Extracting lines of curvature from noisy point clouds

Evangelos Kalogerakis, Derek Nowrouzezahrai,
Patricio Simari, Karan Singh

DGP lab, Department of Computer Science, University of Toronto
40 St. George St., Toronto, Ontario, Canada M5S 3G4

Abstract

We present a robust framework for extracting lines of curvature from point clouds. First, we show a novel approach to denoising the input point cloud using robust statistical estimates of surface normal and curvature which automatically rejects outliers and corrects points by energy minimization. Then the lines of curvature are constructed on the point cloud with controllable density. Our approach is applicable to surfaces of arbitrary genus, with or without boundaries, and is statistically robust to noise and outliers while preserving sharp surface features. We show our approach to be effective over a range of synthetic and real-world input datasets with varying amounts of noise and outliers. The extraction of curvature information can benefit many applications in CAD, computer vision and graphics for point cloud shape analysis, recognition and segmentation. Here, we show the possibility of using the lines of curvature for feature-preserving mesh construction directly from noisy point clouds.

Key words: lines of curvature, robust curvature estimation, point cloud denoising, outlier rejection, quad mesh construction

1 Introduction

Incorporating physical objects that have been scanned into a digital form is an integral part of many engineering and entertainment applications. The raw output of most shape acquisition methods is a point cloud sampling of the scanned surface. Given the increasing popularity of point cloud based

Email address: {kalo, derek, psimari, karan}@dgp.toronto.edu.

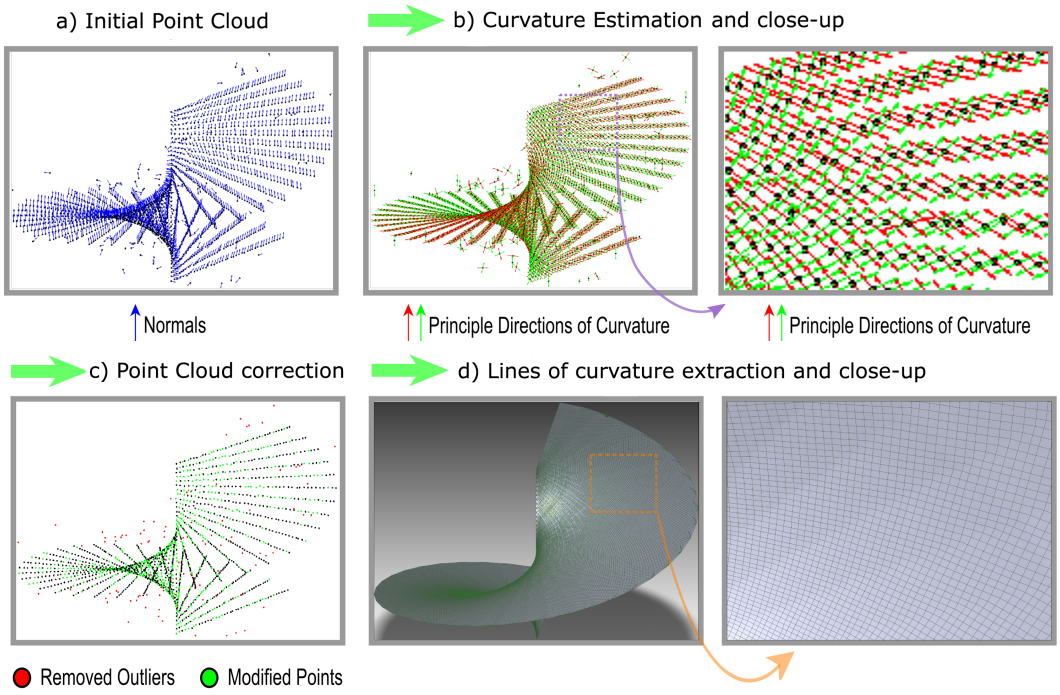


Fig. 1. Overview of the stages of our method. **(a)** Initial point cloud and normal estimates. **(b)** Robust statical estimation of curvature (principal directions are visualized here) **(c)** De-noising of normal estimates and point locations **(d)** Extraction of lines of principal curvature over the point cloud (the result is visualized as a set of faces created by the intersections of the lines).

techniques, the robust estimation and use of surface derivatives, such as curvatures, for point cloud geometry processing is an active area of research. Surface derivatives such as normal and curvature are as important as surface position for the perception, understanding, interrogation of shape (30; 26), registration (42), smoothing (27) and symmetry detection (34) to name a few applications. Aggregates of these attributes are referred to in design as *surface features*. Unfortunately, different shape acquisition processes produce a wide range of characteristic point clouds that commonly exhibit artifacts of irregular sampling, noise and outliers, and make automatic and general purpose differential surface attribute estimation a challenging problem.

In this article, we provide a robust statistical framework to compute surface curvature in point clouds and we show how this framework can be exploited to automatically reject outliers and denoise point clouds. This statistical framework acquires maximum likelihood estimates of curvature through an Iteratively Reweighted Least Squares (IRLS) approach which refines the shape and size of each neighborhood around every point by weighting samples appropriately based on fitting error. The method automatically adapts to small neighborhoods for well conditioned surfaces, and larger, possibly anisotropic neighborhoods in the presence of noise, irregularities and feature boundaries. The robustly estimated curvatures and the statistical weights are used to cor-

rect the surface normals. Then, through a global energy minimization, the point cloud shape is denoised. We follow with the extraction of lines of curvature directly on the point cloud within our statistical framework to preserve surface derivative discontinuities, e.g., near sharp edges. Our approach can be applied to surfaces of arbitrary genus with or without boundaries.

As curvature is a fundamental descriptor for shape analysis (30), we believe that our framework is useful for many applications in CAD, graphics and vision and can also trigger interesting research directions in the field of point-based processing techniques. In this article, we also show the possibility of using our method for directly constructing quad-dominant meshes from point clouds. The advantage of such curvature-aligned construction is the exploitation of the theoretical argument that an optimal piecewise linear approximation of a smooth surface can be built (at least in non-hyperbolic regions) if the mesh edges are aligned with the lines of curvature (45; 5). Moreover, curvature-aligned meshes mimic the flow-lines along which artists usually place geometric elements to create 3D models (1) or hatch strokes for model illustration (16).

A schematic overview of our methodology is shown in Figure 1. The input to our technique is a point cloud with oriented normals. Oriented normal vectors can often be acquired as part of the scanning process or can be estimated with existing algorithms (17; 35; 2). Then, our approach computes curvature robustly in the presence of noise and outliers in both points and normals. Our algorithm corrects noisy normals and the point locations by energy minimization, while also rejecting outliers. We then use a Voronoi space partition to efficiently trace lines of principal curvature and their intersections with a specified density directly over the surface implied by the corrected point cloud.

2 Related work

Our work is related to lines of curvature generation on discretized surfaces. To the best of our knowledge, there has not been any attempt to trace lines of curvature directly on point clouds. There are a number of papers that discuss lines of curvature for shape representations, like polygon meshes or B-spline surfaces. Martin (32) was the first to introduce the idea of principal patches whose sides are lines of curvature for use in Computer Aided Geometric Design. Dupin's cyclide patches, whose lines of curvature are all circular arcs, are used for blending surfaces (39; 10). Maekawa *et al.* (30) discuss the use of lines of curvature and umbilics for shape recognition and feature extraction. Alliez *et al.* (1) propose explicit remeshing on an existing polygonal object representation so that lines of minimum and maximum curvatures are used to determine the edges for the remeshed version in anisotropic regions. In order to track the lines of curvature, the initial mesh is globally parameterized, while

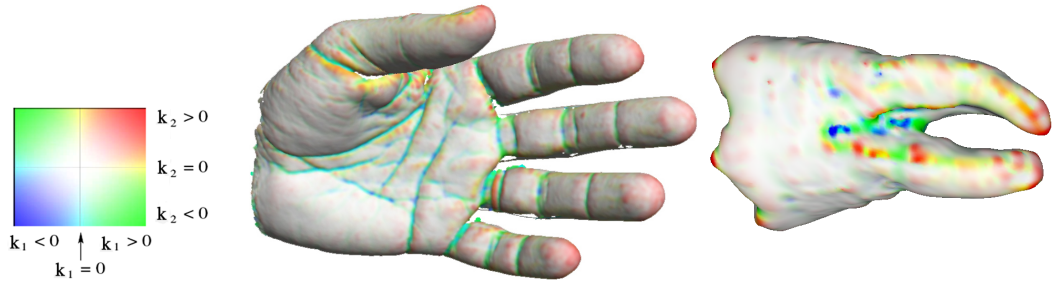


Fig. 2. Visualization of principal curvature values obtained by our method for a scanned point cloud model of a hand and a tooth. The visualization is based on the color scheme suggested in (41). Normals are estimated by locally fitting planes (17). Renderings are created with PointShop3D.

the curvature tensor field also needs to be pre-smoothed. Marinov and Kobbelt (31) instead provide a more efficient framework that does not rely on a global parametrization for anisotropic remeshing. Lai *et al.* (25) apply an iterative relaxation scheme which incrementally aligns the mesh edges to the principal directions without the use of global parametrization of meshes.

