvature or across a sharp feature the correct normal orientation can be found robustly by taking the location of the local mean of nearby points into account. Instead of using the sign of the dot product of the two normals $\vec{n}_i$ and $\vec{n}_j$ for flipping decisions which is a good choice in case of low curvature we use the consistent signed location (inside or outside) of the mean $\vec{m}$ with regards to the to tangent planes in case of high curvature. Therefore we define γ_i and γ_j using the following equation

$$\gamma_\alpha = \frac{< \vec{n}_\alpha, \vec{p}_\alpha - \vec{m} >}{\max(l_{ij}, ||\vec{p}_\alpha - \vec{m}||)} \tag{5}$$

where l_{ij} is the distance between the two point positions $\vec{p}_i$ and $\vec{p}_j$ of an edge. The second normal must be flipped in order to achieve a consistent orientation if the product $\gamma_i \cdot \gamma_j$ is smaller than zero. An appropriate edge weight to prefer propagation along high curvature would be $w_{ij} = 1 - |\gamma_i \cdot \gamma_j|$. Both ideas can be combined by always selecting the criterion that results in the smaller edge weight.

VI. Surface Tessellation

The goal of surface tessellation is to define a set of faces with the input points as corner vertices. For simplicity we restricted our approach to triangular faces.



a) input point cloud b) knn-graph: k=30 c) tangentially Delaunay filtered d) consistent triangles e) region grown f) holes closed

Fig. 7: Results of the five steps performed for surface tessellation. The noisy sampling of the Porsche model is very inhomogeneous. Zoomed up is the view into the left rear-view mirror, which is sampled especially low. In the zooms of b) and c) one vertex neighborhood is emphasized. Notice that the complicated hole at the mirror is closed in e) without any self-intersections.

The basic idea of our surface tessellator is similar to approaches that filter the triangles in the Delaunay tetrahedralization of the point cloud. We took a different venue though because of efficiency issues and in order to exploit the robustly estimated surface normals. Furthermore, we wanted to support sharp features and even self-intersecting sheets, where Delaunay based approaches fail. Similar to the local surface estimation we base the tessellator on the asymmetric k-nearest neighbor graph (knn-graph). The resulting surface tessellation does not depend on k as long as k is large enough to contain all directed edges needed by the tangential Delaunay filter described below. A value of $k = 30$ was sufficient even for highly inhomogeneously sampled point clouds.

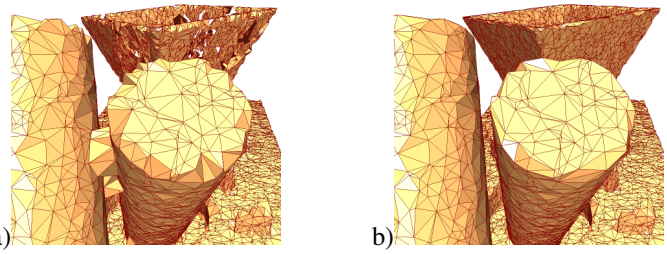
The process of surface tessellation can be interpreted as an edge filter process that removes all directed edges from the knn-graph until only the edges of the surface tessellation are left. Actually, we first tried to enumerate all loops of length 3 in the knn-graph as possible candidate triangles and pass them to a discrete optimization process similar to the one used by Adamy et al. in [AGJ02] to resolve non-manifold areas. Figure 7 b) shows that this leads to a combinatorial explosion. The knn-graph contains in the order of 300 triangles per vertex from which in average only six need to be selected. Therefore, we used a greedy approach that is decomposed into four steps which are illustrated in figure 7 c-f).

In the first step we construct for each input point in its tangential space a local Delaunay triangulation and replace the knn-neighborhood by the 1-ring neighborhood of the Delaunay tessellation. In the second step we extract all triangles that are consistent with the Delaunay filtered knn-graph. The third step grows the still incomplete surface tessellation by exploiting a priority queue (compare figure 7 e). There are still all triangles missing that get flipped when projected into the tangential space of its incident points. These triangles are introduced in the final step that closes narrow holes by minimizing surface area.

The surface tessellator uses the point normals. As for some models it is very hard to find a consistent normal orientation, we designed two versions of the surface tessellator, one that exploits normal orientation and one that ignores it. This implies that the sign of scalar products between normals as well as the order of neighbors in 1-rings have to be ignored. In the notation we distinguish the two versions with the subscripts ori and nor. This approach allows us also to handle Moebius strips and Klein bottles.

