

# Parametric Reconstruction of Bent Tube Surfaces

Ulrich Bauer  
Mathematical Geometry Processing  
Freie Universität Berlin  
Email: ubauer@mi.fu-berlin.de

Konrad Polthier  
Mathematical Geometry Processing  
Freie Universität Berlin  
Email: polthier@mi.fu-berlin.de

**Abstract**—We present a method for parametric reconstruction 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.

**Index Terms**—Pipe Surfaces, Tubes, Surface Reconstruction, Curve Approximation, Moving Least Squares.

## I. INTRODUCTION

Reconstruction of surfaces from point clouds has attracted enormous interest in recent years. Numerous algorithms have been proposed for either interpolation or approximation of 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.

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. I 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*.

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

An important application for our algorithm is 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.

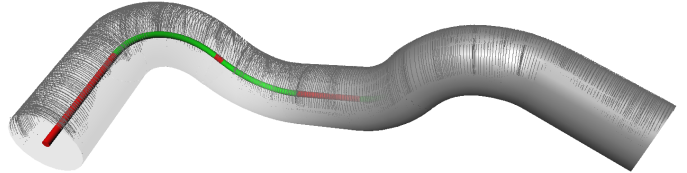


Figure 1. Surface of a bent tube—a pipe surface consisting of cylinder (red spine) and torus (green spine) segments—automatically reconstructed using our algorithm from a laser scan (samples in grey).

### A. Related work

Reconstruction of pipe surfaces from unorganized points has been investigated by Lee in [3] and later extended to canal surfaces in [4]. The approach used here is 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 same paper, Lee [3] also proposed to use the *moving least squares* approximation technique to smooth 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 [5], [2] and references therein. We will adapt this method to find an approximate point cloud of the spine curve.

Several methods have been proposed for computing an approximate medial axis [1] or a curve skeleton [6] 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 drawback to these approaches. Most important, the mentioned methods are not robust to noise and outliers in the input data. Moreover, they are computationally quite expensive. Third, we want 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 reconstruction of planar curves from unorganized point sets has been presented in the seminal paper by Amenta et al. [7] based on Voronoï diagrams and Delaunay triangulations. Another simple but provably correct method, which also works for curves in higher dimensions, is the

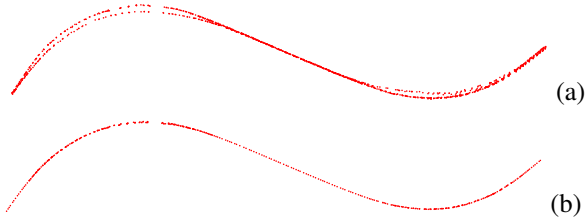


Figure 2. Spine point cloud obtained using (a) Lee’s method (before smoothing) and (b) our method. While in (a) the noise is considerable, in (b) the result is sharp and does not require smoothing.

NN-Crust algorithm, proposed by Dey and Kumar [8]. This is the algorithm we used in our implementation.

The problem of approximating curves by 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  $\epsilon$  in some distance measure. The heuristic by Douglas and Peucker [9] is the standard technique for this problem, although it has no optimality guarantee and a worst-case running time of  $\mathcal{O}(n^2)$ . Optimal algorithms for simplification of space curves, where the vertices of the output polygon are a subset of the input vertices, are presented by Eu and Toussaint [10] with a running time of  $\mathcal{O}(n^3)$ . Agarwal et al. [11] gives an efficient greedy approximation algorithm running in  $\mathcal{O}(n \log n)$  and requiring at most as many line segments as the optimal  $2\epsilon$  approximating polygonal chain. For the *weak simplification* problem, where the output vertices can be chosen freely, Guibas et al. [12] proposed an exact algorithm for planar curves with a running time of  $\mathcal{O}(n^2)$ ; the weak simplification problem for space curves is still unsolved.