Regular remeshing can be achieved in terms of globally parameterizing the mesh where the parameter lines can also be guided by a given frame field, for example by principal curvature frames (40; 20). Liu *et al.* (29) use principal directions to create conical meshes. Other remeshing techniques have also been proposed using smooth harmonic scalar fields (9) or Laplacian eigenfunctions (8; 50).

In our case, we deal with the much more difficult and challenging problem of handling noisy point clouds surfaces of arbitrary topology with outliers as well as fine and sharp features. In our case, we make no demands regarding pre-meshing or globally parameterizing the point cloud surface. The curvature information can be directly exploited by other point cloud based techniques e.g., for registration (41), smoothing (27), and shape recognition and segmentation (18) which do not depend on intermediate reconstruction techniques and need to work directly on point clouds surfaces. In this paper, we also show how to use the extracted curvature information for feature-preserving mesh construction.

Our framework relies heavily on the robust estimation of curvature on point clouds. There is a considerable number of papers in the computer graphics, vision and engineering literature concerning differential operators on discretized surfaces, especially polygon meshes (see (12) for a recent survey). Briefly, curvature estimation methods can be categorized as follows: *a) curve and patch fitting methods* where low-order curves or patches are fitted locally at each point of the surface, typically as height functions e.g., (14; 13; 37), *b) discrete differential geometry methods* where discrete versions of differential geometry theorems are developed and applied to one-ring, two-ring or N-ring neigh-

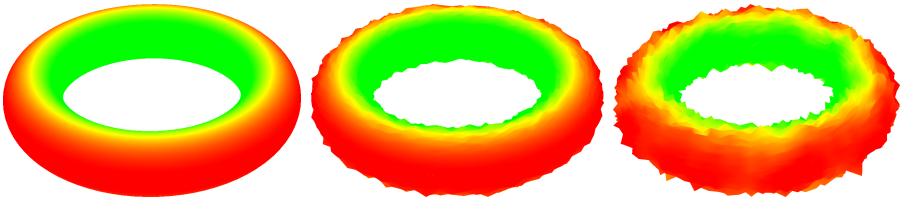


Fig. 3. Visualization of principal curvature values obtained by our method for analytic examples of torus point clouds using the same color scheme as in Figure 2. *From left to right:* Non-noisy torus, torus with random gaussian noise of variance 0.5% of the bounding box diagonal, torus with random gaussian noise of variance 1% of the bounding box diagonal.

borhoods around each vertex of a polygon mesh e.g., (47; 33; 4) and *c) Per triangle curvature tensor estimation* where the second fundamental form is fitted per each surface element (41). Kalogerakis *et al.* presented a method to estimate principal curvature values and directions over polygon meshes and point clouds using a robust statistical framework (22). In contrast to previous approaches, this approach adapts to noise, irregularities and non-uniform sampling and provides maximum likelihood estimates per surface point. This method has been shown to have an order of magnitude less error than other state-of-the-art approaches.

In this paper, we will overview and update this method and we will extend it in order to remove outlier points and denoise point positions in a principled fashion. Based on these estimates and statistical weights, we will robustly extract the lines of curvature directly from point clouds.

3 Statistical estimation of curvature

In (22), it was shown that an IRLS process, in the context of robust M-estimation, can be used to achieve a highly accurate estimation of curvature, minimizing the effects of noise for discretized surfaces. M-estimation (15; 46) consists of robustly fitting a model by minimizing a cost function of the residuals of the samples efficiently with an IRLS scheme. Here, we show an updated version of this method for point clouds and in the following subsections we introduce our method to (respectively) remove surface outliers from the input point set (subsection 4.1) and denoise point positions (subsection 4.2) using the results from the M-estimation process.

The first step of the algorithm is to determine a minimum neighborhood for each point p_i in the initial dataset (see Figure 4a). As in (19), this minimum neighborhood is determined by finding the closest points after projecting them into the local tangent plane of p_i, considering one closest point for each of six 60 °slices around p_i on this plane. If there are no nearest points in two or

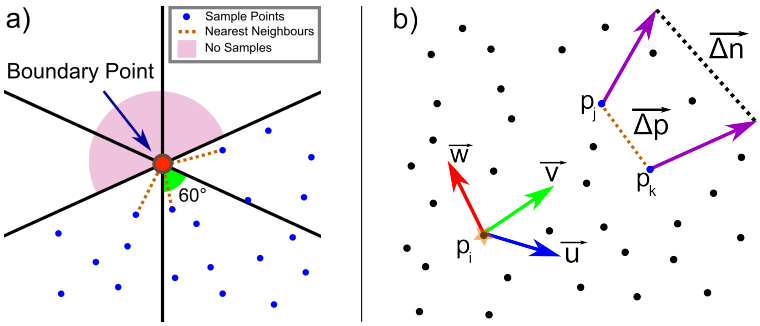


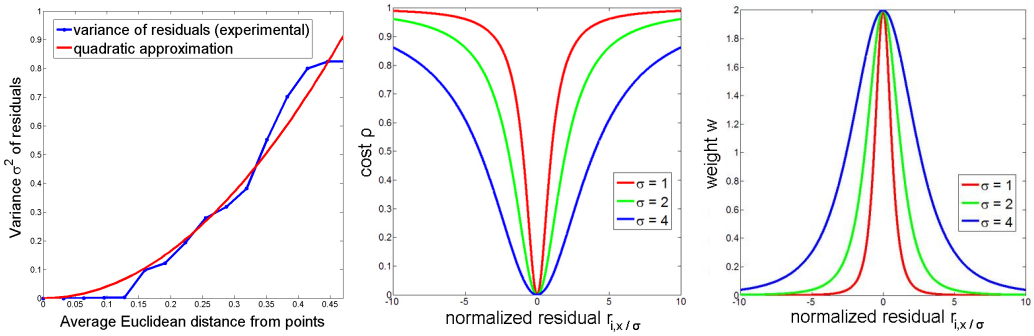
Fig. 4. **(a)** Boundary point definition and conditions. **(b)** Normal variation sample for curvature estimation.

more contiguous slices around p_i within a given threshold (which can easily be changed interactively in our application), the point is marked as a boundary point. If p_i is not a boundary point, we consider all pairs of points and their associated normals inside their minimum neighborhood. Each such pair yields a positional variation $\vec{\Delta}p$ and normal variation $\vec{\Delta}n$ which constrain the curvature tensor as follows:

$$\underbrace{\begin{bmatrix} \nabla_{\vec{u}}\vec{N}\cdot\vec{u} & \nabla_{\vec{v}}\vec{N}\cdot\vec{u} \\ \nabla_{\vec{u}}\vec{N}\cdot\vec{v} & \nabla_{\vec{v}}\vec{N}\cdot\vec{v} \\ \nabla_{\vec{u}}\vec{N}\cdot\vec{w} & \nabla_{\vec{v}}\vec{N}\cdot\vec{w} \end{bmatrix}}_{unknowns} \cdot \begin{bmatrix} \vec{\Delta}p\cdot\vec{u} \\ \vec{\Delta}p\cdot\vec{v} \end{bmatrix} = \begin{bmatrix} \vec{\Delta}n\cdot\vec{u} \\ \vec{\Delta}n\cdot\vec{v} \\ \vec{\Delta}n\cdot\vec{w} \end{bmatrix}$$

where $\vec{N}$ is the input normal vector field, and $\vec{u}$, $\vec{v}$ and $\vec{w}$ form a local orthonormal coordinate frame obtained from the tangent plane (see Figure 4b).