A. *Tangential Delaunay Filter*

The tangential Delaunay filter is computed separately for each point $\vec{p}$. The normal $\vec{n}_{\vec{p}}$ defines a 2d tangential space with $\vec{p}$ serving as origin. All $\vec{q}_i$ in the knn-neighborhood $N_{\vec{p}}$ are projected into tangential space. Then a 2d Delaunay tessellation is computed and the original neighborhood is replaced by the 1-ring in the 2d Delaunay tessellation.



a) b)

Fig. 8: Comparison of surface tessellations computed with a) orthogonal projection to tangential space and b) our variant based on neighbor location and normal.

The crucial part is the projection of the $\vec{q}_i$. An orthogonal projection leads to the connection artifacts shown in figure 8 a). Imagine that the tangential space is parametrized in cylindrical coordinates by an angle ϕ and a radius r. In the modified projection ϕ_i of $\vec{q}_i$ is still computed from the orthogonal projection but the radius is computed according to

$$r_i = ||\vec{q}_i - \vec{p}|| / \omega\left(\langle \vec{n}_{\vec{p}}, \vec{n}_{\vec{q}_i} \rangle\right), \tag{6}$$

where ω measures how well the normals at $\vec{p}$ and $\vec{q}_i$ fit together based on the cosine $c = \langle \vec{n}_{\vec{p}}, \vec{n}_{\vec{q}_i} \rangle$ of the angle between the normals. 0 is returned for orthogonal normals and 1 for parallel normals. For the radius this implies that a neighbor with a very similar normal is mapped to its 3d distance, whereas neighbors with different normals are mapped further away. Far away

points are less probable to find their way into the Delaunay 1-ring of point $\vec{p}$, but they still allow to close the fan completely if no better neighbor can be found in a similar direction.

For the definition of the normal quality one has to distinguish between the oriented and non oriented case:

$$\omega_{ori}(c) = \frac{1 + \mathrm{erf}\left(\frac{qc}{2}\right)}{2} \qquad \omega_{nor}(c) = \frac{1 + \mathrm{erf}\left(\frac{q(2c^2-1)}{2}\right)}{2}. \tag{7}$$

In both cases we used the error function erf with the

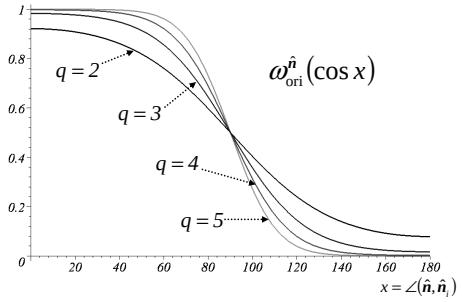


Fig. 9: Plot of $\omega_{ori}(c)$ for different values of q.

additional parameter q to tolerate small normal deviations but significantly penalize large deviations. Figure 9 plots $\omega(s)$ for different values of q. $\omega(c)$ is plotted over angle deviation which runs from 0 to 180 degrees in the oriented case and from 0 to 90 degrees otherwise. In both cases the resulting plot families are identical. In our experiments it turned out that $q = 3$ worked perfectly in figure 9 b) and all our examples. Changing q between 2 and 5 did not affect the resulting tessellation significantly.

The 1-ring of the local Delaunay tessellation can be computed quite efficiently. One first sorts the neighbors cyclically with respect to ϕ_i and checks whether there is a jump of more than 180 degrees. In this case the $\vec{p}$ and the edges to the corresponding neighbors are part of the convex hull and won't be touched by the following edge flipping process anymore. For all remaining edges we perform a local Delaunay check, i.e. we check whether the circumcircle of the triangle to the left contains the distant neighbor of the triangle to the right. We can even simplify the resulting computation by exploiting the fact that $\vec{p}$ is the origin of the tangential space. If the local Delaunay test fails, the edge can simply be removed. Thus we can finally construct the Delaunay 1-ring by removing all neighbors whose edges fail the local Delaunay test.

To avoid numerical problems with degenerate or very regular point locations we jitter the points before performing the tangential Delaunay filter with a noise amplitude of one percent of the distance to nearest neighbor. Furthermore, we discarded all neighbors that project to the origin and selected among neighbors that project to the same ϕ_i only the one with smallest r_i.

The filtered neighbor graph is not symmetric and might even miss some directed edges. The following steps of the tessellator will remove further edges and may also introduce some directed edges back to the 1-rings. To avoid closing large holes in this process we set for each point the maximum length $l_{\max}(\vec{p})$ of reinserted directed edges to

$$l_{\max}(\vec{p}) = h \cdot \max_i ||\vec{q}_i - \vec{p}|\,|, \tag{8}$$

where h is a parameter that should be chosen between 2 and 5. In our examples we chose $h = 2$ for homogeneously sampled point clouds and $h = 4$ for inhomogeneous samplings. Our approach also works on homogeneous samples with $h = 4$, but it would then close relatively large holes.

