Generating Parametric Models of Tubes from Laser Scans

Ulrich Bauer *,[1],
Universität Göttingen, Lotzestraße 16-18, 37073 Göttingen, Germany

Konrad Polthier [2]
Freie Universität Berlin, Arnimallee 6, 14195 Berlin, Germany

Abstract

We present an automatic method for computing an accurate parametric model of a piecewise defined pipe surface, consisting of cylinder and torus segments, from an unorganized point set. Our main contributions are reconstruction of the spine curve of a pipe surface from surface samples, and approximation of the spine curve by G^1 continuous circular arcs and line segments. Our algorithm accurately outputs the parametric data required for bending machines to create the reconstructed tube.

Key words: Pipe Surfaces, Tubes, Surface Reconstruction, Curve Approximation, Moving Least Squares.

1. Introduction

The concept of *parametric modeling* has gained much attention in the CAD community in recent years. Instead of describing a CAD model by a standard surface description such as polygonal meshes or NURBS surfaces, a model is described using higher-level entities that can be modified using few intuitive parameters, reflecting the design process instead of only the result. In the case of bent tubes, a parametric model could have the lengths and angles of the segments as parameters. These features are not explicitly deducible from a polygonal or a NURBS representation of the surface.

Another relevant problem that attracted a lot of interest in recent years is reconstruction of surfaces from point clouds. Numerous algorithms have been proposed for either interpolating or approximating an unorganized, possibly noisy set of points on a surface. A common approach is to find a triangulation of the point set as a subset of its Delaunay complex [1]. Another class of methods describes the surface as the zero level set of a signed distance function, an example being [2]. Common to all these approaches is that they aim at reconstructing a general smooth surface.

Combining these two ideas, we present a method for parametric reconstruction of a special class of shapes, which we call *bent tube surfaces*. These surfaces are pipe surfaces consisting of alternating cylinder and torus segments joined G^1 continuously (see Fig. 1 for an example). A *pipe surface* is defined as the envelope of a sphere with constant radius r moving along a curve (called *spine curve*); in our case, the spine curve is a piecewise defined G^1 continuous curve consisting of circular arc and line segments, called *arc-line spline*.

Parametric models of bent tube surfaces are of special interest since industrial tubes produced by bending machines have a shape of this type. Reconstructing the parametric description of the individual segments allows to derive the data required by bending machines to reproduce the scanned tube. Often, the circular arcs of the curves have one common radius due to the way most tube bending machines work.

The parametric model is generated from 3D laser scan data, which can be acquired from a single view: utilizing the symmetry of the cross section of a pipe surface, the scan does not have to cover the whole surface. The goal is to extract a small number of meaningful parameters from a very large, unorganized, and noisy data set. One important aspect is that the number of segments is correctly identified; a model containing more segments than necessary to faithfully describe the point cloud would surely not be considered a good reconstruction. One the other hand, the input samples should be metrically close to the surface model, a few possible outliers excluded. To combine these two requirements, we roughly take the following approach: we fix the maximal tolerance and then try to find the

* Corresponding author.
URLs: http://ddg.math.uni-goettingen.de/ (Ulrich Bauer), http://geom.mi.fu-berlin.de/ (Konrad Polthier).
[1] Work supported by TubeExpert, Ltd.
[2] Supported by DFG Research Center MATHEON "Mathematics for key technologies" in Berlin.

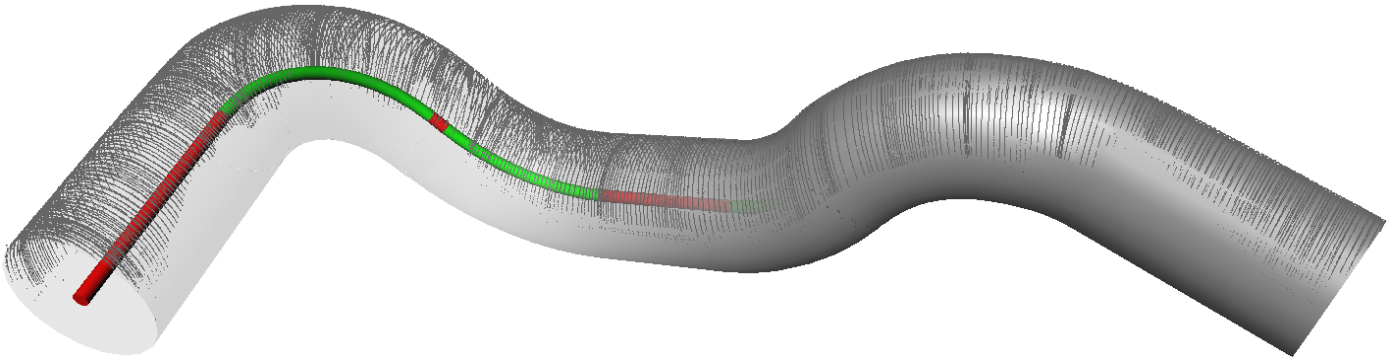


Figure 1. Parametric model of a bent tube – a pipe surface, consisting of cylinder (red spine) and torus (green spine) segments – automatically reconstructed using our algorithm from a laser scan (samples in grey).

model with the least number of segments inside this tolerance.

An important application for our algorithm is *inspection*, i.e., comparison of CAD data of a tube with results after bending, and possible correction of the data to compensate for errors induced in the bending process. These errors may stem from spring-back of the tube. Another application is reverse engineering of tubes for which no bending data is available.

1.1. *Related work*

Simple methods for generating parametric models of bent tubes from range data and for inspection based on given reference models have been proposed in [3,4]. These methods use local estimates of principal curvatures to segment the point cloud into cylindrical and toroidal parts. While this approach works for a broad range of well-behaved shapes, it has a few serious drawbacks. First, the estimation of differential quantities is prone to noise and requires some averaging or other robust methods to avoid over-segmentation. Moreover, using local estimates alone, it is sometimes very difficult to recognize very short cylindrical or toroidal segments that are clearly recognizable from the global shape; examples of this type can be seen in Figure 9. Another problem is that the segmentation of the point cloud alone does not yield cylinder and torus segments that fulfil the G^1 continuity constraint. Enforcing the constraints can lead to significant deviation from the input data. These problems motivated the research that lead to this article. Our approach avoids both these problems by searching only inside a set satisfying both the G^1 continuity constraint and a distance constraint.

Reconstruction of pipe surfaces from unorganized points has been investigated by Lee in [5] and later extended to canal surfaces in [6]. The methods there are based on translating surface samples along the estimated normal direction. This method is susceptible to deviations from the exact normals and therefore requires smoothing of the spine curve point cloud afterwards. See Fig. 2 for a comparison to our new method.

In the aforementioned article, Lee [5] also proposed the use of the *moving least squares* approximation technique for smoothing a noisy point cloud of a curve. This method has later gained much attention in the context of computing surfaces approximating a point cloud, and several variants are proposed in [7,2] and references therein. We will adapt this method to compute a point cloud approximating the spine curve of a pipe surface, given a point cloud of the surface itself.

Several methods have been proposed for computing an approximate medial axis [1] or a curve skeleton [8] derived from the medial axis. Assuming a dense, noise-free sampling of the tube, one of these methods could be used to obtain the spine curve of a pipe surface. There are however several serious drawbacks to these approaches. Most importantly, the mentioned methods are not robust to noise and outliers in the input data. Moreover, they are computationally quite expensive. Third, we seek for a method that works with partial scans that cover only one side of the tube. All of the mentioned algorithms would require input samples around the tube surface.

An algorithm for the reconstruction of planar curves from unorganized point sets, based on Voronoi diagrams and Delaunay triangulations, has been presented in the seminal paper by Amenta et al. [9]. Another simple but provably correct method, which also works for curves in higher dimensions, is the NN-Crust algorithm, proposed by Dey and Kumar [10]. This is the algorithm we used in our implementation for reconstructing a polygonal curve from a point cloud.

The problem of approximating a given curve by another curve with a simpler description has a long history. Commonly considered is the problem of *simplification* of polygonal chains, where the distance between input and output is bounded by ϵ in some distance measure. The heuristic by Douglas and Peucker [11] is the standard technique for this problem, although it has no optimality guarantee and a worst-case running time of $O(n^2)$. For the problem of simplification of a space curve, with the vertices of the output polygon being a subset of the input vertices, Eu and Toussaint [12] present an algorithm which finds an optimal solution within a running time of $O(n^3)$. Agarwal et al. [13] describe an efficient greedy approximation algorithm running in $O(n \log n)$ and requiring at most as many line segments as the optimal 2ϵ approximating polygonal chain. For the *weak simplification* problem, where the output vertices can be chosen freely, Guibas et al. [14] propose an exact algorithm for planar curves with a running time of $O(n^2)$; the weak simplification problem for space curves is still unsolved.