Given enough variation pairs, we obtain an over-constrained system which lets us solve for the curvature tensor values in a least squares fashion. This estimation serves as an initial guess to the IRLS process. Then, all normal variations inside an initial operating region (see below) are sampled and assigned a geometric weighting scheme according to the inverse of their average squared Euclidean distance to the center point p_i. This geometric weighting captures the prior confidence regarding the relative spatial relevance of the samples. This weighting will be multiplied with the M-estimation weights (see below) that represent the confidence in the noisiness of the sample. These confidence weights are necessary to ensure stable curvature tensor fittings, otherwise the samples would be assumed to be of equal quality and, therefore, to have constant variance, which is not true. Normally, it is expected that the further the sample is located from the point of interest, the less geometric weight it should be assigned. The statistics literature (3) suggests that for weighted least squares regression, the weights of the samples should be chosen according to the inverse of the variances of the residuals. As we do not know the variance of the residuals of the samples a priori, we need to model this variance with a function of some geometric feature of the sample. In our ex-



(a) (b) (c)

Fig. 5. **(a)** Variance of the residuals of the normal variation samples around each point versus their average Euclidean distance to the neighborhood center on a cylindrical surface (blue) and the approximating quadratic curve (red). **(b)** The Geman-McLure (GM) cost as a function of the normalized residual. **(c)** The GM weights as a function of the normalized residual.

periments, we noticed that the variances of the residuals of the samples tend to increase with the average Euclidean distance of the points and this increase can be adequately modeled with a quadratic function (see Figure 5a).

The initial operating region captures the scale where the curvature estimation will be performed robustly. In our experiments, we heuristically defined the scale to be locally density-dependent similarly to point cloud modeling techniques (38). The heuristic for the operating region we used is the region covered by the Euclidean ball centered at p_i with radius 3.0 multiplied by the average distance of p_i from its closest neighbors in its minimum neighborhood. Other techniques that optimize the radius of the curvature operator globally can also be used (49) but with increased computational cost. The M-estimation algorithm will reweight the samples accordingly (thus, reshape this initial region) in order to converge to an anisotropic support area during the IRLS process (22).

After the initial guess of the curvature tensor, the sampling and geometric weighting definition in the operating region, the IRLS process begins. According to the M-estimation literature, the goal is to minimize the sum cost of residuals:

$$\min_{x} \sum_{s_i \in S} \rho(r_{i,x}/\sigma) \tag{1}$$

where x is the model with the unknown parameters containing the curvature tensor, s_i is the ith variation sample, S is the sample set, $r_{i,x}$ is the absolute residual of sample s_i with respect to model x, σ is a scale factor that is automatically estimated (see below) and ρ is a robust cost function. In simple least squares the cost function is quadratic. However, such a cost function is very sensitive to outliers. For a robust estimation, we use the Geman-McLure cost function (GM), which is a proven choice in computer vision (11) and

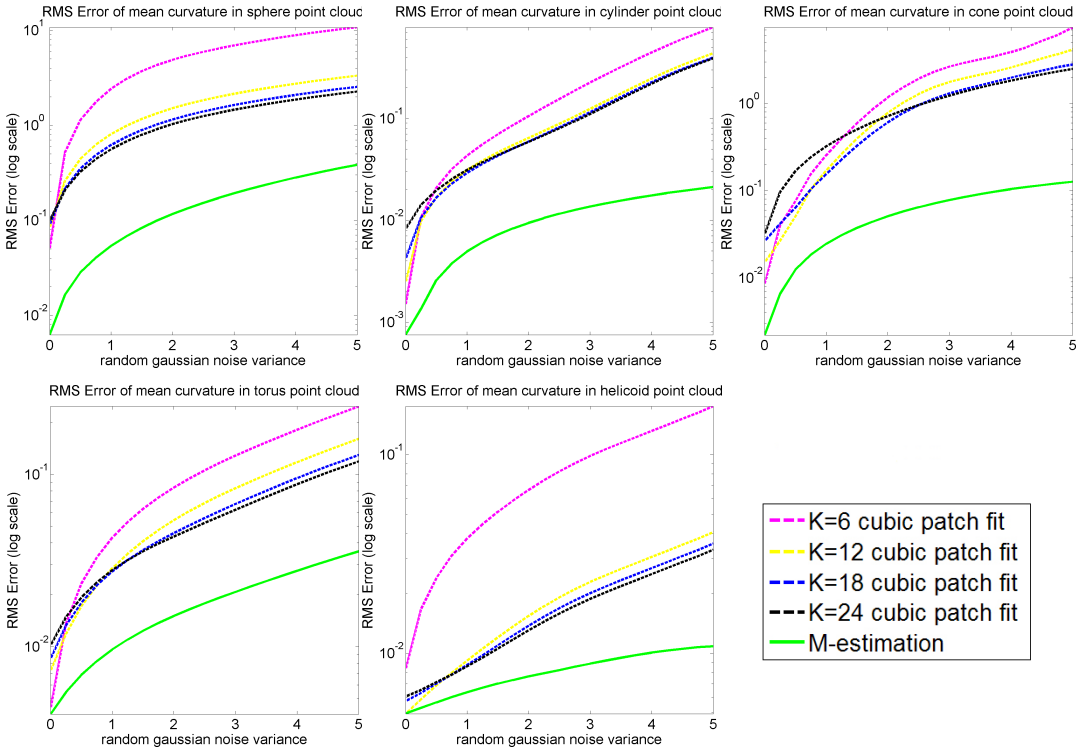


Fig. 6. Robust statistical estimation of curvature applied to analytic point clouds compared to cubic patch fitting on k-nearest neighbors versus increasing noise variance (percentage of median distance of points inside their defined minimum neighborhood as also reported in (22)). Normals are estimated by locally fitting planes (17). Here, we show mean curvature error.

geometry processing (44) as it exhibits very good behavior in quickly rejecting structured outliers.

The IRLS approach assigns statistical weights to the normal variation samples for each iteration given their observed residual $r_{i,x}$ from the currently fitted linear model x. These statistical weights represent the confidence of how good or bad the sample is according to its current residual (see Figure 5). The cost and weight for each sample are defined as:

$$\rho(r_{i,x}/\sigma) = \frac{(r_{i,x}/\sigma)^2}{1+(r_{i,x}/\sigma)^2} \qquad w(r_{i,x}/\sigma) = \frac{2}{(1+(r_{i,x}/\sigma)^2)^2}$$

where $\sigma = 1.4826 \cdot median(r_{i,x})$. This is derived from the fact that the median absolute value of a large enough sample of unit variance normally-distributed 1D values is $1/1.4826 = 0.6745$. This scale estimation causes the estimator to tolerate almost 50% of outliers (43).

At each iteration, the operating region is refined by considering the normal variation samples whose residuals are less than 2σ. The samples which have larger residuals are considered outliers for the curvature estimation of p_i and