B. Search for Consistent Triangles

The tangential Delaunay filter reduced the neighbor graph significantly as shown in figure 7 c). Now a potential 1-ring has been constructed for each point. A point $\vec{p}$ with neighbors $\vec{q}_i, i \in N_{\vec{p}}$ would like to have all triangles $(\vec{p}, \vec{q}_1, \vec{q}_2), \ldots, (\vec{p}, \vec{q}_{n-1}, \vec{q}_n), (\vec{p}, \vec{q}_n, \vec{q}_1)$ in the final tessellation. But the 1-rings of neighbors $\vec{q}_i$ might demand for different triangles. In flat areas of the surface all points agree simply on the triangles of a Delaunay tessellation, but in 3d this is not necessarily the case.

In this filter step we find triangles t consisting of three points $\vec{p}_1(t)$, $\vec{p}_2(t)$ and $\vec{p}_3(t)$, such that in each 1-ring of the three points $\vec{p}_j(t)$ the other two points $\vec{p}_{(j+1)\%3}(t)$ and $\vec{p}_{(j+2)\%3}(t)$ are adjacent 1-ring neighbors. In the orientated case we also check for consistent 1-ring orders. The found triangles are called consistent. Figure 7 d) shows the consistent triangles for the Porsche model. The remaining boundary edges around holes are marked in blue.

We explicitly allow consistent triangles to intersect each other in order to support self-intersecting surface sheets. In the next step we disallow introduction of further self-intersections. Finally the hole filling process is allowed to close a hole with self-intersections if it there is no possibility without.

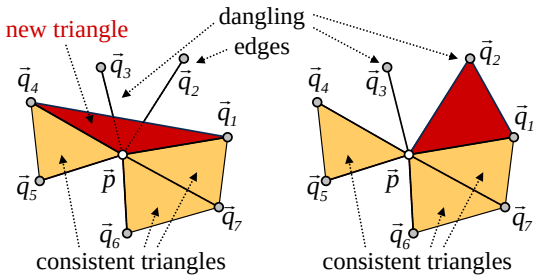
C. Priority Queue Based Growing

The third step in surface tessellation is a typical region growing algorithm. We define events that incorporate further triangles into the set of consistent triangles one at a time. After adding a new triangle we ensure that it is consistently incorporated to the 1-rings of its incident points.

For each grow event we compute the quality of the newly introduced triangle based on the number of missing directed edges in the current neighbor graph and the geometric triangle quality. All events inserted into a priority queue and executed in order of decreasing triangle quality. After each event execution the events with changed triangle quality are discarded and all possible new events inserted to the queue.

The goal of the growing step is to complete all 1-rings with consistent triangles. Therefore we build the event queue by iterating all 1-rings. Figure 10 shows the two types of grow events that we allow: corner and edge grow events. The corner grow event closes one opening in a 1-ring and removes the dangling edges inside the opening. The edge grow shrinks an opening by one triangle. The new triangle must be incorporated also into the 1-rings of the incident neighbors. In figure 10 a) these are $\vec{q}_1$ and $\vec{q}_4$ and in b) $\vec{q}_1$ and $\vec{q}_2$.

For all consistent triangles we know that they are part of the 1-rings of their incident vertices. Therefore in figure 10 a)



a) corner grow event b) edge grow event

Fig. 10: The different grow events used to increase the number of consistent triangles.

the directed edge $(\vec{q}_1, \vec{p})$ must already be in the neighborhood of $\vec{q}_1$. We only need to check for the edge $(\vec{q}_1, \vec{q}_4)$ in the 1-rings of $\vec{q}_1$ and $\vec{q}_4$. If the edge does not exist, we try to insert it into the 1-rings of the neighbors. This is not possible if the inner angles of the newly inserted triangle are blocked in the 1-rings of $\vec{q}_1$ or $\vec{q}_4$ by previously added consistent triangles. Further consistency checks are that the new triangle must not

- flip its orientation in any of the incident tangential spaces (only oriented case)
- introduce a self-intersection
- introduce a non-manifold edge

If one of the checks fails, the grow event is discarded. For the manifold check we simply count the number of triangles per edge. The self-intersection is check by collecting the potentially colliding triangles with a kd-tree and then performing a triangle-triangle intersection test. We modified the approach of [DG02] by a special treatment for the case of corner or edge adjacent triangles.