Segmentation and approximation of planar and space curves

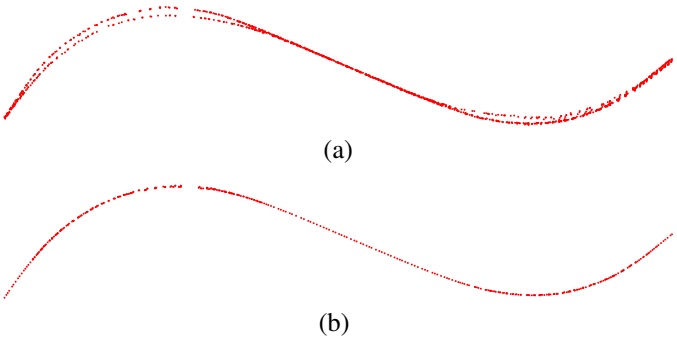


Figure 2. Spine point cloud obtained using (a) Lee's method (before smoothing) and (b) our method. While in (a) the noise is considerable, in (b) the result is sharp and does not require smoothing.

by G^0 and G^1 continuous circular arcs and line segments has been investigated by several authors, in most cases without any guarantee about the optimality of the solution. Hoschek [15] finds a G^1 planar approximation; Rossignac [16] approximates a space curve by G^0 continuous circular arcs. Drysdale et al. [17] compute a minimum segment count G^0 planar approximation with junction point chosen from the input vertices. Several authors [18,19,20] describe the approximation of planar curves using G^1 continuous biarcs (pairs of circular arcs joined G^1 continuously). Biarcs, first introduced in [21], are easier to handle than regular arcs, because both endpoint tangents of a biarc can be chosen independently. The problem of approximating a polygonal chain with a G^1 smooth arc spline using a minimum number of segments is still open. Moreover, all of the mentioned algorithms are either not suited for certain problem instances such as arc segments with an apex angle greater than π, or only produce arc splines consisting of biarcs.

1.2. *Contributions*

We propose a complete method for parametric reconstruction of bent tube surfaces. Our main contributions are:

Spine curve reconstruction
We present a novel technique to reconstruct the spine curve of a pipe surface. We use a moving least squares based technique to find locally best fitting cylinders approximating the input samples, and define a projection of the surface samples onto the estimated spine curve. Compared to a previous approach, our method produces high-quality, thin and smooth point clouds of the spine curve without additional post-processing. Moreover, the method only requires a partial scan of the pipe surface, e.g. from a single view scan.

Curve approximation by arc-line splines
We propose an algorithm to compute an approximation of a polygonal curve by an arc-line spline within a given distance bound distance. The method uses estimated tangent lines of the input curve and computes an approximation with minimum segment count for these estimates. In contrast to many other curve approximation algorithms using circular arc segments, it can also cope with arc segments having an apex angle equal or greater than π.

1.3. *Algorithm overview*

Our algorithm for parametric reconstruction of bent tube surfaces performs the following steps, depicted in Fig. 3.
– First, the samples of the surface, shown in Fig. 3 (b), are projected onto the spine curve. This procedure is described in Section 3. The result of this step is shown in Fig. 3 (c).
– Next, we reconstruct a polygonal curve from the spine point cloud using the NN-Crust algorithm of Dey and Kumar [10], as shown in Fig. 3 (d).
– The polygonal curve is optionally simplified using Eu and Toussaint's algorithm [12] to reduce the problem complexity for further computations.
– This polygonal curve is then approximated using an arc-line spline as described in Section 4, see Fig. 3 (e).
– Finally, the parametric description of the bent tube surface (see Section 5) is optimize with regard to the least squares distance of the surface to the input samples using the Levenberg-Marquardt nonlinear optimization method (Section 6). The final output of our reconstruction is shown in Fig. 3 (f).

2. Review of estimated differential quantities on point clouds

In this section, we will briefly recall the methods used in our algorithm for estimation of curvature of a surface given a noisy point cloud of the surface.

2.1. *Local polynomial surface approximation*

To obtain robust local information about the surface described by a point cloud, we use an approximation by a (weighted) best-fitting polynomial over an estimated tangent plane of the surface. This technique is commonly used for estimating normals and curvatures; Pouget [22] gives a detailed analysis of this popular technique, and Lukácz [23] compares it to other methods for estimating curvature in point clouds. Local fitting of polynomials is also used for defining a distance function to the various definitions of *moving least squares* surfaces, see [7,2].

To obtain an estimate of the tangent plane, let $\theta : \mathbb{R}^+ \to \mathbb{R}^+$ be a monotonically decreasing distance weighting function with support $[0, \epsilon]$. We consider the local neighborhood $N_p = \{p_1, \ldots, p_n\}$ of a point p consisting of all sample points inside a ball of radius ϵ centered at p. A principal component analysis of this set of points provides a first order estimate of the tangential plane at p and the reference domain for the polynomial fitting [2] in the following way. We use the weights $\theta_k = \theta(\|p_k - p\|)$, and assume that the weighted barycenter

$$\bar{p} = \frac{\sum_k \theta_k \, p_k}{\sum_k \theta_k}$$

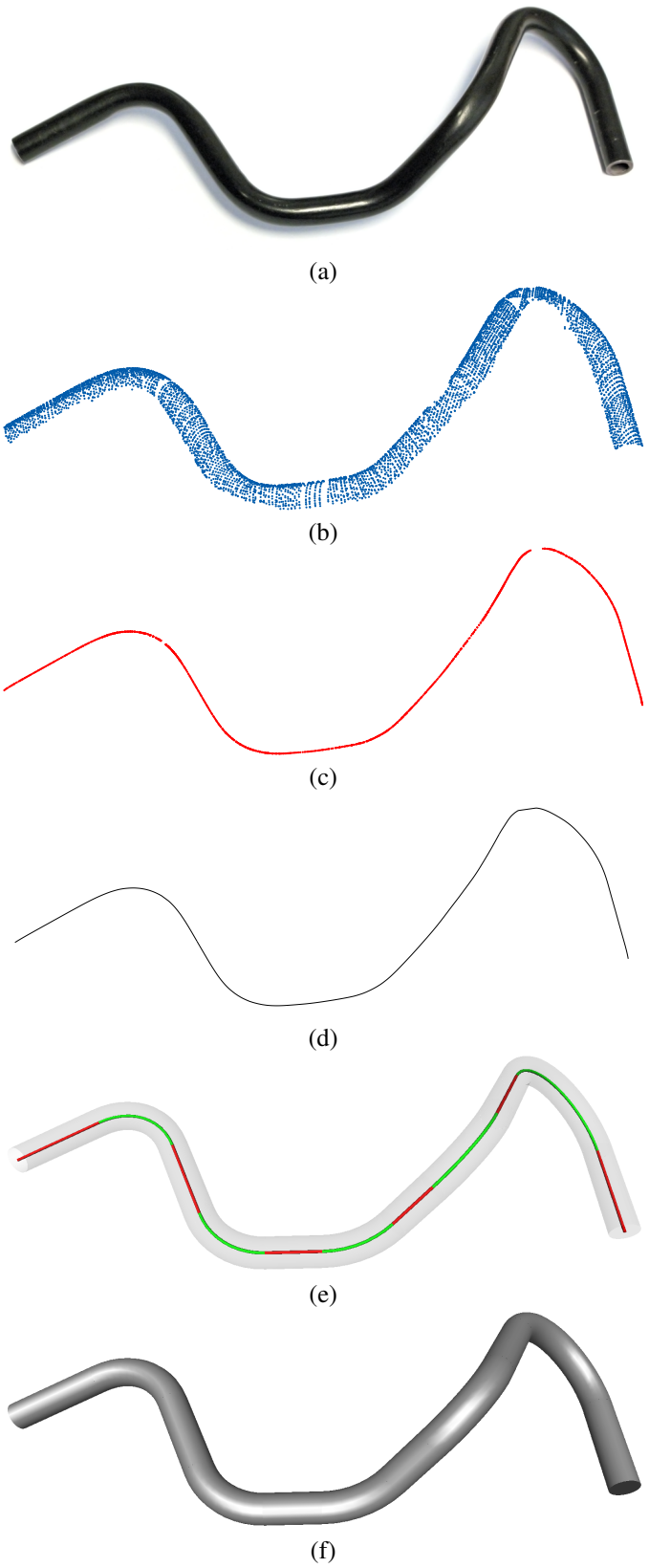


Figure 3. Reconstruction of a tube. (a) A photograph of the original tube. (b) A single-view, irregular scan of the tube surface. (c) Projection of the point cloud onto the MLS spine approximation. (d) Reconstruction of the spine curve. (e) Approximation of the spine curve by a G^1 continuous arc-line spline. (f) The reconstructed tube surface.