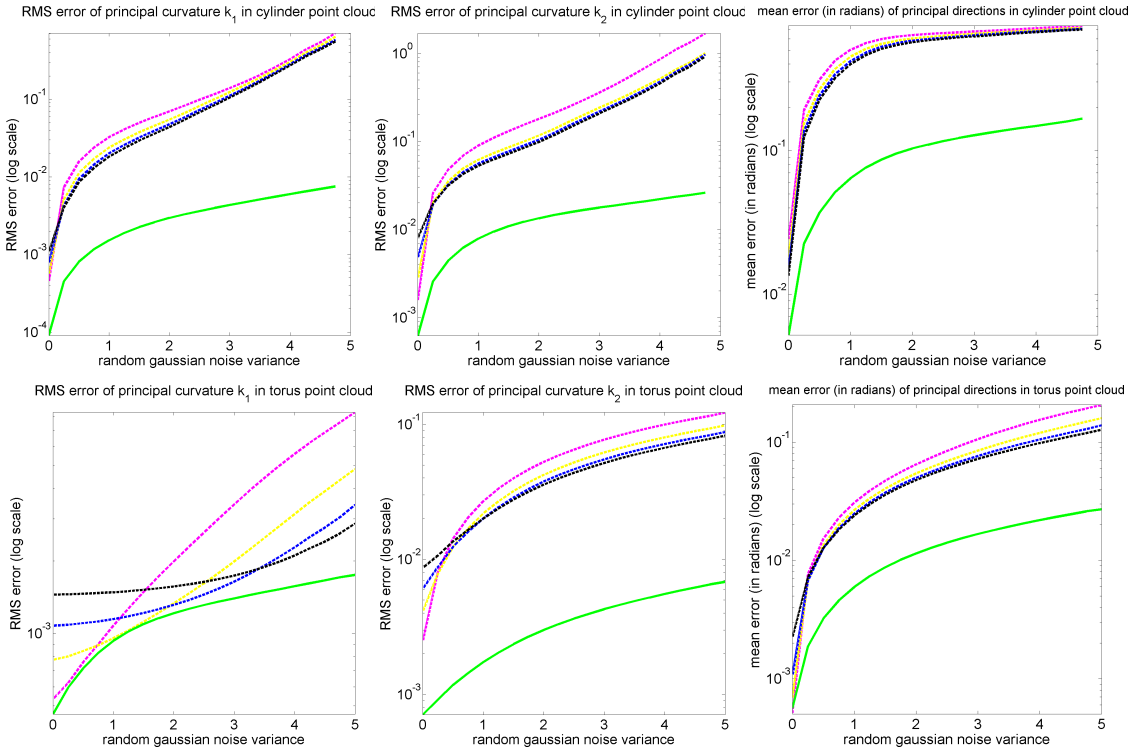


Fig. 7. Robust statistical estimation of curvature applied to analytic point clouds compared to cubic patch fitting on k-nearest neighbors versus increasing noise variance. Here, we show RMS error for principal curvature values and mean error (in radians) for principal curvature directions in cylinder and torus point clouds.

are ignored, as in (43). These statistical weights are chosen so that a cost function of the residuals of the samples is minimized and this corresponds to the maximum likelihood estimates of the curvature tensor (11).

Regarding convergence to true curvature values with increased sampling, maximum likelihood estimates (such as those obtained by M-estimation) converge with increased sample density (maximum likelihood bias tends to zero as the number of samples tends to infinity (24)).

The initial normals can also be corrected using the computed curvature tensors and the final M-estimation weights per each normal variation sample. Firstly, the normal differences between p_i and every point in its final operating region are computed using the values for the unknowns as estimated from the IRLS process. Then, the new normal at p_i is computed as the normalized weighted sum of the normals of its neighbor points in the operating region plus the derived normal differences. These weights are the final weights of the IRLS process assigned to each sample.

We show results for this statistical estimation of curvature for point clouds in Figures 1, 2, 3 and we provide quantitative numerical comparisons in Figures 6 and 7.

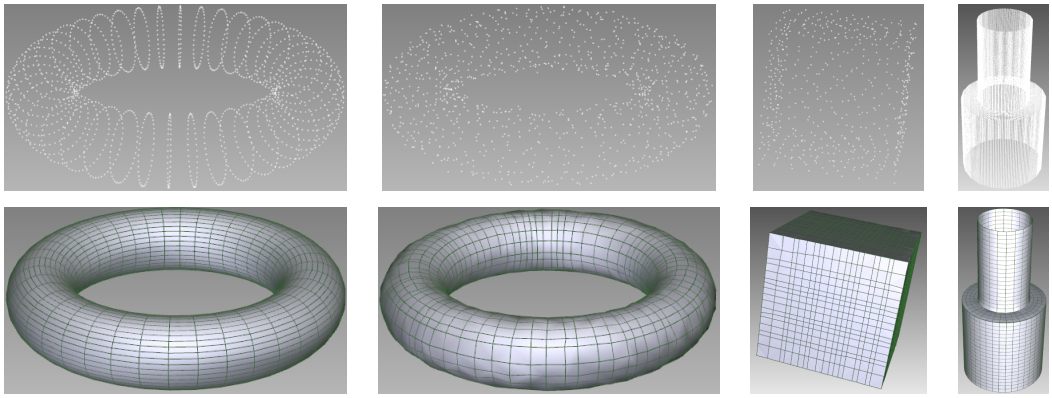


Fig. 8. Results of applying our approach for extracting lines of curvature from sampled analytic surfaces. **From left to right:** regularly sampled non-noisy torus, randomly sampled and noisy torus (noise is gaussian with variance 1% of the bounding-box diagonal), randomly sampled cube, a surface with mixed umbilical and non-umbilical regions.

4 Point cloud correction

The robust statistical estimation of curvature subsequently drives the outlier rejection, point position correction and principal direction smoothing presented in the following subsections.

4.1 Surface outlier rejection

Let us denote with $w^i_{j,k}$ the final weight associated to the variation pair (p_j, p_k) for the estimation of curvature at point p_i (see Figure 4b). For each point p_i we can define a sparse weight vector $\mathbf{w}_i$ as follows:

$$\mathbf{w}_i[p_j] = \sum_k w^i_{j,k}$$

which intuitively represents how much p_j contributes to determining curvature at p_i. If the weight is close to 1, then its associated normal variation is strongly related to the curvature of the point (p_i considers it an inlier). If it is 0, its associated normal variation is unrelated (p_i considers it an outlier).

Consider two points in the dataset, p_1 with weight vector $\mathbf{w}_1$ and p_2 with weight vector $\mathbf{w}_2$. In the case where $\mathbf{w}_1[p_2] > 0$ and $\mathbf{w}_2[p_1] = 0$, the point p_2 considers p_1 as an outlier in its curvature estimation, while the same does not hold for p_1 (this is a vote from p_2 for p_1 for being an outlier). On the other hand, if $\mathbf{w}_1[p_2] = 0$ and $\mathbf{w}_2[p_1] = 0$, the points are mutually irrelevant to each other's curvature estimation (no vote). If $\mathbf{w}_1[p_2] > 0$ and $\mathbf{w}_2[p_1] > 0$, both points contribute to each other's curvature.

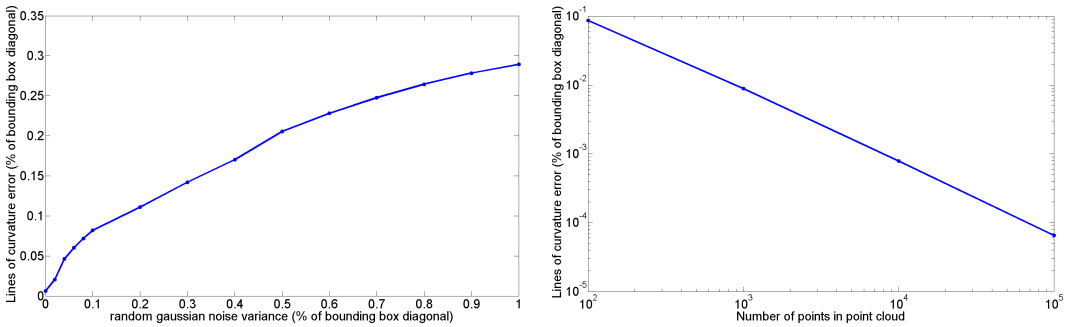


Fig. 9. *Left:* Average RMS position error of the sample points of the extracted lines of curvature compared to ground truth for the case of torus point cloud with increasing random noise variance (measured as percentage of the bounding box diagonal). Note that the resulting error is much lower than the noise of the input point cloud. *Right:* The approximation error of the extracted lines of curvature linearly decreases with respect to point cloud resolution.

If over half of a point's votes are in favor of considering it an outlier, we mark it as such and ignore it during the next steps of our method. Isolated boundary points are also ignored. We show this in the case of the helicoid (Figure 1) and fish (Figure 14) datasets. After the outliers are rejected, the minimum neighborhood for each point is reselected.