It remains to define the triangle quality $Q(t)$ used to prioritize the queue. It is on the number $n_{ins}(t)$ of to be inserted directed edges, the minimal interior angle $\alpha_{\min}$ of the triangle and the quality of the triangle normal $\vec{n}(t)$ with respect to the normals at the incident points

$$Q(t) = \frac{3 - n_{ins}(t)}{2} + \alpha_{\min}(t) + \min_{j \in \{1,2,3\}} \omega\left(\langle \vec{n}_{\vec{p}_j(t)}, \vec{n}(t) \rangle\right)$$

3 is the maximum value for n_{ins} and can arise in an edge grow event. In figure 10 b) both directed edges from $\vec{q}_2$ and one from $\vec{q}_1$ might need to be added when introducing the new triangle. Thus the connectivity term in the definition of $Q(t)$ ranges from 0 to $3/2$. The largest value for $\alpha_{\min}$ is $\pi/3 \approx 1$ and the normal quality lies in the range $[0, 1]$. Therefore, connectivity and geometry contributions to $Q(t)$ of similar size.

The region growing step closes most holes and only misses triangles that cannot be added due to self-intersections, non-manifold creation or triangle flips in tangential space. There are actually several triangles of the latter kind that can be added with the final hole fill step.

All points that are still disconnected after the region growing step are classified as outliers and discarded.

D. Filling Holes

Holes can be easily found by navigating through the 1-rings of the vertices. As we only added consistent triangles all holes must be closed. The reason is that for each incoming edge of a 1-ring a unique outgoing edge is found by following the entered opening. The holes can be non-manifold though. A non-manifold hole passes through several openings of the same 1-ring. This can be easily detected by sorting the indices of the points on the boundary of the hole into a set data structure and comparing the size of the set with the length of the hole. If both are equal, the hole is manifold and non-manifold otherwise.

We resolved the non manifold holes by removing sets of connected triangles in the non manifold 1-rings of vertices, through that non manifold holes pass repeatedly, until only one valid fan of triangles was left. We favored removal of small sets of connected triangles with long edges in this non manifold resolution process.

The remaining non-manifold 1-rings with several openings become manifold if all holes are filled. As there is no guarantee that each hole can be filled, we split the hole filling procedure into a test and a tessellation step. With the test method we check for each remaining non-manifold 1-ring whether all holes at its openings can be closed. Actually, one hole may remain. If two or more holes remain, we resolve the non manifold 1-ring as described above.

After the cleaning of all non-manifold 1-rings we close the closeable holes resulting in a surface tessellation that is always a topological manifold with boundary, although it can have geometric self-intersections.

Our hole tessellation itself is based on an ear-cutting strategy. Ears are formed by three successive points along the hole and define a triangle that cuts away the point in the center of the ear. We allow the cutting of an ear with center $\vec{p}$ only if

- the newly introduced edge is not longer than the maximum allowed length $l_{\max}(\vec{p})$ (see equation 8) and
- the ear triangle does not intersect any other triangle.

The second constraint is turned off in a second attempt when self-intersecting surfaces are considered.

We implemented an exhaustive search for a valid hole tessellation with minimal surface area as well as a greedy approach that tries the ears in the order of increasing length of the newly introduced edge. The greedy approach stops as soon as the first valid triangulation has been found. The greedy approach typically finds a hole with small surface area but is not as good as the exhaustive search. We therefore filled all holes with less than 10 vertices with the exhaustive search and the rest with the greedy approach.

VII. Results

A. Datasets and Basic Results

We use uniform random sampling and homogenous sampling techniques to create synthetic test cases. The homogenous sampling is done by a dart throwing method, which produce random noise with blue noise characteristics. Noisy

models are created by adding synthetic Gaussian noise in normal direction in a given percentage with zero mean and a given standard deviation.

B. Quality of Local Surface Estimation

To measure, test and compare the quality of our local surface estimator we generated several synthetic test cases with known ground truth normal information. The test cases and the ground truth information is provided by taking random samples from existing surface meshes.

For each test case the mean and the standard deviation of the absolute value of the angular error in degrees is computed. Additionally, the angular error is color mapped onto the points to get a better impression of the localization of normal errors (see Figure 11). In a) the result of applying a weighted least squares fit (WLS) of a paraboloid with a relatively small neighborhood of $k = 13$ on the first test case is shown. In planar regions and in regions of high degree of curvature good results can be achieved. Sharp features typically cannot be approximated sufficiently and produce larger errors.