lies at the origin of the coordinate system. Then the estimated normal ν of this tangent plane to p is the solution of the optimization problem

$$\min_{\|\nu\|=1} \sum_k \langle \nu, p_k \rangle^2 \, \theta_k$$

which is the eigenvector with the smallest corresponding eigenvalue of the weighted covariance matrix $W(p) = (w_{ij})$ with

$$w_{ij} = \sum_k \langle e_i, p_k \rangle \langle e_j, p_k \rangle \, \theta_k \ .$$

This estimated tangent plane is only used as the reference domain to compute a bivariate polynomial approximation to the neighborhood N_p with the estimated tangent plane as reference domain. Let Π_2^2 denote the bivariate polynomials of degree 2. The paraboloid $\pi(x, y) = ax^2 + by^2 + cxy + dx + ey + f$ minimizing

$$\min_{\pi \in \Pi_2^2} \sum_k (\pi(x_k, y_k) - z_k)^2 \, \theta_k$$

can be found by solving the linear equation system

$$A^T \begin{pmatrix} \theta_1 & & 0 \\ & \ddots & \\ 0 & & \theta_n \end{pmatrix} A \begin{pmatrix} a \\ b \\ \vdots \\ f \end{pmatrix} = A^T \begin{pmatrix} \theta_1 z_1 \\ \theta_2 z_2 \\ \vdots \\ \theta_k z_n \end{pmatrix}$$

with

$$A = \begin{pmatrix} x_1^2 & y_1^2 & x_1 y_1 & x_1 & y_1 & 1 \\ & & \vdots & & & \\ x_n^2 & y_n^2 & x_n y_n & x_n & y_n & 1 \end{pmatrix} .$$

In the presence of noise, the normal to the graph of the polynomial at the origin $(x, y) = (0, 0)$ also provides a substantially better approximation of the surface normal than our previously computed first order estimate of the tangent plane.

2.2. *Shape operator estimation*

Consider the paraboloid embedded in $\mathbb{R}^3$ parameterized by

$$\pi(x, y) = (x, y, (ax^2 + by^2 + cxy + dx + ey + f))$$

at the point $(x, y) = (0, 0)$. The matrix representations of the first and second fundamental forms at this point for the canonical basis $\frac{\partial}{\partial x}, \frac{\partial}{\partial y}$ are

$$G = \begin{pmatrix} d^2 + 1 & de \\ de & e^2 + 1 \end{pmatrix}$$

and

$$H = \frac{1}{\sqrt{1 + d^2 + e^2}} \begin{pmatrix} 2a & c \\ c & 2b \end{pmatrix} .$$

The shape operator S has the matrix representation $G^{-1}H$. The eigenvectors of this matrix represent the principal curvature directions in the basis $\frac{\partial}{\partial x}, \frac{\partial}{\partial y}$ (note that this is generally

not an orthogonal basis). The eigenvalues are the corresponding principal curvatures $\kappa_{\min}$ and $\kappa_{\max}$. These values therefore provide the estimates for the principal curvatures of the surface at the point p.

3. Spine curve reconstruction from samples of a pipe surface

We present a method to reconstruct the spine curve of a pipe surface from an unorganized, noisy set of points on the pipe surface. We use a moving least squares technique to project each surface point to its closest point on the spine curve. This spine curve point cloud can then be smoothed and be used for computing a polygonal curve using any curve reconstruction algorithm.

Our MLS projection is based on the projection procedure described in [7]. At a glance, this method consists of finding a weighted local polynomial approximation to the points in the neighborhood of a point p, similar to the method described in Section 2.1, and projecting the point p onto this polynomial.

We extend this concept by fitting other geometric primitives than bivariate polynomials to the local neighborhood. We reconstruct the spine curve of a pipe surface by weighted least-squares distance fitting of a cylinder [24] to points in the neighborhood of a point p_c close to the spine curve, and projecting this point onto the axis of the best-fitting cylinder.

We first define a distance function of a point p to its locally best-fitting cylinder. For each point $p \in \mathbb{R}^3$, we can try to find the cylinder $C \in Cyl_r$ minimizing the weighted least-squares distance to the samples, using a locally supported distance weighting function θ:

$$\min_{C \in Cyl_r} \frac{\sum_i d(C, p_i)^2 \theta_i}{\sum_i \theta_i}$$

where $d(C, p_i)$ is the (signed or unsigned) distance function of a point p_i to the cylinder C, a_C is the axis of the cylinder C, and $\theta_i = \theta(\|\tilde{p} - p_i\|)$, where $\tilde{p}$ is the projection of p onto the axis. Note that $\tilde{p}$ can be considered as the center of a radial weighting function. The name *moving least squares* hints at the fact that this center is not fixed for a given p, but also depends on the cylinder C to be optimized.

Let $a_{\min}$ be the axis of the optimal cylinder $C_{\min}$. The spine MLS distance function is now defined as

$$d_S(p) = d(a_{\min}, p) \ .$$

Our MLS spine is defined as the zero level set of this distance function. We can now use a simple projection procedure to generate a point cloud of the spine curve from a point cloud of the surface.

(i) Find an initial estimate for a point x and a tangent direction t of the spine curve.
(ii) Find the best-fitting cylinder $C_{\min}$ with radial weights centered at $P(x, a_{\min})$, using (x, t) as initial estimates.
(iii) Let x' be the projection of x onto the axis $a_{\min}$ of $C_{\min}$.

We want to note that, just as for MLS surfaces, it is important to avoid running into a local but not global minimum of the error function for cylinder fitting. Proper initial values for x and t are therefore crucial for obtaining good results. How to find these is described in the following subsections.

3.1. *Estimating the pipe radius*

Cylinder fitting can be done using either a fixed or a variable cylinder radius. Depending on the quality of the input sample, the one or the other option may be more desirable.

Under the general assumption that the bend radius r_B of the pipe is always more than twice as large as the pipe radius r_P, the larger principal curvature κ_{max} corresponds to the pipe radius of the surface: $r_P = \frac{1}{\kappa_{max}}$. The principal curvatures are obtained using polynomial fitting as described in Section 2.2. This assumption also implies that smoothness of the surface is the same as of the spine curve.

For cylinder fitting with fixed radius, we estimate the global pipe radius as the median of the larger principal curvature values $\kappa_{\max}$ for a sufficient number of samples of the point cloud.

3.2. *Choosing a neighborhood size*

We now discuss appropriate choices for the neighborhood size, i.e. the support of the weighting function θ. Similar to MLS curves, a large support for θ has a smoothing and shrinking effect on curved parts of the spine curve, while small neighborhoods can cause instability in the cylinder fitting step. We found that a neighborhood of about $1.2\ r_B$ (where r_B is the bend radius) provides a good trade-off, ensuring a stable projection while keeping shrinkage at a low level. For a cylindrical point set, this neighborhood is a cylinder with a height of about $1.3\ r_P$ (where r_P is the pipe radius). Under the mentioned assumption $r_B > 2r_P$, numerical evaluation using random sampling of a torus shows that the deviation of the spine curve approximation found by cylinder fitting from the real spine curve is still less than $0.038\ r_P$ in the worst case.

3.3. *Tangent direction estimation*

An estimation for the tangent direction of the spine curve is required as an initial guess for cylinder fitting. A good initial guess is crucial for obtaining the correct solution, as a bad initial guess can cause the optimization algorithm to run into a local (but not the global) minimum of the cylinder distance function.