Segmentation and approximation of planar and space curves by  $G^0$  and  $G^1$  continuous circular arcs and line segments has been investigated by several authors, in most cases without any guarantees about the optimality of the solution. Hoschek [13] finds a  $G^1$  planar approximation; Rossignac [14] approximates a space curve by  $G^0$  continuous circular arcs. Sturm et al. [15] computes a minimum segment count  $G^0$  planar approximation with junction point chosen from the input vertices. [16], [17], [18] compute  $G^1$  planar approximations using biarcs. Biarcs, first introduced in [19], 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 previously published algorithms are either not suited for certain problem instances such as arc segments with an apex angle greater than  $\pi$ , or only produce arc splines consisting of biarcs.

## B. Contributions

We propose a complete scheme 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 with distance  $< \epsilon$ . It uses estimated tangent lines of the input curve and computes a minimum segment number approximation 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  $\pi$ .

## C. 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 III. 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 [8], as shown in Fig. 3 (d).
- The polygonal curve is optionally simplified using Eu and Toussaint’s algorithm [10] to reduce the problem complexity for further computations.
- This polygonal curve is then approximated using an arc-line spline as described in Section IV, see Fig. 3 (e).
- Finally, the parametric description of the bent tube surface (see Section V) is modified to optimize the least squares distance of the surface to the input samples using the Levenberg-Marquardt nonlinear optimization method (Section VI). The final output of our reconstruction is shown in Fig. 3 (f).

## II. REVIEW OF ESTIMATED DIFFERENTIAL QUANTITIES ON POINT CLOUDS

### A. 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 [20] gives a detailed analysis of this popular technique, and Lukácz [21] 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 [5], [2].

To obtain an estimate of the tangent plane, let  $\theta : \mathbb{R}^+ \rightarrow \mathbb{R}^+$  be a monotonically decreasing distance weighting function with support  $[0, \epsilon]$ . We consider the local neighborhood  $N_p = \{p_1, \dots, p_n\}$  of a point  $p$  consisting of all sample points inside a ball of radius  $\epsilon$  centered at  $p$ . A principal component

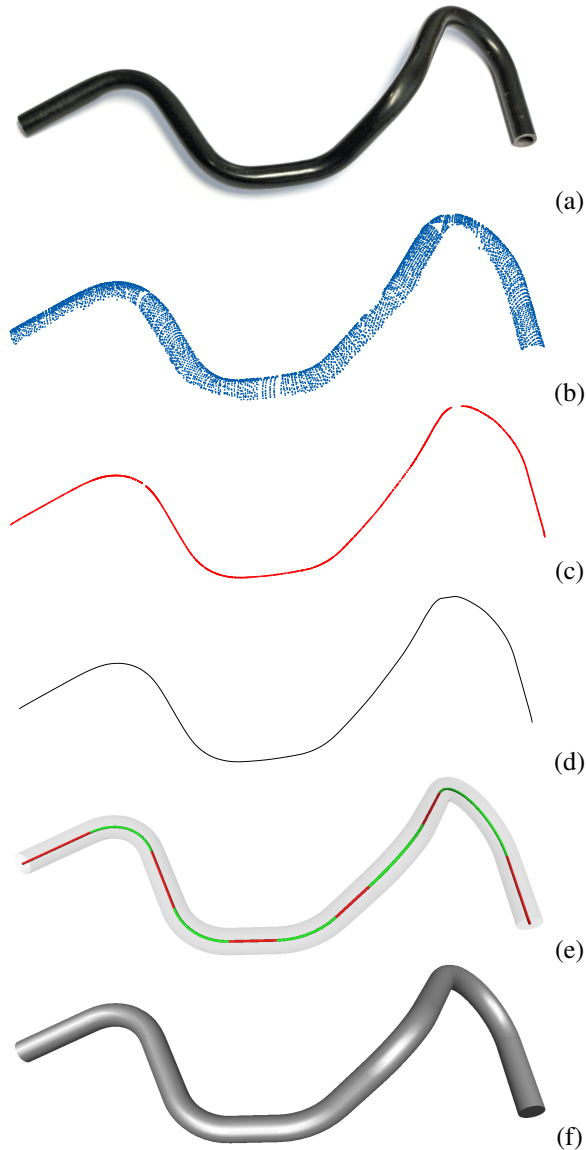


Figure 3. Reconstruction of a tube. (a) A photograph of the original tube. (b) A single-view, irregular scan of the tube surface. (c) Projection of the point cloud onto the MLS spine approximation. (d) Reconstruction of the spine curve. (e) Approximation of the spine curve by a  $G^1$  continuous arc-line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