The result of our method is depicted in b) the overall error is smaller than in a). Typically a small number of points nearby sharp edges can be found with a relatively high error. This can be explained by the fact that our RANSAC estimator has classified these points belonging to the other incident surface sheet than the point was originally sampled from. This also explains the relative high standard deviation in the error (see Table I). In practical cases these normals can be assumed to be correct.

In the second test case the WLS method has a better mean error due to the large smooth regions but fails to correctly estimate the normals close to the self intersection, what can also be recognized in the large standard deviation of the absolute angular error.

The third test case is a more illustrative example showing the problems of the WLS method if k is chosen to high.

C. Surface Reconstruction Performance

We compare our surface reconstruction results with Poisson surface reconstruction [KBH06], cocone [DG01], tight cocone (tconone) [DG03] and robust cocone [DG04]. We chose these schemes due to their availability and state-of-the-art performance.

We performed a comparison on six data sets with all kinds of difficulties. Their size, the number of triangles and holes in our reconstruction and the running times are tabulated in Table II. All point clouds except the laser range scan of the church were sampled uniformly or homogeneously from polygonal meshes in order to be able to compare to the ground truth. Synthetic noise was add to the Porsche point cloud yielding the most difficult reconstruction problem. We actually used the original surface normals in this case such that comparison to the cocone algorithm is not quite fair as it cannot exploit the original normals. We always fed the Poisson surface reconstructor with the same normals as our scheme. In all examples we chose $k = 2$ and $q = 3$. The maximum length parameter h was set to 2 for homogeneous sampling and to 4 for uniform samplings. For the Klein Bottle we ignored normal orientation.

The resulting reconstructions are compared visually in Figure 12. The first row of Figure 12 compares the two differently sampled camel models. The close up of tail and legs were taken from the uniform sample and the head from the homogeneous sample. Our approach is the only one that reconstructs the tail and the hooves, keeps the knees in separate sheets and even reconstructs the mouth with extremely close sheets perfectly. Please notice that the right head is not the original mesh but our reconstruction. We always chose the best result of the three cocone approaches in this visual comparison. The other approaches typically worked significantly worse.

The second row shows the noisy test case. Our approach reconstructs the original features of the surface best. The right most image shows our reconstruction on the point cloud without noise. The third row illustrates that our approach works robustly also on large real world data and that it can handle the complicated hole structure illustrated in the first two images. The Poisson reconstructor was ignored as it cannot detect boundaries. The two right most images compare our results to the cocone approach that connects distant holes with large triangles.

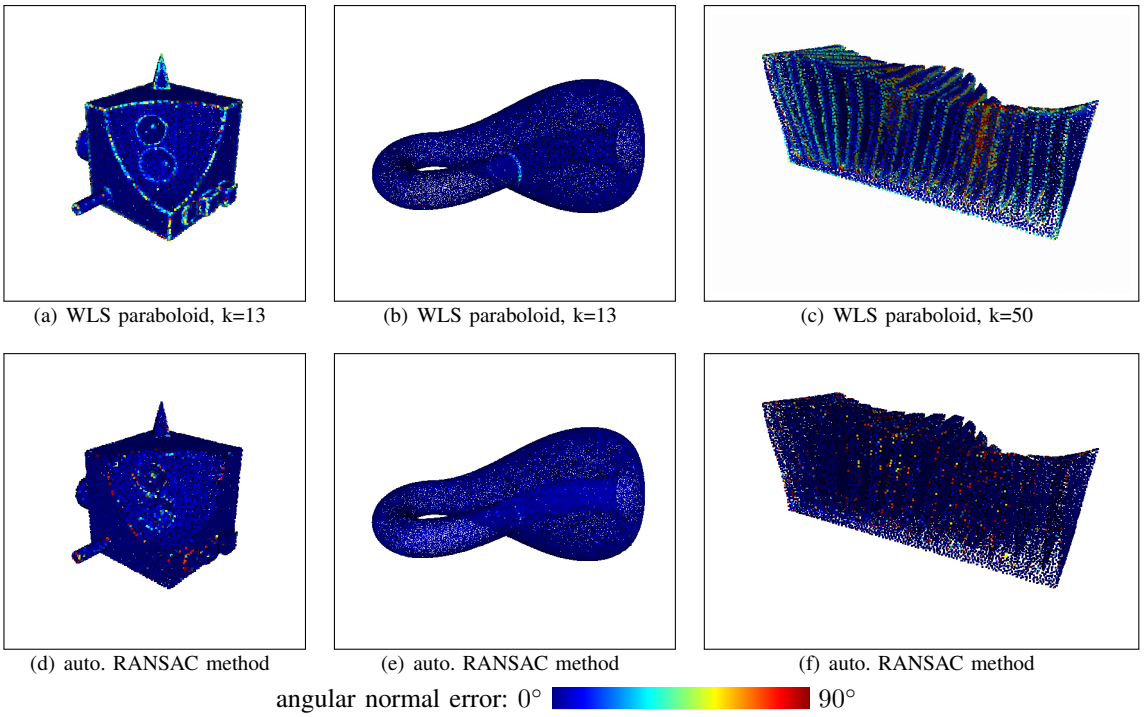
The last row shows the gear model on the left where the Poisson reconstructor fails even on a homogeneously sampled point cloud with original normals. The tight cocone nearly worked well, but again only our scheme did not connect nearby surface sheets. Finally, we show on the right the results on the Klein Bottle, which also worked on a global scale for the cocone algorithm but failed at the self-intersection. Our scheme works perfectly and sews together the crossing layers perfectly.