4.2 Point cloud denoising

The recomputed normals from the M-estimation process can be used to correct the position of the rest of the surface points. This is based on a global energy minimization process where the goal is to move the position of the points in such a way so that the local first-order approximation of the normal in the minimum neighborhood of each point p_i matches its robustly corrected normal $\vec{n}_i$. The cost function is defined as follows:

$$E = \sum_{i=1}^{N} \sum_{j=1}^{K-1} \sum_{k=j+1}^{K} \sqrt{||\vec{n}_i - s \cdot \hat{n}(p_i, q_j^i, q_k^i)||}$$

where q_j^i and q_k^i denote the j-th and k-th nearest neighbors of p_i respectively, $\hat{n}(p,q,r) = unit((q-p) \times (r-p))$, and $s = 1$ if $\vec{n_i} \cdot \hat{n}(p,q,r) \geq 0$ and $s = -1$ otherwise. Intuitively, $s \cdot \hat{n}(p,q,r)$ represents the oriented normal of the plane defined by p and its neighbors q and r. N is the number of points in the dataset, and $K = 6$ is the number of nearest neighbors we always consider. We take the square root of the norm of the difference to the local corrected normals as we noticed this better preserves features, similarly to the square-root potentials used in (7).

Such an optimization requires an analytic gradient in order to be performed

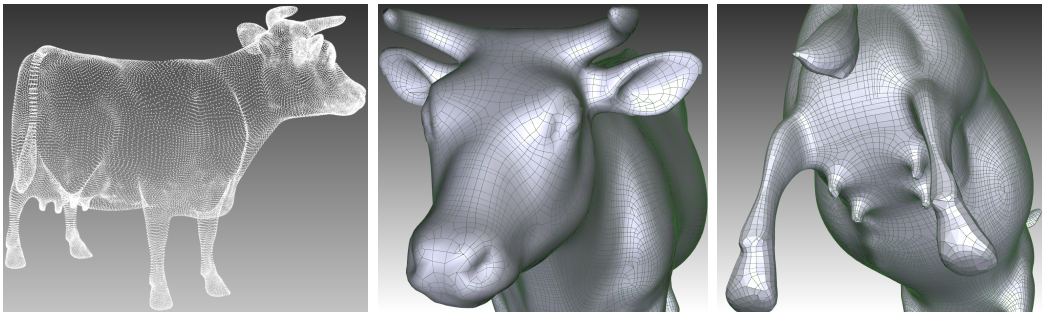


Fig. 10. Lines of curvature extracted for the synthetic cow dataset.

efficiently, which we provide. We employ the L-BFGS-B optimization method (52). L-BFGS-B is a limited-memory quasi-Newton method, which is intended for large-scale optimization problems in which the Hessian matrix is difficult to obtain, as in our case. In Figure 1 we show an example for a noisy helicoid. Figures 8, 12 and 14 also illustrate this technique. The reduction of noise can reach 70% or more in the noisy cases of torus (see Figure 9).

Notice that we do not explicitly place a penalty in our energy term for point movement and they are only locally moved based on their corrected normals. The minimization process does not result in global translation of points, since, in such a case, this would not result in lower global energy. Given our choice of the optimization algorithm, the fact that we use the original point positions as the initial guess, and the use of an analytic gradient, it is ensured that we find a minimum for locally optimal point placement which is close to the original point positions.

4.3 Global Principal Direction Smoothing

Optionally, we can also globally smooth the principal direction vectors. This is especially desired to reduce the number of singularities (which is useful for meshing applications). In order to achieve this, we use a global energy minimization, similar to the one defined in (16) and in (40). The energy is defined as:

$$E(\theta_i) = (1-\alpha)\cdot\sum_i \frac{|k_{i1}|}{|k_{i2}|}\cdot\sin^2(\theta_i-\theta_{0i}) - \alpha\cdot\sum_{i,j}\cos^2((\theta_i-\phi_{ij})-(\theta_j-\phi_{ji}))$$

where θ_i are the unknown angles between the target smoothed principal direction of k_1 and a reference tangential direction and ϕ_{ij} is the angle of the projection of the vector $v_i - v_j$ to the tangent plane of vertex v_i. The angles θ_{0i} are the initial angles and the user parameter $\alpha \in [0...1]$ controls the smoothing intensity. The points v_j are all points neighboring v_i that have non-zero statistical weights for the curvature estimation of v_i. Note that for our meshing purposes, the energy should be only invariant to differences of $n\pi$ on θ_i and

not $n\pi/2$ as it happens for cross-hatching (16). For this energy minimization, we also provide the analytic gradient and use the L-BFGS-B algorithm again (52).

5 Tracing lines of curvature and their intersections

After rejecting surface outliers and denoising the point dataset, our method starts to create the flow lines of principal curvature. As there is no prior smooth surface representation in our case, there is a need for special treatment of this process in order to make sure that the flow lines are extracted properly and no intersections are lost.

5.1 Tracing flow lines

A flow line is a piecewise linear curve created by sequentially sampling the underlying surface in adaptive step sizes following the robust curvature estimates. Let us call such a curve $\mathcal{C} = \{c_0, c_1, ..., c_n\}$, where c_i is the i-th sample point.

The sampling algorithm starts by building a priority queue of seed points selected from the corrected dataset, as well as creating a Voronoi structure over said set which will be used for efficiently implementing intersections. The points with highest priority are those that exhibit the highest confidence during the M-estimation process, i.e. those p_j with highest $\sum_i \mathbf{w}_i[p_j]$, as they contribute the most to the estimation of curvature of the other points in the dataset.

The first point in a given curve, c_0, is initialized by popping a point from the queue. From this point, we will start to trace flow lines in each of the principle curvature directions $\vec{d} \in \{\vec{k_1}, \vec{k_2}, -\vec{k_1}, -\vec{k_2}\}$. Note that during the tracing, we conform the principal directions locally to have the same sign.

Each new sampling point is generated firstly as $c_i \leftarrow c_{i-1} + s \cdot \vec{d_i}$, thus moving towards the current signed principal direction $\vec{d_i}$. As we will see, the step size s will be adaptively chosen and is initialized to half the distance of c_0 to its nearest neighboring point. Of course, c_i currently lies on the tangent plane of c_{i-1} and possibly not on the underlying surface. Therefore, we proceed with a process that corrects this by performing the following steps:

(1) Retrieve all points neighboring c_i that have non-zero statistical weights for the curvature estimation of c_i (as stored upon completion of the IRLS process). Let us call these points $q_1, q_2, \ldots, q_k$.

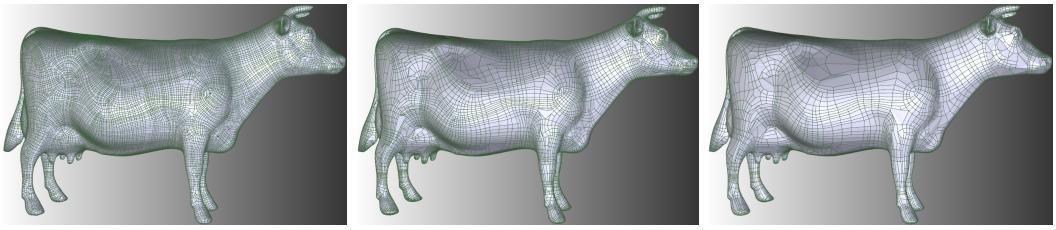


Fig. 11. *From left to right:* Lines of curvature with decreasing density parameter for the cow dataset (*Left:* $\kappa = 1e-5$, *Middle:* $\kappa = 1e-4$, *Right:* $\kappa = 1e-3$).

(2) Project c_i using LOP (Locally Optimal Projection) (28) using as support region the neighborhood defined by the previous step. As c_i is already close to the underlying surface, very few LOP iterations are performed (typically 4-5).
(3) Define the normal of c_i to be the normalized weighted average of the normals of q_j using their statistical weights. We also define its principal curvatures in the same way. We set $\vec{d}_i'$ according to this interpolated principal curvature direction and we update the point c_i. Then we again locally project the sampling point to get its new estimated position c_i'. We found this scheme provided the best stability compared to other integration methods. In order to adapt the step size according to the error of the integration, we also perform a second-order estimate c_i'' (by considering the average of the principal directions $\vec{d}_i'$ and $\vec{d}_i'$ this time) and compute the relative error $\Delta = |c_i' - c_i''|/|c_i'|$. The step size is then limited to $s \cdot \sqrt{\tau/\Delta}$ (if this quantity is smaller than the current step size), where τ is the prescribed user tolerance as commonly used in numerical integration techniques (in all our experiments, we used $\tau = 0.01$). Such choice of adaptive step sizes for backward Euler schemes follows the numerical integration literature e.g., see (48; 23).
(4) Check if the current flow line is crossing into a new datapoint Voronoi cell by intersecting one of the separating hyperplanes of the current cell. If so, further limit the step size s and update c_i (see below for reasoning).
(5) The statistical weights for c_i are set to the weights of the closest point in the dataset. Therefore, find this closest point and update it if necessary.