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]: the estimated normal  $\nu$  of this tangential plane to a point  $p$  is the solution of the optimization problem

$$\min_{\|\nu\|=1} \sum_k \langle \nu, (p - p_k) \rangle^2 \theta(\|p - p_k\|)$$

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

$$w_{ij} = \sum_k \langle e_i, (p - p_k) \rangle \langle e_j, (p - p_k) \rangle \theta(\|p - p_i\|) .$$

This estimated tangential plane is only used as the reference

domain to compute a bivariate polynomial approximation to the neighborhood  $N_p$  with the estimated tangential plane as reference domain. Let  $\Pi_2^2$  denote the bivariate polynomials of degree 2. Using weights  $\theta_k = \theta(\|p - p_k\|)$ , 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

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

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 tangential plane.

### B. 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  $\kappa_{\max}$ .

### III. 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 point on the surface to its closest point on the spine curve. This spine curve point cloud can then be smoothed and reconstructed as a polygonal chain using any curve reconstruction algorithm.

Our MLS projection is based on the procedure described in [5]. This method consists of finding a local polynomial

approximation to the neighborhood of a point  $p$  as described in Section II-A, 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 [22] to points in the neighborhood of a point  $p_c$  close to the spine curve, and then projecting this point onto the axis of the 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 find the cylinder  $C \in Cyl_r$  minimizing the weighted least-squares distance to the samples using a locally supported distance weighting function  $\theta$ :

$$\min_{C \in Cyl_r} \frac{\sum_i d(C, p_i)^2 \theta(\|P(x, a_C) - p_i\|)}{\sum_i \theta(\|P(x, a_C) - p_i\|)}$$

where  $d(C, p)$  is the (signed or unsigned) distance function of a point  $p$  to the cylinder  $C$ ,  $a_C$  is the axis of the cylinder  $C$ , and  $P(x, a_C)$  is the projection of  $x$  onto the axis. Note that this is the center of the weighting function, which is therefore not fixed, but dependent 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.

- 1) Find an initial estimate for a point  $x$  and a tangent direction  $t$  of the spine curve.
- 2) Find the best-fitting cylinder  $C_{\min}$  with radial weights centered at  $P(x, a_{\min})$ , using  $(x, t)$  as initial estimates.
- 3) 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.

#### A. 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  $\kappa_{max}$  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 II-B. This assumption also implies that smoothness of the surface is the same as of the spine curve.

For cylinder fitting with fixed radius, we compute 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.

#### B. Choosing a neighborhood size

We now discuss appropriate choices for the neighborhood size, i.e. the support of the weighting function  $\theta$ . Similar to MLS curves, a large support for  $\theta$  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  $1.2 r_B$  (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.

#### C. 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.

We can estimate a tangent direction of the spine curve at the projection of  $p$  onto the curve from the shape operator estimation at a point  $p$ . It is easy to observe that the tangent direction of the spine curve can be determined on the pipe surface itself using its principal curvature directions. Since we know 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.

#### D. A starting point for cylinder fitting

Estimates of the surface normal  $\nu$  (Section II-A) and the pipe radius  $r_P$  (Section III-A) provide an initial point 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 estimates 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.

### IV. 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  $\epsilon$  to each of the vertices of the input polygon.

#### A. Algorithmic framework

The algorithm is based on a general framework used in several polygon simplification algorithms [10], [23]. The main idea is to construct a shortcut graph  $G = (V, E)$  over the

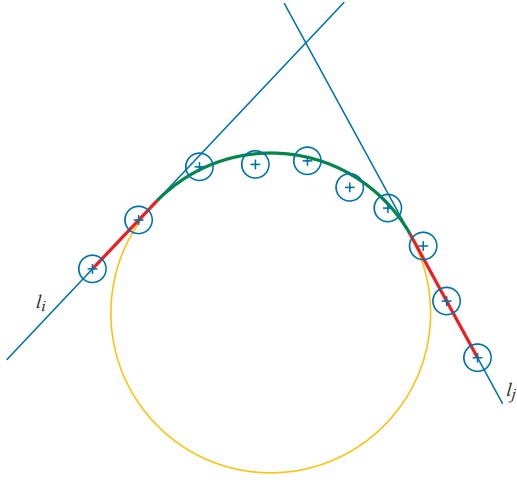


Figure 4. A circular arc and two line segments approximating a set of points.

vertices of the input curve, with an edge  $e_{ij} \in E$  if and only if the chain  $[p_i, p_{i+1}, \dots, p_j, p_{j+1}]$  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  $\mathcal{O}(n)$  space, see [23].