From the shape operator estimate at a surface point p, we can derive an estimate for the tangent direction of the spine curve at $\tilde{p}$ (the projection of p onto the spine curve). It is easy to observe that the tangent direction of the spine curve correlates to the principal curvature directions onthe pipe surface itself. Since we know by assumption that the curvature of the spine curve is less than $\frac{1}{2r_P}$, the maximal directional curvature $\kappa_{\max}$ is always less than $\frac{1}{r_P}$, and the corresponding principal curvature direction is perpendicular to the spine curve. Consequently, the principal curvature direction corresponding to $\kappa_{\min}$ is parallel to the tangential vector of the spine curve.

3.4. *A starting point for cylinder fitting*

The above estimates of the surface normal ν (Section 2.1) and the pipe radius r_P (Section 3.1) provide an initial point p_0 on the cylinder axis for cylinder fitting. We translate the surface sample p by the pipe radius in normal direction: $p_0 = p - r_P \cdot \nu$. For exact values of the normal and pipe radius, p_0 is a point on the spine curve. Together with the tangent direction, this provides a complete initial guess for cylinder fitting.

4. Curve approximation by arc-line splines

In this section, we propose an algorithm for the problem of approximating a polygonal curve in $\mathbb{R}^3$ with a G^1 continuous curve consisting of circular arcs and line segments. Our goal is to minimize the number of arc segments of the approximating curve, while guaranteeing a distance of at most ϵ from each of the vertices of the input polygon to the approximating curve.

4.1. *Algorithmic framework*

The algorithm is based on a general framework used in several polygon simplification algorithms [12,25]. The main idea is to construct a shortcut graph $G = (V, E)$ over the vertices of the input curve, with an edge $e_{ij} \in E$ if and only if the chain $[p_i, p_{i+1}, \ldots, p_j, p_{j+1}]$ can be approximated by a circular arc and two line segments, as shown in Fig. 4. To ensure tangential continuity of the segments, we fix estimated tangent lines on which the line segments should lie. For the vertex i in the graph, we simply choose the line $l_i = p_i p_{i+1}$ as the estimated tangent line.

An approximating arc-line with minimum number of segments can then simply be found using breadth first search on this graph. The graph does not have to be constructed explicitly, therefore our algorithm only requires $O(n)$ space, see [25].

For the computation of an edge e_{ij} of the shortcut graph, the estimated tangent lines t_i and t_j are generally not coplanar. Therefore, we project the tangent line of the second segment t_j onto the plane P spanned by p_i, p_{i+1}, and p_{j+1} and use this modified tangent line t'_j to decide whether there is an edge e_{ij} in the shortcut graph (see Subsection 4.2).

Note that we get a different modified tangent line t'_j for each edge e_{ij}. To ensure G^1 continuity of the solution, the modified tangent line t'_j according to the shortest path in the graph from vertex 1 to vertex j is used for subsequent computations of outgoing edges of j, since this path is a candidate for a prefix of the optimal solution.

It is important to note that optimality can not be guaranteed for our algorithm, because we only consider a subset of all possible solutions, defined by the estimated tangent lines. This restriction is comparable to the situation for polygon simplification, where algorithms using a shortcut graph only find simplified curves with vertices from the input curve. An approximating polygon with fewer vertices may exist if the vertices are not restricted to this set.

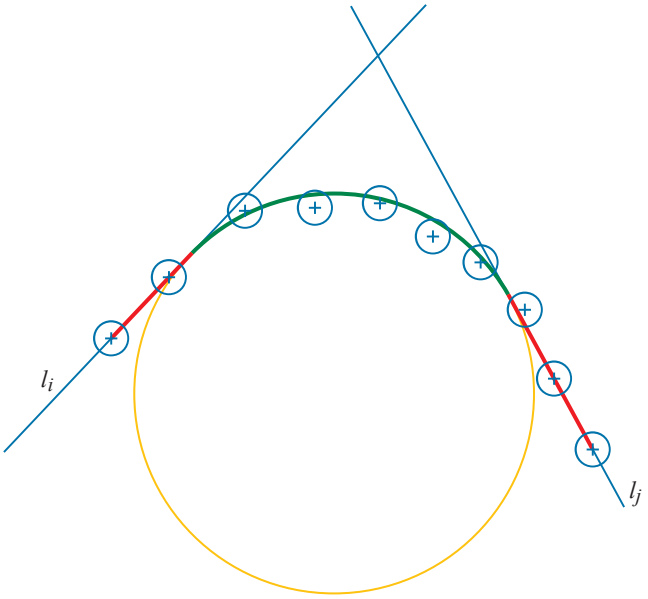


Figure 4. A circular arc and two line segments approximating a set of points.

In the case of polygon simplification, the set of allowed solutions is restricted to a canonical subset of all possible solutions. Because of the G^1 continuity constraints in our problem, such a canonical subset does not exist: the tangent directions can, in general, not be chosen from the polygonal input curve.

4.2. *Building the graph*

We will now describe how to compute the edges e_{ij} of the shortcut graph. Given two lines on a common plane P, we have to decide if a polygonal chain $[p_i, p_{i+1}, \ldots, p_j, p_{j+1}]$ can be approximated by two line segments on these lines, and a circular arc touching both lines and smoothly joining the line segments (see Fig. 4). The error criterion is that all vertices of the chain must have distance less or equal than ϵ to one of the line segments or to the arc. This can be decided as follows:

For any vertex p_k, let p'_k be its projection onto the plane P. If $\left\|p_k - p'_k\right\| > \epsilon$, then the chain cannot be approximated. Else, let $\epsilon'_k = \sqrt{\epsilon^2 - \left\|p_k - p'_k\right\|^2}$. This is the maximum allowed distance for the vertex p_k to the curve in the plane, since $\epsilon'^2_k + \left\|p_k - p'_k\right\|^2 = \epsilon^2$.

Next, we calculate for each p_k the set of approximating arcs. We assume that the first line segment starts at p_i and the second line segment ends at p_{j+1}. The center of the arc joining the line segments must lie on the bisector of l_1 and l_2 separating p_i and p_{j+1}.

The locus of the centers of valid circles consists of zero, one, or two line segments. The endpoints of these line segments can be obtained by constructing the classical Apollonius line–line–circle (LLC) problem (Fig. 5, green and yellow). This problem asks for the circles touching two lines and one circle, in our case a circle with radius ϵ'_k around the vertex p'_k. It is easy to see that all circles touching the two lines and having distance less than ϵ'_k from p'_k have their centers between a pair of centers

Algorithm 1 Computing an approximating arc-line spline

Require: A polygonal chain $C = [p_1, p_2, \ldots, p_n]$ and a tolerance value ϵ

Ensure: A G^1 arc-line spline $\{l_1, a_1, l_2, a_2, \ldots, l_m\}$ approximating P with distance at most ϵ to each vertex $p_i \in P$

for all $1 \le i < (n-1)$ **do**
  for all $j > i$ **do**
    let P be the plane spanned by $\{p_i, p_{i+1}, p_{j+1}\}$
    for all $p_k \in \{p_i, p_{i+1}, \ldots, p_{j+1}\}$ **do**
      let p'_k be the vertex p_k projected onto P
      $\epsilon'_k = \sqrt{\epsilon^2 - \|p_k - p'_k\|^2}$
    end for
    $l_1 = p'_i p'_{i+1}$; $l_2 = p'_j p'_{j+1}$; $L = l_1 \cap l_2$
    for all $p'_k \in \{p'_{i+1}, p'_{i+2}, \ldots, p'_j\}$ **do**
      compute all circles tangential to:
        l_1, l_2, and the circle around p'_k with radius ϵ'_k
      if there are four solutions **then**
        let C_1, C_2 be the circle centers farthest from L
        $V_k = \overline{C_1 C_2}$
      else if there are two solutions **then**
        let C_1 be the circle center farthest from L
        $V_k = \overline{C_1 L}$
      else
        $V_k = \emptyset$
      end if
      $V = V \cap V_k$
    end for
    if $V \neq \emptyset$ and $d_j > d_i + 1$ **then**
      $d_j = d_i + 1$
      $prev(j) = i$
      choose any $C_j \in V$
      let B_j and E_j be projections of C_j onto l_1 and l_2
      $p_j = p'_j$
    end if
  end for
end for
$\{i_1, i_2, \ldots, i_m\} = \{1, \ldots, prev(prev(n-1)), prev(n-1), n-1\}$
for all $1 \le k < m$ **do**
  let l_k be the line segment $\overline{E_{i_{k-1}} B_{i_k}}$
    (with $E_{i_0} = p_1$ and $B_{i_m} = p_n$)
  let a_k be the arc starting at B_{i_k} and ending at E_{i_k}
    with center C_{i_k}
end for
return $\{l_1, a_1, l_2, a_2, \ldots, l_m\}$

of Apollonius circles. Note that we are only interested in one of the possibly two pairs of solutions: the two circle centers that are furthest away from the intersection of the two lines l_1 and l_2 (green in Fig. 5).