We continue this process, by repeating steps 1 through 5 for the updated flow point c_i. As noted in step 4, we also keep track of the flow points that belong to each of the Voronoi cells of the dataset and register them accordingly. This will be very important for the tracking of flow line intersections. For each Voronoi site in the dataset, we only need to check for intersections of the corresponding registered flow segments. This is efficient and guarantees no intersections will be lost.

The integration scheme we use allows for the efficient tracing of flow lines with satisfactory stability. It is also reminiscent of the flow tracing method in (9),

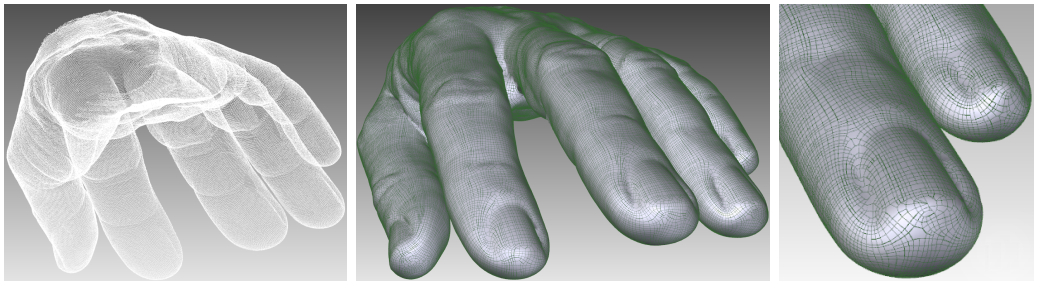


Fig. 12. Lines of curvature extracted for the scanned hand point cloud.

where instead of "walking" across a triangle, we walk across the neighborhood of each point by interpolating the tangential directions. This tracing is relatively fast, very stable and handles surfaces of arbitrary genus avoiding any assumptions for local parametrization.

Preserving features: Notice that our choice of neighborhood and the use of the statistical weights serve to preserve features during the tracing of the flow lines. For example, in the case of a cube (see Figure 8), for points near the cube edges, the weights of the points past the corresponding feature boundaries are zero. Thus, the flow line interpolates correctly along the cube faces and preserves hard edges (see also the preservation of sharp features of the cow in Figure 10).

Stopping conditions: Each current flow line stops if one of the following conditions is met:

(1) if the current flow point has a distance less than $d(\kappa)$ to a point of a different flow line (of the same principal curvature), where $d(\kappa) = 2\sqrt{\varepsilon(2/|\kappa| - \varepsilon)}$. This density threshold is adapted to the corresponding curvature κ of the flow point as in (1). For the examples of this paper, we have used $\kappa = 1e - 5$. In Figure 11, we show results with varying density of lines using different values of κ.
(2) If the current flow point is closer than 2 multiplied by the current step size to the starting point of the line, then there is a self-intersection.
(3) If a flow line reaches an umbilical point (the current flow point has distance less than the step size to an umbilic).

The proximity queries are performed by running a breadth-first search (BFS) based on the six nearest points of each point in the dataset. The BFS stops when there are no more points in the dataset within a distance equal to the density threshold and the current step size. For each retrieved point, we access its Voronoi cell structure and retrieve its registered flow points. Then, we check the above conditions based on the distance of the current flow point to the retrieved flow points of the nearby cells, as given by the BFS. Of course, this is done for efficiency reasons, as a KD-tree query for each flow point would be prohibitively slow, as also noticed in (31).

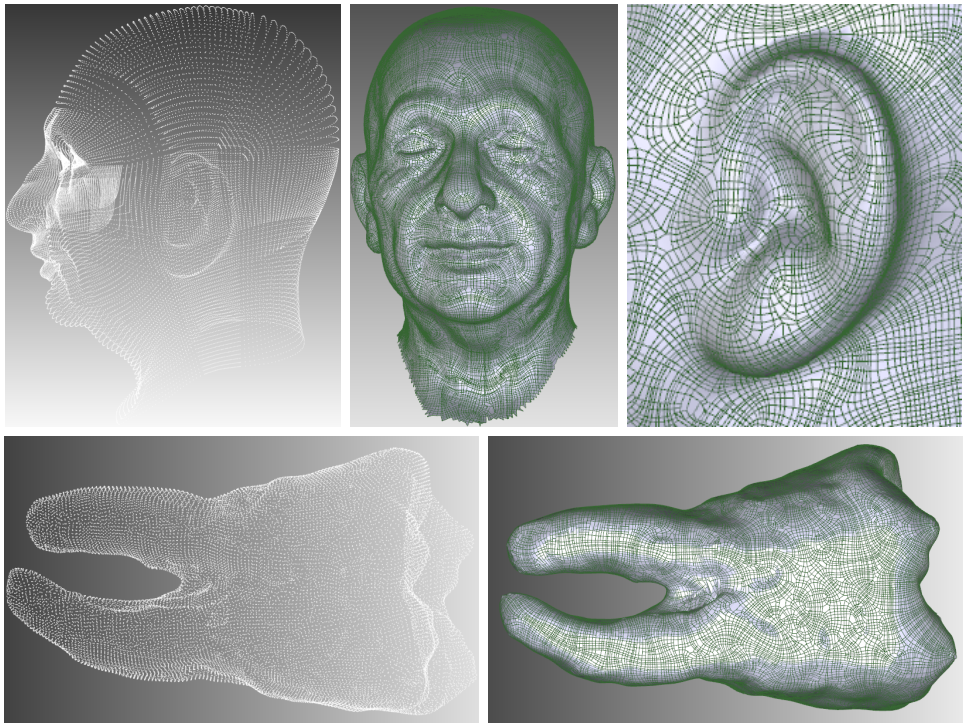


Fig. 13. Results of applying our method to head and tooth datasets.

In umbilical regions, the principal directions are not well defined. An umbilical region is defined as a contiguous set of umbilical points in the dataset (like in the case of the region of points lying on the face of a cube). A point is defined to be umbilic if its principal curvatures are equal. Because of numerical errors, we use a threshold to classify if a point is umbilic (as also used in other umbilic detection approaches like (34)). In our experiments, we found it reasonable to set a point to be umbilic if the ratio of its principal curvatures is larger than the threshold of 0.95. This value works well in our experiments, allowing for minute differences in values due to numerical errors. If an umbilical region contains more than three points, we perform Principal Component Analysis (PCA) on the region including the umbilical points and their non-umbilical neighbor points. The main reason for this is that the local symmetry axes (given by the eigenvectors of PCA) is an approximation of the averaged principal curvature directions of this region (51), which is expected to be more stable. We set the principal directions to be the projections of the eigenvectors which correspond to the highest eigenvalues on their tangent plane. If a flow line starting from an umbilical region reaches its boundary, then it stops (see example of cube in Figure 8). However, if a surface is comprised of patches of umbilical and non-umbilical regions, the lines of curvature can have discontinuities on their boundaries. In this case, the global optimization process of section 4.3 can be used to globally smooth the directions as in (16). A result of this process is shown in the example model on the right of Figure 8.

The result of executing the above process for each point in the priority queue

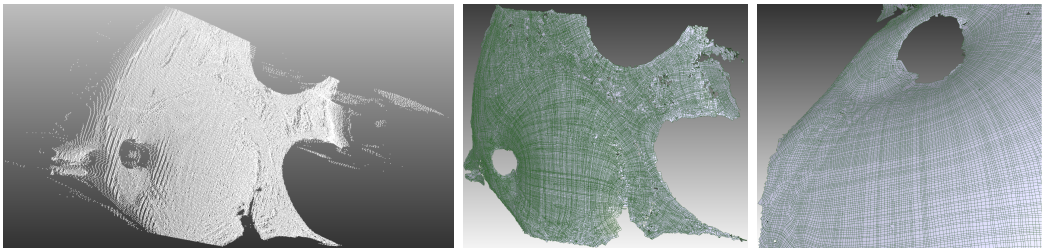


Fig. 14. Results of applying our method to highly noisy and outlier rich data acquired using scatter-trace photography. (36)

is a network of principal curvature lines (see Figure 8 of the torus) where each Voronoi cell data structure has the registered flow points. We can now track the intersections of the flow lines efficiently.