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 should be an edge  $e_{ij}$  in the graph (see Subsection IV-B).

Note that we get a different modified tangent line  $t'_j$  for each edge  $e_{ij}$ . To ensure a complete  $G^1$  smooth 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  $j$ .

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 is similar to the polygon simplification problem, 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.

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, the endpoints of the segments can, in general, not be chosen from the vertex set of the input curve.

---

**Algorithm 1** Computing an approximating arc-line spline

---

**Require:** A polygonal chain  $C = [p_1, p_2, \dots, p_n]$  and a tolerance value  $\epsilon$

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

```

for all  $1 \leq i < (n - 1)$  do
  for all  $j > i$  do
    let  $P$  be the plane spanned by  $\{p_i, p_{i+1}, p_{j+1}\}$ 
    for all  $p_k \in \{p_i, p_{i+1}, \dots, p_{j+1}\}$  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  $p'_k \in \{p'_{i+1}, p'_{i+2}, \dots, p'_j\}$  do
      compute all circles tangential to:
         $l_1, l_2$ , and the circle around  $p'_k$  with radius  $\epsilon'_k$ 
      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
       $V = V \cup V_k$ 
    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
 $\{i_1, i_2, \dots, i_m\} = \{1, \dots, prev(prev(n - 1)), prev(n - 1), n - 1\}$ 
for all  $1 \leq k < m$  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  $\{l_1, a_1, l_2, a_2, \dots, l_m\}$ 

```

---

## B. Building the graph

We will now describe how to compute the edges  $e_{ij}$  of the shortcut graph. Given two coplanar lines on a common plane  $P$ , we have to decide if a polygonal chain  $[p_i, p_{i+1}, \dots, p_j, p_{j+1}]$  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  $\epsilon$  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  $\|p_k - p'_k\| > \epsilon$ , then the chain cannot be approximated. Else, let  $\epsilon'_k = \sqrt{\epsilon^2 - \|p_k - p'_k\|^2}$ . This is the maximum allowed distance for the vertex  $p_k$  to the curve in the plane, since  $\epsilon'_k{}^2 + \|p_k - p'_k\|^2 = \epsilon^2$ .

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  $\epsilon'_k$  around the vertex  $p'_k$ . It is easy to see that all circles touching the two lines and having distance less than  $\epsilon'_k$  from  $p'_k$  have their centers between a pair of centers 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  $\epsilon'_k$ -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: an arc with center beyond the upper solution is still valid, since  $p'_k$  has distance less than  $\epsilon$  to a line segment on  $l_1$ .

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

## C. Constructions

The Apollonius circles for the LLC problem (Fig. 5) are constructed as follows: The 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 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 line–line–point (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  $l_1, l_2$ , with center  $M$ . The line  $OP$  joining the intersection  $O$  of  $l_1$  and  $l_2$  and the

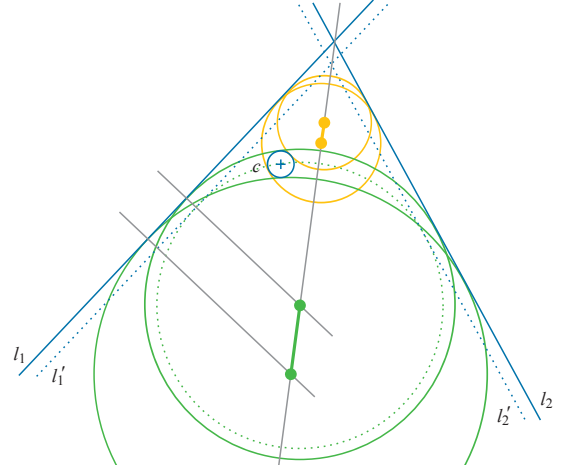


Figure 5. The Apollonius LLC problem and centers of approximating arcs. The solid circles (green and yellow) are tangent to the lines  $l_1$  and  $l_2$  and the circle  $c$ . The dotted lines and circle show the auxiliary LLP construction.