Table II tabulates the running times on a Pentium 5 with 2GHz and 1GB main memory. The time for local surface estimation depends on the amount of flat areas in the point cloud and typically grows sub-linear in the number of input points. The running time for the surface tessellator is mostly linear in the number of points with a performance in the order of 10,000 points per second. Compared to Poisson surface reconstruction we are faster for small data sets and a bit slower for larger ones. But our results are far more detailed. We only give timings for the tight cocone as this was the fastest among the three Delaunay based approaches, but its performance is far slower than ours.

VIII. Conclusions

We proposed a new surface reconstruction pipeline that can handle inhomogeneous samplings, noise, sharp features, nearby surface sheets, self-intersections, boundaries and non orientable surfaces with minimal user input. The runtime performance allows to process very large point clouds. Both our denoising / normal estimation as well as the surface tessellator are important contributions by themselves.

In future work we plan to add further stages to our pipeline. We think of a stage that samples the undersampled sharp features by a technique similar to the edge sharpener [AFRS03] but based on vertex normals. Also an edge flip optimizer and an edge collapse simplifier can make direct use of the vertex normals. Finally, we want to implement a streaming version of our technique to handle huge point clouds.



(a) WLS paraboloid, k=13 (b) WLS paraboloid, k=13 (c) WLS paraboloid, k=50

(d) auto. RANSAC method (e) auto. RANSAC method (f) auto. RANSAC method

Fig. 11: Comparison of estimated normals to ground truth. The left test case contains sharp features like edges corners and spikes and several sheets with different curvatures. In the middle a non-orientable surface is used. The critical regions lie in the areas near to the self-intersecting surface sheets. On the right side a test case is depicted which illustrates the problems produced by several nearby surface sheets.

	left test case		middle test case		right test case	
used method	μ	σ	μ	σ	μ	σ
WLS fit of paraboloid	7.49847	14.7163	1.4135	3.64514	14.7122	19.9111
our RANSAC estimation	2.58256	10.9735	1.84496	2.72814	3.82579	13.8575

TABLE I: Error comparison of estimated normals

model	points	triangles	holes	t_{local}	t_{tess}	t_{ours}	$t_{poisson}$	$t_{tcocone}$
camel homogeneous	28926	57848	0	5.4	2.7	8.1	7.2	32.3
camel uniform	12000	24000	0	3.4	1.2	4.6	5.1	11.6
Porsche noisy	8001	15920	1	2.7	1.2	3.9	2.9	7.1
gears uniform	40000	57848	0	7.9	4.5	12.4	10.8	49.4
church scan	193601	383590	85	24.2	31.5	55.5	19.8	532.7
Klein Bottle	50400	100800	0	7.7	4.6	7.7	11.1	73.2

TABLE II: Size of data sets and runtime comparison for local surface estimation, surface tessellation, our approach, Poisson surface reconstruction and tight cocone.

REFERENCES

[AB99] AMENTA N., BERN M.: Surface reconstruction by voronoi filtering. In *SCG '98: Proceedings of the fourteenth annual symposium on Computational geometry* (New York, NY, USA, 1999), ACM, pp. 39–48.

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

[ACK01] AMENTA N., CHOI S., KOLLURI R. K.: The power crust. In *SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications* (New York, NY, USA, 2001), ACM, pp. 249–266.

[AFRS03] ATTENE M., FALCIDIENO B., ROSSIGNAC J., SPAGNUOLO M.: Edge-sharpener: recovering sharp features in triangulations of non-adaptively re-meshed surfaces. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 62–69.