If the ϵ'_k-disk around p'_k intersects one of the two lines, say l_1, then there is only one pair of solutions to the LLC problem. In this case, only the lower solution is of interest for us: any arc with center beyond the upper solution is valid, since p'_k has distance less than ϵ to a line segment on l_1.

Intersecting the ranges of valid arc centers for all $p'_k \in \{p'_i, p'_{i+1}, \ldots, p'_j, p'_{j+1}\}$ gives the range of valid centers for the whole chain $[p_i, p_{i+1}, \ldots, p_j, p_{j+1}]$. If this intersection is non-empty, then we know there exists a curve of two line segments and a circular arc approximating the chain.

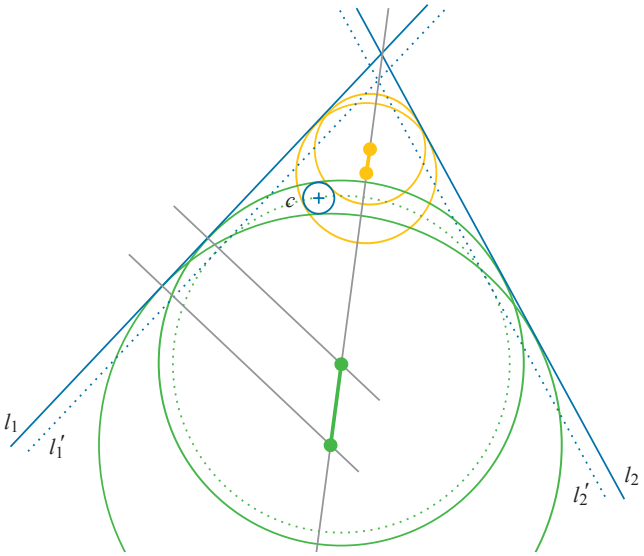


Figure 5. The Apollonius LLC problem and centers of approximating arcs. The solid circles (green and yellow) are tangent to the lines l_1 and l_2 and the circle c. The dotted lines and circle show the auxiliary LLP construction.

4.3. *Constructions*

The Apollonius circles for the line-line-circle (LLC) problem (Fig. 5) are constructed as follows. The solution circles (tangential to two lines l_1, l_2 and a circle c centered at P with radius r) have the same centers as the solution circles of a derived line–line–point (LLP) problem (the circles going through P and tangential to two lines l'_1, l'_2 at distance r from the original lines l_1, l_2, shifted towards P). Therefore, the LLC problem can be reduced to the LLP problem (Fig. 6). The other pair of solutions is obtained by shifting the lines in the other direction.

To solve the LLP problem for two lines l_1, l_2 and a point P, construct any circle c tangential to l_1 and l_2, with center M. The line OP joining O, the intersection of l_1 and l_2, and the point P, intersects c in two points I and J. The lines through P parallel to IM and JM intersect the bisector b of l_1 and l_2 in C_1 and C_2, the centers of the solution circles.

The centers of these two circles can be calculated by solving one single quadratic equation. Let L be the point where a solution circle with center C_i touches line l_1. Now we have

$$r = \|C_i - L\| = \|C_i - P\| \ .$$

Let s, t and u, v be points and tangent vectors of l_1 and the bisector b of l_1 and l_2, respectively. With $C_i = s + \lambda t$ and $L = u + \langle C_i - u, v\rangle\, v$, we get

$$\|\lambda(t - \langle t, v\rangle\, v) + (s - u - \langle s - u, v\rangle\, v)\| = \|\lambda t + s - P\| \ .$$

Setting $w := (t - \langle t, v\rangle\, v)$, $y := (s - u - \langle s - u, v\rangle\, v)$, and $z := (s - P)$, we obtain

$$\|\lambda w + y\| = \|\lambda t + z\|$$

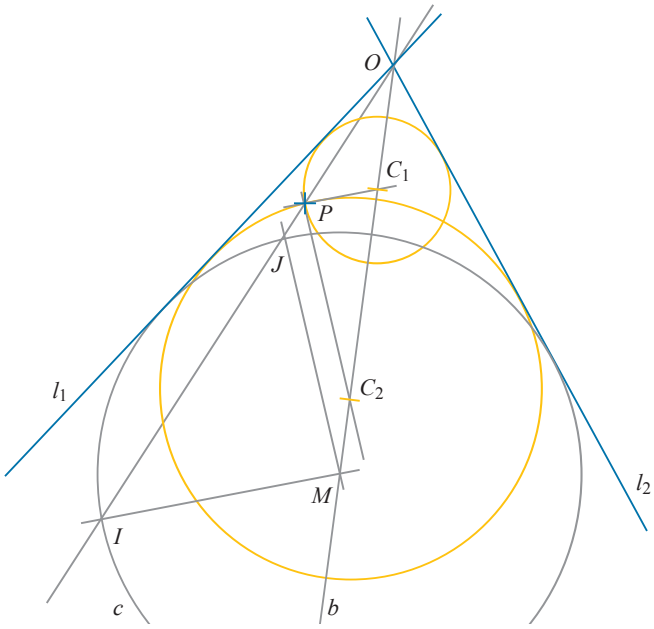


Figure 6. The Apollonius LLP problem. The yellow circles touch the two blue lines l_1 and l_2 and the point P.

which can be written as

$$(\|w\|^2 - \|t\|^2)\lambda^2 + 2(\langle w, y\rangle - \langle t, z\rangle)\lambda + (\|y\|^2 - \|z\|^2) = 0 \ .$$

Solving this equation gives us the two possible centers of the circles solving the LLP problem.

4.4. *Complexity*

The worst-case complexity of our algorithm on input polygons of size n is $O(n^3)$, because each of the three nested loops can have $O(n)$ iterations. However, on real-world data, the inner loop often does not have to be executed if the distance of one of the points $\{p_i, p_{i+1}, \ldots, p_{j+1}\}$ to the plane P is larger than ϵ. Therefore, we observed a better running time in practice, in the order of magnitude of about $O(n^{2.5})$.

4.5. *Preprocessing*

To speed up the computation and to stabilize the estimation of the tangent lines, we apply a polygon simplification algorithm on the polygonal spine curve as a preprocessing step. We used the algorithm from [25], which computes a minimum segment approximation using a subset of the input vertices with distance less or equal to ϵ in time $O(n^3)$, using space $O(n)$. We observed a real-world running time in the order of magnitude of about $O(n^2)$.

Additionally, we choose from all possible optimal curves the one with the smallest sum of squared errors. This can be done in a greedy way without increasing the asymptotic running time bound of the algorithm.

5. Representation of arc-line splines

We now investigate properties and representations of arc-line splines, the class of curves that our curve approximation algorithm should output.

5.1. *Local control of arc-line splines*

We show that no parametric representation of arc-line splines exists which has the *local control* property, i.e., the property that every parameter has only influence to a constant number of segments of the curve. For polygonal chains and other piecewise defined curves such as B-splines, the canonical representations fulfill this property. Local control would be very desirable as it would simplify construction and local modification of arc-line splines, especially in the context of least squares fitting (see Section 6).

Consider the configuration space of an arc-line spline with point and tangent constraints on both endpoints; we assume alternating arc and line segments, and line segments at the beginning and end of the curve. This space can be considered as an algebraic set in the cartesian product of the configuration spaces of the line and arc segments

$$S = L \times A \times \cdots \times L$$

subject to the condition that the end points and end tangents of two consecutive segments coincide. The configuration space L of line segments embedded in $\mathbb{R}^3$ has dimension 6; the space A of arc segments embedded in $\mathbb{R}^3$ is 8-dimensional. The continuity constraints can be expressed using 5 equations: 3 for G^0 continuity – since the junction point between two consecutive segments is an element of $\mathbb{R}^3$ – and 2 for G^1 continuity, since the unit tangent at this point can be identified with a point on the sphere $\mathbb{S}^2$. Therefore, the dimension of the configuration space of an arc-line spline is

$$d = 6 - 5 + 8 - 5 + \cdots + 6 = 4n + 6$$

where n is the number of arc segments of the curve.