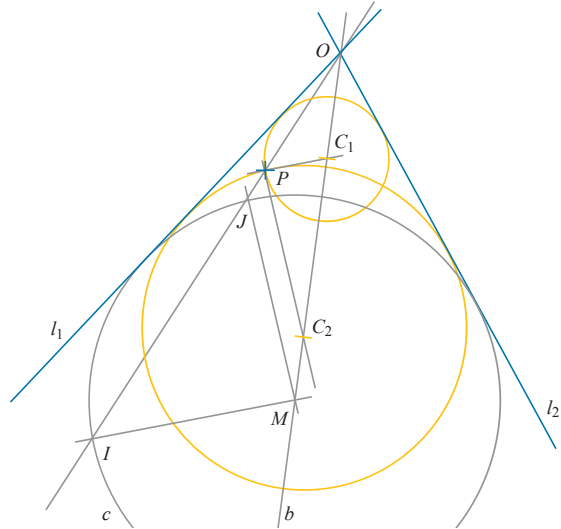


Figure 6. The Apollonius LLP problem. The yellow circles touch the two blue lines  $l_1$  and  $l_2$  and the point  $P$ .

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

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

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.

#### D. Complexity

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

#### E. Preprocessing

To speed up the computation and to stabilize the estimation of the tangent lines, we run a polygon simplification algorithm as a preprocessing step. We implemented the algorithm from [23], which computes a minimum segment approximation using a subset of the input vertices with distance less or equal to  $\epsilon$  in time  $\mathcal{O}(n^3)$ . We observed a real-world running time in the order of magnitude of about  $\mathcal{O}(n^2)$ , using space  $\mathcal{O}(n)$ .

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.

### V. 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.

#### A. 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., every parameter has only influence to a constant number of segments of the curve. For polygonal chains and other piecewise defined spline curves such as Bézier 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 VI).

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

$$S = L \times A \times \dots \times L$$

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 as 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 tangent at this point is a unit vector in  $\mathbb{R}^3$  and can be identified with a point on the 2-dimensional unit sphere  $\mathbb{S}^2$ . Therefore, the dimension of the configuration space of an arc-line spline is

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

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 parametrization of the curve that has the form

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

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.

A necessary condition for the existence of such a parametrization scheme is that for any choice of constraints, the subspace  $S_c$  is a connected  $(d - 10)$ -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, 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 singularity 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.

#### B. Control polygon representation

CAD engineers most commonly represent tube geometries as 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  $\pi$  or more cannot be described in 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.

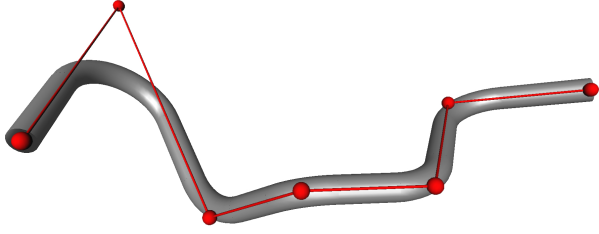


Figure 7. A tube and its control polygon.

### C. LRA (length, rotation, angle) representation

A popular format for describing bending data for tubes is the LRA representation (Fig. 8, see also [24]). 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 first 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.

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

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

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

## VI. LEAST-SQUARES DISTANCE FITTING

After computing an arc-line spline approximation of our spine curve polygon as explained in Section IV, 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 V, we use the LRA representation of the tube as parameters for the optimization. An analogous technique was proposed by Hoschek [13] for least-squares fitting of an  $\mathbb{R}^2$  arc spline.

## VII. 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 don't 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 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. Later, it can affect convergence of the cylinder fitting step due to bad initial guesses as well as errors in the data.

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 II-A. 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  $\kappa_{max}$  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.

After MLS projection of the samples onto the spine curve, we have 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 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  $0.1 r_P$ .