[AGJ02] ADAMY U., GIESEN J., JOHN M.: Surface reconstruction using umbrella filters. *Comput. Geom. Theory Appl. 21*, 1 (2002), 63–86.

[AMN*98] ARYA S., MOUNT D. M., NETANYAHU N. S., SILVERMAN R., WU A. Y.: An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM 45*, 6 (Nov. 1998), 891–923.

[DG01] DEY T., GIESEN J.: Detecting undersampling in surface reconstruction. In *SCG '01: Proceedings of the seventeenth annual symposium on Computational geometry* (New York, NY, USA, 2001), ACM, pp. 257–263.

[DG02] DEVILLERS O., GUIGUE P.: *Faster Triangle-Triangle Intersection Tests*. Research Report 4488, INRIA, 2002.

[DG03] DEY T., GOSWAMI S.: Tight cocone: A water-tight surface

reconstructor. *Proceedings of the 8th ACM Symposium on Solid Modeling and Applications* (2003).

[DG04] DEY T. K., GOSWAMI S.: Provable surface reconstruction from noisy samples. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry* (2004), pp. 330–339.

[DMSB99] DESBRUN M., MEYER M., SCHRÖDER P., BARR A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 317–324.

[DMSB00] DESBRUN M., MEYER M., SCHRDER P., BARR A. H.: Anisotropic feature-preserving denoising of height fields and bivariate data. In *In Graphics Interface* (2000), pp. 145–152.

[DS05] DEY T. K., SUN J.: An adaptive mls surface for reconstruction with guarantees. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2005), Eurographics Association, p. 43.

[FB81] FISCHLER M. A., BOLLES R. C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24*, 6 (June 1981), 381–395.

[FCOS05] FLEISHMAN S., COHEN-OR D., SILVA C. T.: Robust moving least-squares fitting with sharp features. *ACM Trans. Graph. 24*, 3 (2005), 544–552.

[FDCO03] FLEISHMAN S., DRORI I., COHEN-OR D.: Bilateral mesh denoising. *ACM Trans. Graph. 22*, 3 (2003), 950–953.

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MC-DONALD J., STUETZLE W.: Surface reconstruction from unorganized points. In *Proceedings of ACM SIGGRAPH 1992* (1992), vol. 26, pp. 71–78.

[JDD03] JONES T. R., DURAND F., DESBRUN M.: Non-iterative, feature-preserving mesh smoothing. *ACM Trans. Graph. 22*, 3 (2003), 943–949.

[JDZ04] JONES T. R., DURAND F., ZWICKER M.: Normal improvement for point rendering. *IEEE Comput. Graph. Appl. 24*, 4 (2004), 53–56.

[JWB*06] JENKE P., WAND M., BOKELOH M., SCHILLING A., STRAER W.: Bayesian point cloud reconstruction. *Comput. Graph. Forum 25*, 3 (2006), 379–388.

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Symposium on Geometry Processing* (2006), pp. 61–70.

[KSO04] KOLLURI R., SHEWCHUK J. R., O'BRIEN J. F.: Spectral surface reconstruction from noisy point clouds. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (New York, NY, USA, 2004), ACM, pp. 11–21.

[Lev03] LEVIN D.: Mesh-independent surface interpolation. *Advances in Computational Mathematics* (2003).

[MNG04] MITRA N. J., NGUYEN A., GUIBAS L.: Estimating surface normals in noisy point cloud data. In *Internat. J. Comput. Geom. & Applications* (2004). p. to appear.

[OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.-P.: Multi-level partition of unity implicits. *ACM Trans. Graph. 22*, 3 (2003), 463–470.

[PKKG03] PAULY M., KREISER R., KOBBELT L., GROSS M.: Shape modeling with point-sampled geometry. In *Proceedings of ACM SIGGRAPH 2003* (2003), ACM Press, pp. 641–650.

[TL94] TURK G., LEVOY M.: Zippered polygon meshes from range data. *Computer Graphics(SIGGRAPH'94 Proceedings)* (1994), 311–318.

[TM98] TOMASI C., MANDUCHI R.: Bilateral filtering for gray and color images. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision* (Washington, DC, USA, 1998), IEEE Computer Society, p. 839.

[TZ00] TORR P. H. S., ZISSERMAN A.: Mlesac: a new robust estimator with application to estimating image geometry. *Comput. Vis. Image Underst. 78*, 1 (2000), 138–156.

[Whi98] WHITAKER R. T.: A level-set approach to 3d reconstruction from range data. *Int. J. Comput. Vision 29*, 3 (1998), 203–231.