5.2 Checking for intersections

For each Voronoi site in the dataset, we search for intersections of the flow segments incident on said site. As the flow segments may not intersect exactly in 3D, we project them onto the tangent plane of the associated point. A sweep-line algorithm (6) is employed to quickly find the intersections. If there is an intersection between two flow segments on the tangent plane, we find their intersecting points and we reproject them. The new intersection (a new vertex) is set to be the midpoint of these reprojected points. We also check if the flow segments of other lines of curvature meet at an existing intersection. In this case, we update the vertex structure with all the meeting flow lines. Moreover, we set the normal of the point to be the average of the normals of the flow points of the intersecting segments in order to remain consistent with the original surface orientation.

Dataset	# of points	Curvature estimation	Normal correction	Outlier rejection	Point cloud denoising	Extraction of lines	Meshing	Total time	Maximum memory
torus	1600	0.7	0.2	0.1	2.4	2.9	0.9	7.2	14.5
cube	600	0.2	0.1	0.1	1.3	2.1	0.5	4.3	10
helicoid	1730	0.9	0.3	0.2	3.3	3.9	1.1	9.7	22.1
cow	46K	16.5	5.7	0.9	43.5	99.7	34.6	200.9	240.3
tooth	21K	10.5	2.9	0.4	30.2	65.2	20.4	129.6	135.7
Einstein	29K	13.9	4.1	0.6	26.8	79.1	28.7	153.2	169.2
fish	113K	87.4	30.5	4.9	170.4	378.9	115.0	787.1	714.9
hand	195K	94.9	31.4	3.5	176.3	466.3	159.9	932.3	989.5

Table 1
Indicative running times (in seconds) for each stage of our method captured on a 3GHz Intel Pentium IV processor with 2GB memory. Total execution time (in seconds) and maximum memory used (in megabytes) are also reported.

6 Application: curvature-aligned mesh construction

After the generation of lines of curvature and the checking for intersection, the application to curvature-aligned mesh construction follows from our method in a straightforward manner. Our approach can provide dense mesh reconstructions, is statistically robust to noise and preserves fine features.

After tracking the new vertices of the intersecting segments, it is easy to proceed with the construction of the half-edge structure. In the vertex structure, we keep indices to the flow points of the intersecting flow lines. Each flow point has pointers to its previous and next flow points in the line. We traverse the flow lines to find the neighboring intersections of each vertex. In this way, we create all the edges between the vertices.

Similarly to (31), for every vertex, we project all its edges onto its tangent plane, as given by its computed normal. Having one of the projected edges as a reference, we find the angles of all the other edges to it and we sort them according to this angle in a counter-clockwise direction. This results in the correct cyclic half-edge order. Based on this process, we build all the half-edges for each vertex.

We select a half-edge from the list of all retrieved half-edges and traverse the next half-edge until the starting vertex is met. We mark these half-edges as visited and we create a face. Then we continue this process, until all half-edges are visited. Optionally, the faces can be triangulated. Otherwise the meshing process naturally produces a quad-dominant mesh.

7 Results

We show results of our curvature estimation algorithm in Figures 1, 2, 3, 6, and 7. We show the results of our approach for the extraction of lines of curvature on analytic examples with varying noise and sampling quality (see Figures 1, 8 and 9), models with sharp features, large umbilic regions, as well

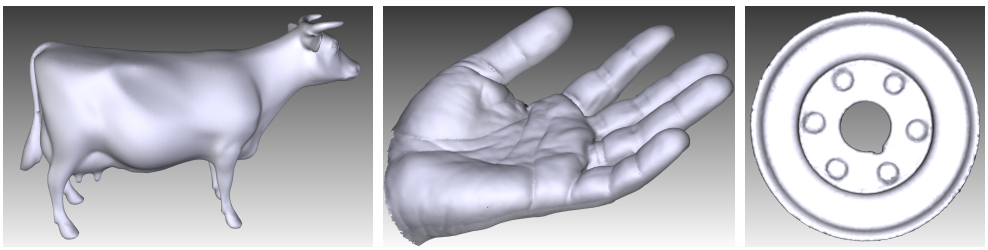


Fig. 15. The lines of curvature traced on the point clouds by our method can also be used for direct mesh reconstruction.

as synthetic and commercially scanned real-world examples (Figures 1, 8, 10, 12 and 13) and even highly noisy, reflective objects (Figure 14) with many outliers, acquired using scatter-trace photography (36). We show examples of surface reconstruction in Figure 15.

Our implementation uses CGAL for its data structures and the QHULL algorithm for the Voronoi cell computation. Indicative running times for each stage of our algorithm for our datasets are shown in table 1. We note that there our implementation is not optimized, since we were more interested in ease of prototyping than in speed.

8 Discussion, Limitations and Future Work

We presented a method that allows the generation of lines of curvature directly on noisy point clouds with outliers. We believe that there are many other point cloud based techniques that can benefit from our method: point cloud shape recognition, registration, feature extraction, symmetry detection to name a few. The entire technique is well grounded on a robust statistical estimate of curvature and normals used in the denoising of the point cloud, excluding outliers and smoothly extracting the lines of curvature in a feature-preserving manner.

We acknowledge that the computational cost and memory requirements of the current implementation of our method are relatively high. Moreover, as the meshing explicitly follows the lines of curvature, the resulting meshes are not very regular. We note that our algorithm mainly focuses on producing a faithful mesh reconstruction with all the fine features of the underlying surface well represented, rather than building a smooth global parametrization (such as in (40)), based also on the fact that curvature-aligned meshes optimally approximate a smooth surface at least in non-hyperbolic regions (45; 5).

There are many extensions to our work that we are currently exploring, which could further enhance this novel type of lines of curvature extraction. A statistical technique to automatically improve the sampling density over an arbitrary genus surface, in the lines of the method presented in (7), could improve the reconstruction quality. A robust statistical detection of boundaries and crest lines from the point clouds would also be very important for better surface reconstruction. An interesting extension of our work could be to generate isotropic flow lines on the point cloud given a global parametrization. Another improvement would be to automatically set some user parameters (e.g, the density threshold) through an example-based learning technique. A data-driven method like (21) could be employed to track lines of curvature at interactive rates. Our technique could potentially be used for automatic hole

filling and repairing of incomplete meshes. Finally, it would be interesting to explore if our framework could be used for parametrization of noisy point clouds.

9 Acknowledgements

We would like to thank Aaron Hertzmann and Eugene Fiume for their useful comments on the paper. We thank Nigel Morris and Kyros Kutulakos for the fish point cloud. We thank Hanson Robotics for the Einstein's head dataset. We acknowledge the AIM@SHAPE 3D library and its authors for the use of 3D models and point clouds. This work has been funded by the National Sciences and Engineering Research Council of Canada (NSERC), the Ontario Ministry of Education and Training and the Canadian Research Network for Mathematics of Information Technology and Complex Systems (MITACS).

References

[1] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, M. Desbrun, Anisotropic polygonal remeshing, ACM Transactions on Graphics 22 (3) (2003) 485–493.

[2] P. Alliez, D. Cohen-Steiner, Y. Tong, M. Desbrun, Voronoi-based variational reconstruction of unoriented point sets, in: Eurographics Symposium on Geometry Processing, 2007, pp. 39–48.

[3] R. Carroll, D. Ruppert, Transformation and Weighting in Regression, Chapman and Hall, 1988.

[4] D. Cohen-Steiner, J. M. Morvan, Restricted delaunay triangulations and normal cycle, in: Symposium on Computational geometry, 2003, pp. 312–321.

[5] E. F. D'Azevedo, Are bilinear quadrilaterals better than linear triangles?, SIAM Journal on Scientific Computing 22 (1) (2000) 198–217.

[6] M. de Berg, M. van Krevelda, M. Overmars, O. Schwarzkopf, Computational Geometry: Algorithms and Applications, 2nd ed., Springer-Verlag, 2000.