Now consider the subspace of this configuration space S_c obtained by additionally fixing the endpoints and end tangents of the curve. This subspace is an algebraic set of dimension at least $d - 10 = 4n - 4$ for $n > 1$. To achieve local control of the curve, we have to find a parameterization of the curve that has the form

$$\mathbb{R}^d \to (\mathbb{R}^3 \times \mathbb{S}^2) \times S_c \times (\mathbb{R}^3 \times \mathbb{S}^2) \ .$$

10 parameters should describe the constraints at the endpoints, and the remaining $(d - 10)$ parameters should be mapped onto the space S_c of all curves with n segments fulfilling these constraints. In other words, we require that the two ends of the curve are locally controlled.

A necessary condition for the existence of such a parameterization scheme is that for any choice of constraints, the subspace S_c is a $(d-10)$-dimensional manifold. This means that all constraint equations must be independent. This is however not the case if the two constraints at the endpoint are collinear,

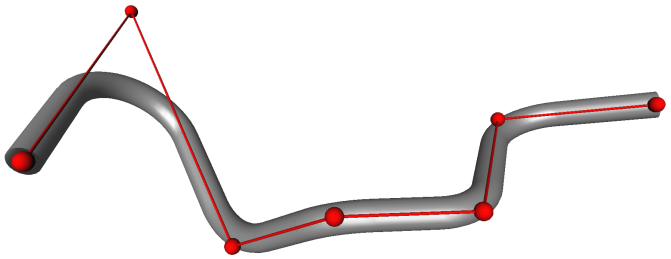


Figure 7. A tube and its control polygon.

i.e., if the two endpoints can be connected by a line with the same direction as the two end tangents.

A geometric explanation for this problem can be given as follows: since the line connecting the two endpoints has the same direction as the two end tangents (modulo sign), the curve between the constraints gains an additional degree of freedom by rotation around this axis. This shows that no parametrization of an arc-line spline can exist which achieves local control of parameters.

It is worth noting that the same type of problem also arises for circular arc splines in $\mathbb{R}^2$. In the general case, a biarc with fixed end points and tangents has only one degree of freedom for choosing the junction point. Now consider a biarc with coinciding start and end points and tangents. In this case, every point in $\mathbb{R}^2$ is a valid junction point.

5.2. *Control polygon representation*

CAD engineers most commonly represent tube geometries using control polygons, whose vertices are the intersection points of the lines defined by the straight line segments of the spine curve (see Fig. 7). For curves with constant arc radius, one additional parameter defines the arc radius. For general arc-line splines, there is one arc radius parameter for every vertex of the control polygon except the first and the last.

This representation, however, has serious drawbacks: first, it can not express all possible spine curves; in particular, bends with an apex angle of π or more cannot be represented this way. Another problem is that not every control polygon describes a proper tube: if two vertices are too close together, the two arcs corresponding to these vertices may intersect.

5.3. *LRA (length, rotation, angle) representation*

A popular format for describing bending data for tubes is the LRA representation (Fig. 8, see also [26]). This is the representation used by bending machines, since it naturally describes the motion sequence of the bending machine. For an arc-line spline with n arcs, it consists of a sequence of $n+1$ triples (l, r, a) denoting the length l of a straight segment, the apex angle a of a circular arc, and the rotation r of the normal (in the Frenet frame) of an arc segment to its predecessor. For two consecutive arc segments (or arcs at the beginning or end of the curve), a line segment with length $l = 0$ is used.

By convention, the first triple has $r = 0$, because for the first arc no rotation of the plane normal to the previous arcs is defined. Moreover, the last triple usually has $a = 0$ and $r = 0$, meaning that it only defines a line segment but no arc segment. This means that the LRA representation defines arc-line splines with alternating arc and line segments and starting and ending with a line segment. This is a natural assumption in the context of bending tubes.

Again, for constant arc radii one additional parameter is required, whereas for general arc-line splines, quadruples (l, r, a, R) with bend radius R are used instead.

This information describes any arc-line spline up to rigid transformations in $\mathbb{R}^3$. The starting point and orientation of the fist line segment of the curve is given by a point $p = (x, y, z) \in \mathbb{R}^3$ and a unit quaternion $q = (q_a + i\,q_b + j\,q_c + k\,q_d) \in \mathbb{H}$. The rotation represented by the quaternion is given by $q\,v\,q^{-1}$, where v is the vector to be rotated, considered as a quaternion with zero real part. Since the whole configuration space of arc-line splines is covered by the LRA representation, it is free of the drawbacks of the control polygon representation.

6. Least-squares distance fitting

After computing an arc-line spline approximation of our spine curve polygon as explained in Section 4, we have determined the combinatorial structure of the spine curve. Using a nonlinear optimization algorithm (Levenberg-Marquardt), we now modify the parameters of the spine curve to minimize the least squares orthogonal distance from the tube surface to the samples of the input point cloud. The Jacobian of the distance energy function is approximated using finite differences.

For reasons explained in Section 5, we use the LRA representation of the tube as parameters for the optimization. An analogous technique was proposed by Hoschek [15] for least-squares fitting of an $\mathbb{R}^2$ arc spline.

7. Outlier classification

During the various processing stages of our pipe surface reconstruction framework, outliers can occur due to several reasons. The raw scan data often contains a number of outlier samples that do not belong to the surface and are caused by technical deficiencies or human errors during the scanning process. Moreover, the scanning data is noisy and can show distortions

x_0	y_0	z_0	
q_a	q_b	q_c	q_d
l_1	$-$	a_1	R_1
l_2	r_2	a_2	R_2
	$\vdots$		
l_n	$-$	$-$	$-$

x_0	y_0	z_0	R
q_a	q_b	q_c	q_d
l_1	$-$	a_1	
l_2	r_2	a_2	
	$\vdots$		
l_n	$-$	$-$	

Figure 8. LRA representation of arc-line splines with variable and constant arc radius.

significantly above the expected noise level, caused by unfavorable circumstances such as bad surface reflectance properties or steep angle of incidence of the laser line. Irregular and noisy sampling can give rise to bad estimates of normals and shape operators. As a result, errors in the input data can also affect convergence of the cylinder fitting step due to bad initial guesses.

By imposing a set of constraints to the data in each processing step, determined by thresholds for several locally evaluated quantities, we can classify outliers and significantly improve robustness of our geometry processing pipeline.

Estimation of normals and shape operators is done by polynomial fitting as described in Section 2.1. To obtain robust estimates for these quantities, a sufficiently large number of samples is required. We use a ball with radius $\frac{2}{3}r_P$ as the neighborhood and require the sample count in the neighborhood to be at least 10.

The maximum curvature κ_{max} obtained by polynomial fitting should be close to the inverse of the pipe radius $\frac{1}{r_P}$. If the value deviates by a factor of more than 0.1 from the expected curvature, we consider the point to be an outlier. The standard deviation of the samples from the fitting polynomial also provides a criterion for outlier classification.

After MLS projection of the samples onto the spine curve, we have yet another criterion to check for outliers in the data. The distance between the projected sample and its original position should be close to the pipe radius. If this is not the case, the sample is discarded as an outlier. This error can stems from both displacement of an individual sample p from the surface and from the deviation of the best fitting cylinder's axis to the actual spine curve. We set our threshold for outlier classification using this error to 0.1 r_P.

8. Results

We have tested our algorithm on a benchmark data set consisting of scans of 30 different tubes, with radii ranging from 2 mm to 50 mm. Our algorithm was able to determine the correct segmentation of the tubes in 28 of 30 cases. In the two cases where the algorithm failed, it produced one and two additional arc segments, respectively, while still properly approximating the spine curve. Figure 9 shows the results of a few interesting examples from our test data set.

After optimization by least-squares distance fitting, the median distance of input surface samples to the reconstructed surface was between 50 and 150 μm, which is in the order of magnitude of the accuracy of the used laser scanners (FARO Platinum Laser ScanArm and KonicaMinolta VIVID 9i). Figures 9 (g) and (h) visualize the distances of the input samples to the reconstructed surface. The measurement is accurate enough to make the deformation of the tube caused by the bending visible. It is clearly recognizable that the most significant deviation of the samples from the model surface is caused by this deformation.