## VIII. 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  $\mu\text{m}$ , 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 deviations of the samples from the model surface are 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.

## IX. 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 is supported by the DFG Research Center MATHEON “Mathematics for key technologies” in Berlin. Thanks to Martin Kavalar from TubeExpert for providing scan data.

## REFERENCES

- [1] N. Amenta, S. Choi, and 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 and 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] I.-K. Lee, “Curve reconstruction from unorganized points,” *Comput. Aided Geom. Des.*, vol. 17, no. 2, pp. 161–177, 2000.
- [4] I.-K. Lee and 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.
- [5] D. Levin, “Mesh-independent surface interpolation,” in *Geometric Modeling for Scientific Visualization*, G. Brunnett, B. Hamann, H. Müller, and L. Linsen, Eds. Springer, 2003, pp. 37–49.
- [6] N. D. Cornea, D. Silver, and P. Min, “Curve-skeleton properties, applications, and algorithms,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 3, pp. 530–548, 2007.
- [7] N. Amenta, M. Bern, and D. Eppstein, “The crust and the beta-skeleton: combinatorial curve reconstruction,” *Graphical Models and Image Processing*, vol. 60, no. 2, pp. 125–135, 1998.
- [8] T. K. Dey and 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.
- [9] D. H. Douglas and T. K. Peucker, “Algorithms for the reduction of the number of points required to represent a digitized line or its caricature,” *The Canadian Cartographer*, vol. 10, no. 2, pp. 112–122, 1973.
- [10] D. Eu and G. T. Toussaint, “On approximating polygonal curves in two and three dimensions,” *CVGIP: Graph. Models Image Process.*, vol. 56, no. 3, pp. 231–246, 1994.
- [11] P. K. Agarwal, S. Har-Peled, N. H. Mustafa, and Y. Wang, “Near-linear time approximation algorithms for curve simplification,” *Algorithmica*, vol. 42, no. 3, pp. 203–219, 2005.
- [12] L. J. Guibas, J. E. Hershberger, J. S. B. Mitchell, and J. S. Snoeyink, “Approximating polygons and subdivisions with minimum-link paths,” Vancouver, BC, Canada, Canada, Tech. Rep., 1992.
- [13] J. Hoschek, “Circular splines,” *Computer-Aided Design*, vol. 24, no. 11, pp. 611–618, 1992.
- [14] A. Safonova and J. Rossignac, “Compressed piecewise-circular approximations of 3d curves,” *Computer-Aided Design*, vol. 35, no. 6, pp. 533–547, 2003.
- [15] S. Drysdale, G. Rote, and 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.
- [16] D. S. Meek and D. J. Walton, “Approximation of discrete data by  $G^1$  arc splines,” *Computer-Aided Design*, vol. 24, no. 6, pp. 301–306, 1992.
- [17] L. Piegl and W. Tiller, “Data approximation using biarcs,” *Engineering with Computers*, vol. 18, no. 1, pp. 59–65, 2002.
- [18] M. Held and J. Eibl, “Biacr approximation of polygons within asymmetric tolerance bands,” *Computer-Aided Design*, vol. 37, no. 4, pp. 357–371, 2005.
- [19] K. Bolton, “Biacr curves,” *Computer-Aided Design*, vol. 7, no. 2, pp. 89–92, 1975.
- [20] F. Cazals and M. Pouget, “Estimating differential quantities using polynomial fitting of osculating jets,” *Comput. Aided Geom. Des.*, vol. 22, no. 2, pp. 121–146, 2005.
- [21] P. Krsek, G. Lukács, and 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.
- [22] G. Lukács, R. Martin, and D. Marshall, “Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation,” in *5th European Conference on Computer Vision*, ser. Lecture Notes in Computer Science, vol. 1406. Springer, 1998, pp. 671–686.
- [23] O. Daescu, “New results on path approximation,” *Algorithmica*, vol. 38, no. 1, pp. 131–143, 2003.
- [24] P. Chapman, “State of the art cnc tube bending,” *TubeNet.org*. [Online]. Available: [http://www.tubenet.org.uk/technical/addison\\_m.html](http://www.tubenet.org.uk/technical/addison_m.html)