[7] J. R. Diebel, S. Thrun, M. Brünig, A bayesian method for probable surface reconstruction and decimation, ACM Transactions on Graphics 25 (1) (2006) 39–59.

[8] S. Dong, P. T. Bremer, M. Garland, V. Pascucci, J. C. Hart, Spectral surface quadrangulation, ACM Transactions on Graphics 25 (3) (2006) 1057–1066.

[9] S. Dong, S. Kircher, M. Garland, Harmonic functions for quadrilateral

remeshing of arbitrary manifolds, Computer Aided Geometric Design 22 (5) (2005) 392–423.
[10] D. Dutta, R. R. Martin, M. J. Pratt, Cyclides in surface and solid modeling, IEEE Computer Graphics and Applications 13 (1) (1993) 53–59.
[11] D. A. Forsyth, J. Ponce, Computer Vision: A Modern Approach, Prentice Hall, 2002.
[12] T. Gatzke, C. Grimm, Estimating curvature on triangular meshes, International Journal of Shape Modeling 12 (1) (2006) 1–29.
[13] J. Goldfeather, V. Interrante, A novel cubic-order algorithm for approximating principal direction vectors, ACM Transactions on Graphics 23 (1) (2004) 45–63.
[14] B. Hamann, Curvature approximation for triangulated surfaces, Geometric modelling (1993) 139–153.
[15] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, W. A. Stahel, Robust Statistics: The Approach Based on Influence Functions, Wiley-Interscience, 1986.
[16] A. Hertzmann, D. Zorin, Illustrating smooth surfaces, in: ACM SIGGRAPH, 2000, pp. 517–526.
[17] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Surface reconstruction from unorganized points, in: ACM SIGGRAPH, 1992, pp. 71–78.
[18] A. Jagannathan, E. L. Miller, Unstructured point cloud matching within graph-theoretic and thermodynamic frameworks, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2, 2005, pp. 1008–1015.
[19] P. Jenke, M. Wand, M. Bokeloh, A. Schilling, W. Strasser, Bayesian point cloud reconstruction, in: Eurographics, 2006, pp. 379–388.
[20] F. Kalberer, M. Nieser, K. Polthier, Quadcover - surface parameterization using branched coverings, Computer Graphics Forum 26 (3) (2007) 375–384.
[21] E. Kalogerakis, D. Nowrouzezahrai, P. Simari, J. McCrae, A. Hertzmann, K. Singh, Data-driven curvature for real-time line drawing of dynamic scenes, ACM Transactions on Graphics 28 (1).
[22] E. Kalogerakis, P. Simari, D. Nowrouzezahrai, K. Singh, Robust statistical estimation of curvature on discretized surfaces, in: Eurographics Symposium on Geometry Processing, 2007, pp. 13–22.
[23] D. Kavetski, P. Binning, S. W. Sloan, Adaptive backward euler time stepping with truncation error control for numerical modelling of unsaturated fluid flows, International Journal for Numerical Methods in Engineering 53 (6) (2001) 1301–1322.
[24] S. M. Kay, Fundamentals of Statistical Signal Processing, Prentice Hall PTR, 1993.
[25] Y. K. Lai, L. Kobbelt, S. M. Hu, An incremental approach to feature aligned quad dominant remeshing, in: ACM Symposium on Solid and Physical Modeling, 2008, pp. 137–145.

[26] Y.-K. Lai, Q.-Y. Zhou, S.-M. Hu, J. Wallner, H. Pottmann, Robust feature classification and editing, IEEE Transactions on Visualization and Computer Graphics 13 (1) (2007) 34–45.
[27] C. Lange, K. Polthier, Anisotropic smoothing of point sets, Computer Aided Geometric Design 22 (7) (2005) 680–692.
[28] Y. Lipman, D. Cohen-Or, D. Levin, H. Tal-Ezer, Parameterization-free projection for geometry reconstruction, ACM Transactions on Graphics 26 (3) (2007) 22.
[29] Y. Liu, H. Pottmann, J. Wallner, Y. L. Yang, W. Wang, Geometric modeling with conical meshes and developable surfaces, in: ACM SIGGRAPH, 2006, pp. 681–689.
[30] T. Maekawa, F. Wolter, N. M. Patrikalakis, Umbilics and lines of curvature for shape interrogation, Computer Aided Geometric Design 13 (2) (1996) 133–161.
[31] M. Marinov, L. Kobbelt, Direct anisotropic quad-dominant remeshing, in: Pacific Graphics, 2004, pp. 207–216.
[32] R. R. Martin, Principal patches–a new class of surface patch based on differential geometry, in: Eurographics '83, 1983, pp. 47–55.
[33] M. Meyer, M. Desbrun, P. Schröder, A. H. Barr, Discrete differential-geometry operators for triangulated 2-manifolds, in: Visualization and Mathematics III, Springer-Verlag, 2002, pp. 35–57.
[34] N. J. Mitra, L. J. Guibas, M. Pauly, Partial and approximate symmetry detection for 3d geometry, ACM SIGGRAPH 25 (3) (2006) 560–568.
[35] N. J. Mitra, A. Nguyen, L. Guibas, Estimating surface normals in noisy point cloud data, Special issue of International Journal of Computational Geometry and Applications 14 (4–5) (2004) 261–276.
[36] N. J. W. Morris, K. N. Kutulakos, Reconstructing the surface of inhomogeneous transparent scenes by scatter trace photography, in: International Conference on Computer Vision, 2007, pp. 1–8.
[37] Y. Ohtake, A. Belyaev, H.-P. Seidel, Ridge-valley lines on meshes via implicit surface fitting, ACM Transactions on Graphics (2004) 609–612.
[38] M. Pauly, R. Keiser, L. P. Kobbelt, M. Gross, Shape modeling with point-sampled geometry, ACM Transactions on Graphics 22 (3) (2003) 641–650.
[39] M. J. Pratt, Cyclides in computer aided geometric design, Computer Aided Geometric Design 7 (1-4) (1990) 221–242.
[40] N. Ray, W. C. Li, B. Lévy, A. Sheffer, P. Alliez, Periodic global parameterization, ACM Transactions on Graphics 25 (4) (2006) 1460–1485.
[41] S. Rusinkiewicz, Estimating curvatures and their derivatives on triangle meshes, 3DPVT (2004) 486–493.
[42] S. Rusinkiewicz, M. Levoy, Efficient variants of the ICP algorithm, in: Proceedings of the Third Inernational Conference on 3D Digital Imaging and Modeling, 2001, pp. 145–152.
[43] H. S. Sawhney, S. Ayer, M. Gorkani, Model-based 2d&3d dominant motion estimation for mosaicing and video representation, in: International Conference on Computer Vision, 1995, pp. 583–590.

[44] P. Simari, E. Kalogerakis, K. Singh, Folding meshes: hierarchical mesh segmentation based on planar symmetry, in: Eurographics Symposium on Geometry Processing, 2006, pp. 111–119.
[45] R. B. Simpson, Anisotropic mesh transformations and optimal error control, Applied Numerical Mathematics: Transactions of IMACS 14 (1–3) (1994) 183–198.
[46] C. V. Stewart, Robust parameter estimation in computer vision, SIAM Rev. 41 (3) (1999) 513–537.
[47] G. Taubin, Estimating the tensor of curvature of a surface from a polyhedral approximation, in: ICCV, 1995, pp. 902–909.
[48] R. M. Thomas, I. Gladwell, Variable-order variable-step algorithms for second-order systems, International Journal for Numerical Methods in Engineering 26 (1) (1988) 39–53.
[49] W. S. Tong, C. K. Tang, Robust estimation of adaptive tensors of curvature by tensor voting, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (3) (2005) 434–449.
[50] Y. Tong, P. Alliez, D. Cohen-Steiner, M. Desbrun, Designing quadrangulations with discrete harmonic forms, in: Eurographics Symposium on Geometry Processing, 2006, pp. 201–210.
[51] Y.-L. Yang, Y.-K. Lai, S.-M. Hu, H. Pottmann, Robust principal curvatures on multiple scales, in: SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing, 2006, pp. 223–226.
[52] C. Zhu, R. H. Byrd, P. Lu, J. Nocedal, Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization, ACM Transactions on Mathematical Software 23 (4) (1997) 550–560.