Running time of the algorithm pipeline is clearly dominated by the spine curve reconstruction and the least squares minimization. For a typical scan of a tube with 4 mm pipe radius and 268 mm length, consisting of 6005 samples, shown in Figures 3 and 9 (g), reconstruction of the spine curve took about 16 seconds on a 2 GHz PC, while least-squares fitting took 34 seconds. All other steps of the reconstruction (curve reconstruction, curve simplification, and arc-line spline approximation) took less than one second in total. The polygonal spine curve approximation consisted of 471 vertices before simplification and 30 vertices after simplification.

Running time for the least-squares optimization is proportional to the number of samples and the number of parameters of the LRA description. This means that for complex tube objects, such as the ones shown in Figures 9 (c) and (d), optimization can take several minutes. This could however be improved by employing more elaborate optimization strategies than the one used in our implementation.

9. Future work

As we mentioned above, the least-squares fitting of the reconstructed tube surface to the input point cloud still leaves room for enhancements. A possible direction would be the use of constrained optimization methods, such as interior point methods or other barrier methods (see [27]). This would allow to optimize in the configuration space $S = L \times A \times \cdots \times L$ of the segments, while enforcing the G^1 continuity and the ϵ tolerance of the solution using constraints.

There is an increasing trend in the industry to use more complicated shapes of tubes. Several assumptions, such as a constant tube radius or a circular cross section of the tube, do not necessarily hold anymore. It would be interesting to extend our method to these more general cases.

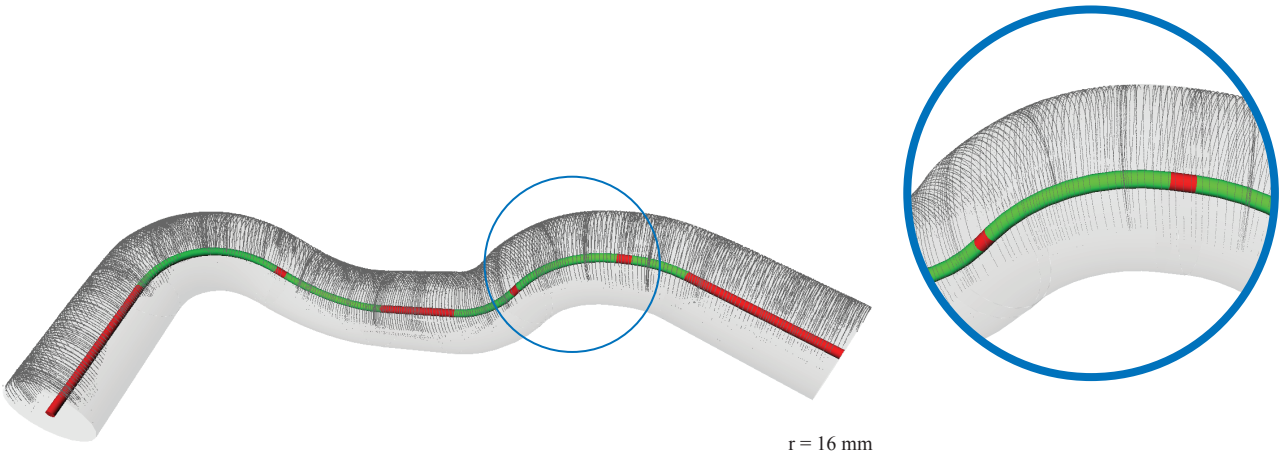
10. Conclusion

We present a complete process to parametric reconstruction of bent tube surfaces. We propose a moving least squares method to compute the spine curve of a pipe surface from a point cloud of the surface. We then give a heuristic to the problem of approximating a polygonal curve by a G^1 continuous spatial arc-line spline. We construct a graph representing a certain subset of all possible solutions in a greedy way and then find the optimal solution contained in this subset.

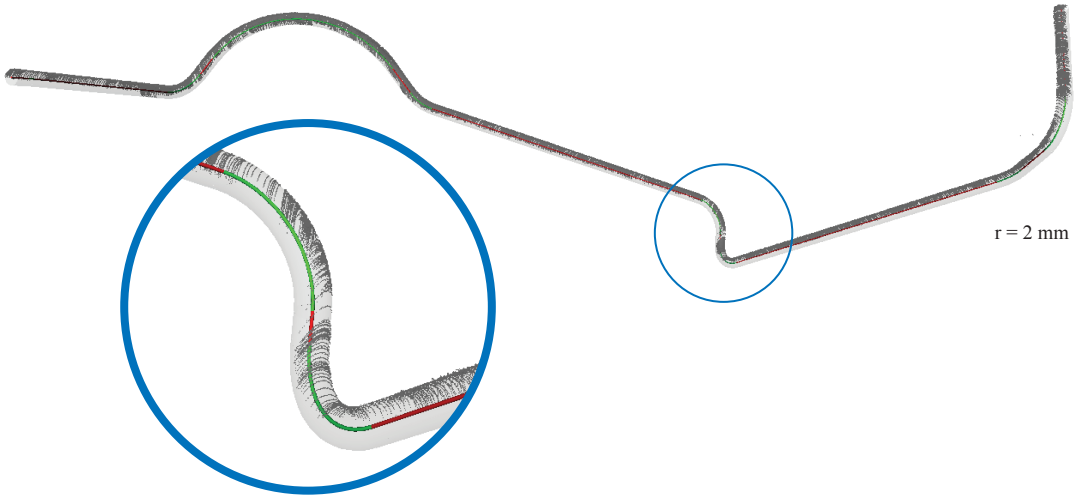
Our algorithm has been implemented in a software product of our industry partner and is already used in the tubing process of several car manufacturers.

Acknowledgments

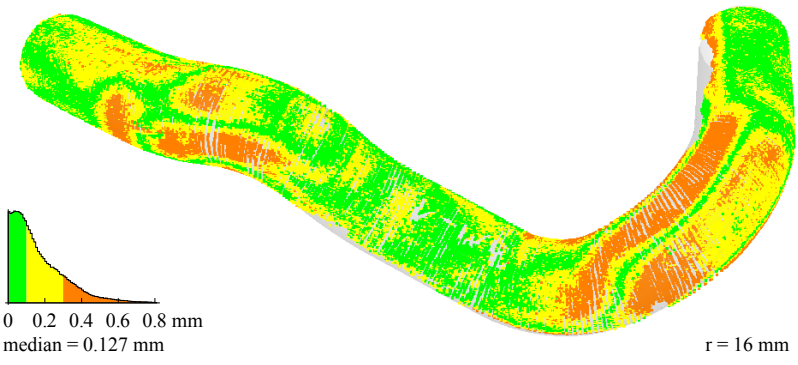
The work of Ulrich Bauer was supported by TubeExpert Ltd. Konrad Polthier was supported by the DFG Research Center MATHEON "Mathematics for key technologies" in Berlin. Thanks to Martin Kavalar from TubeExpert for providing scan data.



(a)

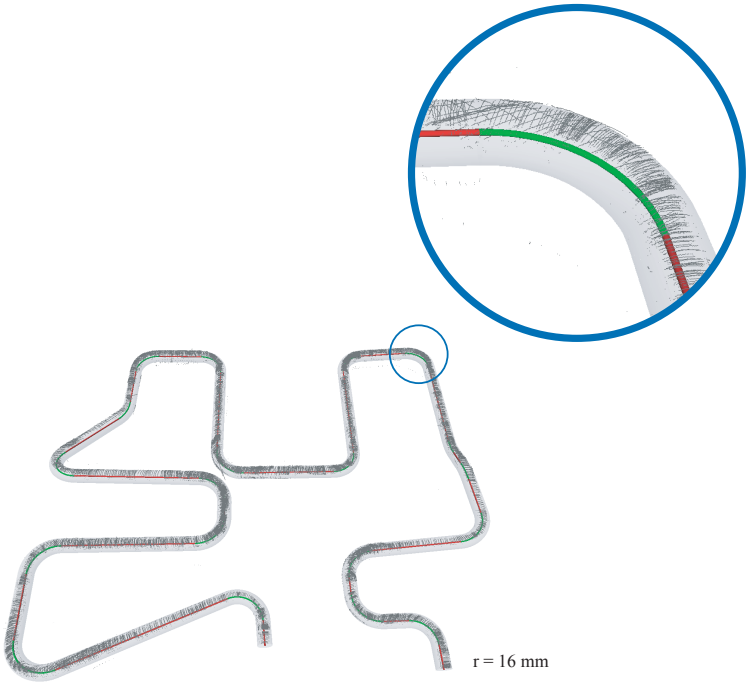


(b)