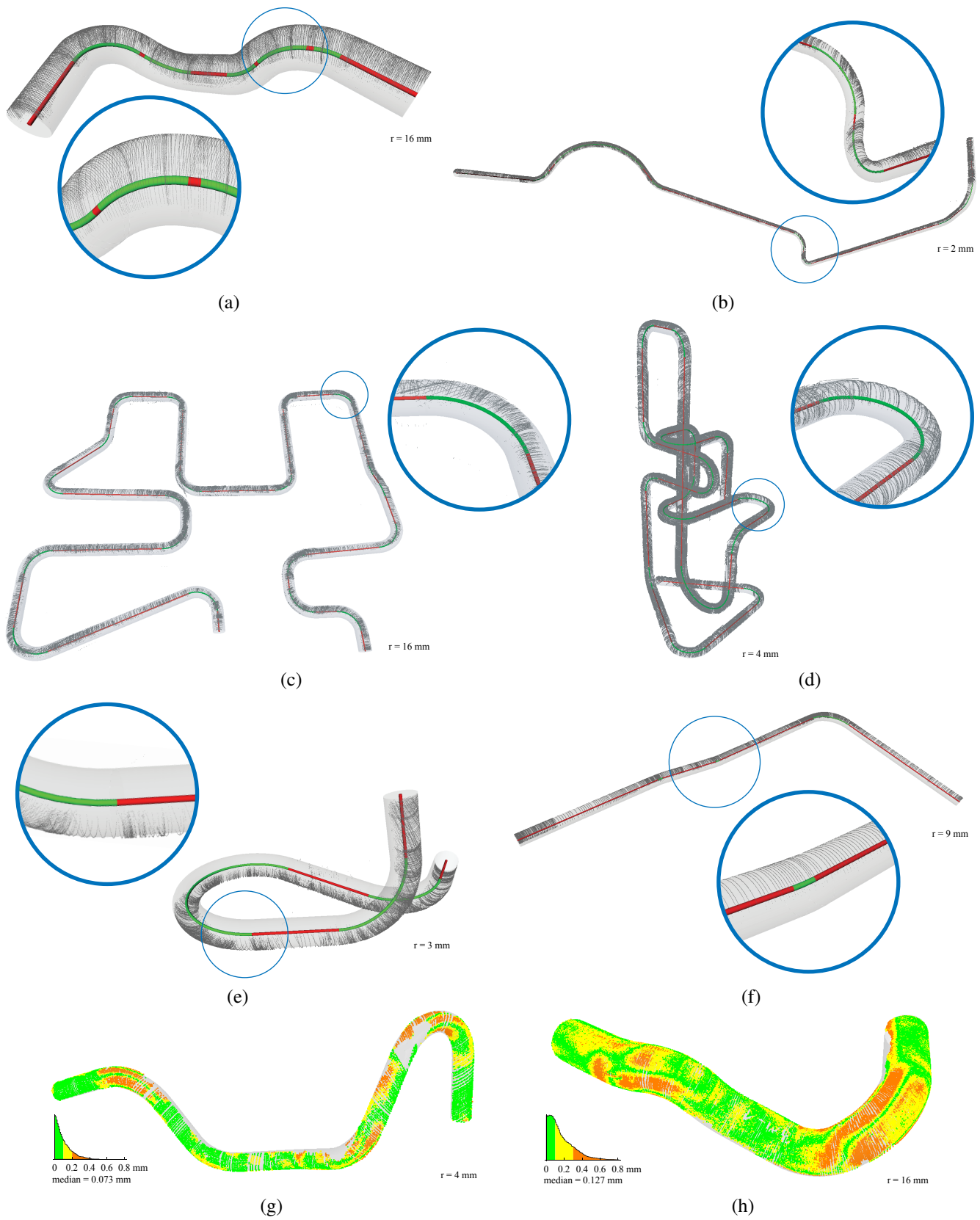


Figure 9. Results of our algorithm on difficult real-world data. Our method can reconstruct thick tubes (a) just as well as with thin wires (b). It can reconstruct fairly complicated tubes (c,d) and special cases such as multiple consecutive bends joined by very short straight segments (a), bends with angles greater than  $\pi$  (e), and very small angles (f). 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 (g,h).