[ZOF01] ZHAO H.-K., OSHER S., FEDKIW R.: Fast surface reconstruction using the level set method. In *VLSM '01: Proceedings of the IEEE Workshop on Variational and Level Set Methods (VLSM'01)* (Washington, DC, USA, 2001), IEEE Computer Society, p. 194.

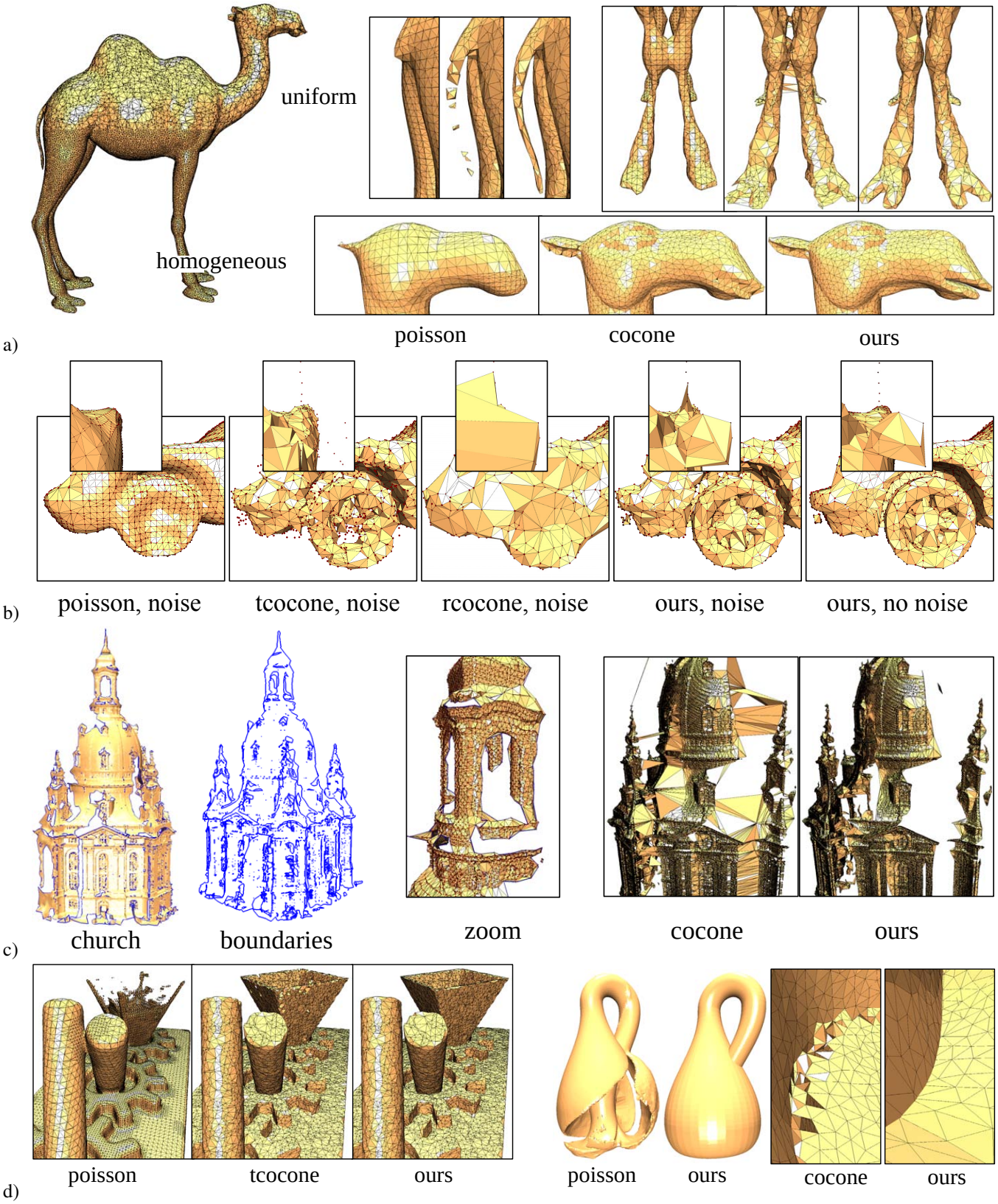


Fig. 12: Visual comparison of our approach to Poisson surface reconstruction [KBH06] and the family of cocone algorithms [DG01], [DG03], [DG04]. The camel model in the top row was sampled homogeneously (bottom half) and uniformly (top half). All close up triples compare Poisson, cocone and ours. In the last row the first three images show the gear model and the other the Klein Bottle with a close up of the self-intersection on the right.