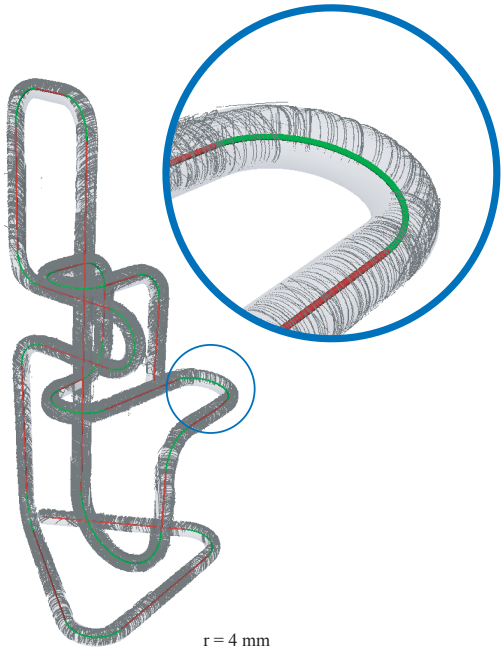


(c)

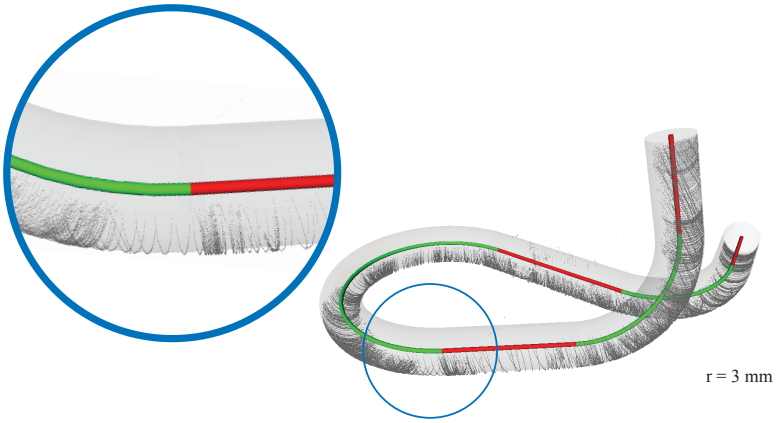
Figure 9. Results of our algorithm on difficult real-world data. Our method can reconstruct thick tubes (a) just as well as with thin wires (b). It can handle fairly complicated tubes (d,e) and special cases such as multiple consecutive bends joined by very short straight segments (a), bends with angles greater than π (f), and very small angles (g). Coloring the distances between the samples and the reconstructed surface clearly shows that deviations are mainly caused by deformation of the tubes at the bends (c,h). (Continued on next page.)



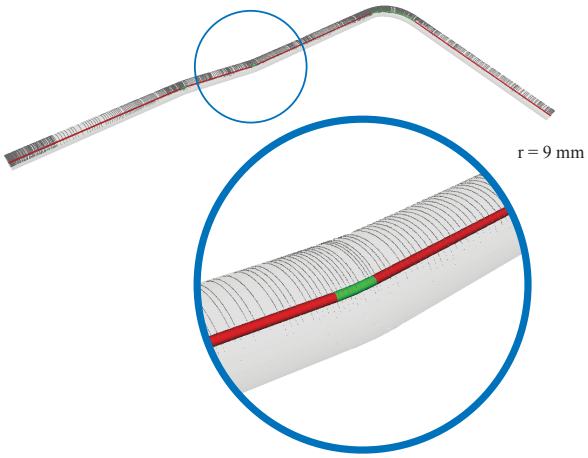
(d)



(e)



(f)



(g)

(h)

References

[1] N. Amenta, S. Choi, R. K. Kolluri, The power crust, in: SMA '01: Proceedings of the sixth ACM symposium on Solid modeling and applications, ACM Press, 2001, pp. 249–266.

[2] A. Adamson, M. Alexa, Approximating and intersecting surfaces from points, in: SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, Eurographics Association, 2003, pp. 230–239.

[3] W. E. L. Grimson, T. Lozano-Perez, N. Noble, S. J. White, An automatic tube inspection system that finds cylinders in range data, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1993, pp. 446–452.

[4] F. Goulette, Automatic cad modeling of industrial pipes from range images, in: First International Conference on Recent Advances in 3-D Digital Imaging and Modeling (3DIM '97), IEEE Computer Society, Los Alamitos, CA, USA, 1997, pp. 229–233.

[5] I.-K. Lee, Curve reconstruction from unorganized points, Comput. Aided Geom. Des. 17 (2) (2000) 161–177.

[6] I.-K. Lee, K.-J. Kim, Shrinking: Another method for surface reconstruction, in: GMP '04: Proceedings of the Geometric Modeling and Processing 2004, IEEE Computer Society, 2004, p. 259.

[7] D. Levin, Mesh-independent surface interpolation, in: G. Brunnett, B. Hamann, H. Müller, L. Linsen (Eds.), Geometric Modeling for Scientific Visualization, Springer, 2003, pp. 37–49.

[8] N. D. Cornea, D. Silver, P. Min, Curve-skeleton properties, applications, and algorithms, IEEE Transactions on Visualization and Computer Graphics 13 (3) (2007) 530–548.

[9] N. Amenta, M. Bern, D. Eppstein, The crust and the beta-skeleton: combinatorial curve reconstruction, Graphical Models and Image Processing 60 (2) (1998) 125–135.

[10] T. K. Dey, P. Kumar, A simple provable algorithm for curve reconstruction, in: SODA '99: Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 1999, pp. 893–894.

[11] D. H. Douglas, T. K. Peucker, Algorithms for the reduction of the number of points required to represent a digitized line or its caricature, The Canadian Cartographer 10 (2) (1973) 112–122.

[12] D. Eu, G. T. Toussaint, On approximating polygonal curves in two and three dimensions, CVGIP: Graph. Models Image Process. 56 (3) (1994) 231–246.

[13] P. K. Agarwal, S. Har-Peled, N. H. Mustafa, Y. Wang, Near-linear time approximation algorithms for curve simplification, Algorithmica 42 (3) (2005) 203–219.

[14] L. J. Guibas, J. E. Hershberger, J. S. B. Mitchell, J. S. Snoeyink, Approximating polygons and subdivisions with minimum link paths, Internat. J. Comput. Geom. Appl 3 (1993) 383–415.

[15] J. Hoschek, Circular splines, Computer-Aided Design 24 (11) (1992) 611–618.

[16] A. Safonova, J. Rossignac, Compressed piecewise-circular approximations of 3d curves, Computer-Aided Design 35 (6) (2003) 533–547.

[17] S. Drysdale, G. Rote, A. Sturm, Approximation of an open polygonal curve with a minimum number of circular arcs, in: Proceedings of the 22nd European Workshop on Computational Geometry (EWCG), IEEE Computer Society, 2006, pp. 25–28.

[18] D. S. Meek, D. J. Walton, Approximation of discrete data by G^1 arc splines, Computer-Aided Design 24 (6) (1992) 301–306.

[19] L. Piegl, W. Tiller, Data approximation using biarcs, Engineering with Computers 18 (1) (2002) 59–65.

[20] M. Held, J. Eibl, Biarc approximation of polygons within asymmetric tolerance bands, Computer-Aided Design 37 (4) (2005) 357–371.

[21] K. Bolton, Biarc curves, Computer-Aided Design 7 (2) (1975) 89–92.

[22] F. Cazals, M. Pouget, Estimating differential quantities using polynomial fitting of osculating jets, Comput. Aided Geom. Des. 22 (2) (2005) 121–146.

[23] P. Krsek, G. Lukács, R. Martin, Algorithms for computing curvatures from range data, in: Proceedings of the 8th IMA Conference on The Mathematics of Surfaces, 1998, pp. 1–16.

[24] G. Lukács, R. Martin, D. Marshall, Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation, in: 5th European Conference on Computer Vision, Vol. 1406 of Lecture Notes in Computer Science, Springer, 1998, pp. 671–686.

[25] O. Daescu, New results on path approximation, Algorithmica 38 (1) (2003) 131–143.

[26] P. Chapman, State of the art cnc tube bending, TubeNet.org.
URL `http://www.tubenet.org.uk/technical/addison_m.html`

[27] J. Nocedal, S. J. Wright, Numerical Optimization, Springer, 1999.