Shrinking: Another Method for Surface Reconstruction

In-Kwon Lee
(corresponding author)
Dept. of Computer Science
Yonsei University
Seoul 120-749, South Korea
iklee@yonsei.ac.kr

Ku-Jin Kim
The Graduate School of
Information and Communication
Ajou University
Suwon 442-749, South Korea
kujinkim@ajou.ac.kr

Abstract

We present a method to reconstruct a pipe or a canal surface from a point cloud (a set of unorganized points). A pipe surface is defined by a spine curve and a constant radius of a swept sphere, while a variable radius may be used to define a canal surface. In this paper, by using the shrinking and moving least-squares methods, we reduce a point cloud to a thin curve-like point set which will be approximated to the spine curve of a pipe or canal surface. The distance between a point in the thin point cloud and a corresponding point in the original point set represents the radius of the pipe or canal surface.

1. Introduction

A *pipe surface* is defined as the envelope of a sphere with a constant radius r moving through a *spine curve* $\mathbf{C}(t)$. A *canal surface* is a generalization of a pipe surface, where a variable radius function $r(t)$ is used instead of r. Pipe and canal surfaces are used in many practical applications such as surface blending and transition surfaces between pipes [1, 2]. Furthermore, there are many real or synthetic objects which can be represented by only pipe or canal surfaces (see Figure 1). In this paper, we consider the problem of reconstructing a pipe or canal surface from an unorganized point cloud having no ordering or structure of the point elements. The input point cloud is usually scanned from a real pipe/canal surface by a 3D scanner.

The motivation of this work comes from recent research in reverse engineering area, attempting to reconstruct surfaces such as helical surfaces and surfaces of revolution [3], profile surfaces [4], developable surfaces [5], and planar faces [6]. The details about these series of works can be found in [7] and [8]. The reconstruction of the sweep surfaces generated by translational sweeping [9] is also discussed by other researchers. Instead of reconstructing spline surfaces or polygonal meshes, this research concentrates on reconstructing a profile (cross-section) curve and a kinematic motion that defines a trajectory of the moving profile curve. This procedural description of a geometric model reduces the size of the object database and makes design/manipulation processes efficient and easy.

Figure 1. Some objects created using only canal surfaces

Ramamoorthi and Arvo [10] suggested a system to reconstruct various kinds of generative models, including pipe and canal surfaces, from point clouds with the aid of a predefined user-specified hierarchy of the various generative models. Unlike their work, we suggest an algorithm which avoids serious user interaction by narrowing down our interest to reconstruct pipe or canal surfaces only. Our solution uses only local linear optimization procedures that are much more efficient and robust than complex nonlinear optimization methods.

In our previous papers [11, 12], we have roughly sketched out the idea behind reconstructing pipe surfaces (see Figure 2). First, an appropriate subregion R of a given point cloud is taken (Figure 2(a)). Then, a torus is fitted to R. Note that the local shape of a pipe surface fits a torus. The torus implies the radius of a swept sphere of the pipe surface. Each data point is translated by the radius towards a (target) spine curve along an estimated normal vector of the point (Figure 2(b)). After the translation, the point set

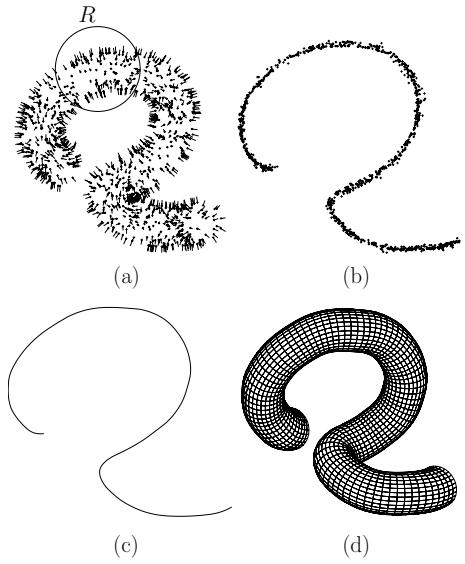

Figure 2. Pipe surface reconstruction using torus fit: (a) input points and estimated normal vectors – R represents a selected region where a torus is fitted, (b) translating each point by the radius of torus, (c) spine curve approximated from thin point cloud in (b), and (d) reconstructed pipe surface from the spine curve and the radius of the swept sphere.

has a very thin curve-like shape. This thin point cloud is approximated to a smooth spine curve (Figure 2(c)). In this method, automatic computation of a local region R does not work well in many cases; thus, in many cases, we need some sort of user interaction to specify an appropriate local region. Furthermore, this torus fitting technique is not appropriate to generalize for canal surface reconstruction.

In this paper, we introduce a *shrinking* method to find a spine curve and the (variable) radius of a swept sphere from a given point cloud. Let S denote a set of input points. Using the shrinking method, S is reduced to a point cloud $\hat{S}$ whose shape represents the spine curve of the pipe or canal surface. Then, the point cloud $\hat{S}$ is processed by a method called *moving least-squares*, which transforms a point cloud into a very thin curve-like shape. The output of the moving least-squares method, $\tilde{S}$, is approximated to a spine curve . The distance between two points of each corresponding point pair in S and $\tilde{S}$ represents the approximated radius of a pipe/canal surface.

We summarize the contributions of this paper as follows:

- Using the shrinking method, our algorithm eliminates the difficulty of selecting an appropriate local region for torus fitting.
- Almost the same algorithms can be applied to reconstruct a pipe and a canal surface.
- Our method uses only local linear optimization procedures that are much more efficient and robust than complex nonlinear optimization methods.

This paper is organized as follows. In the next two sections, we describe an algorithm to reconstruct a pipe surface and a canal surface from a given point cloud, respectively. In the fourth section, we show experimental results of the pipe/canal surface reconstruction. Finally, in the last section, we conclude this work and suggest future research directions.

2. Pipe Surface Reconstruction

A pipe surface $\mathbf{B}(s,t)$ with a spine curve $\mathbf{C}(t)$, $t \in I_2$, and a radius r is defined by

$$\mathbf{B}(s,t) = \mathbf{C}(t) + r\mathbf{N}(s,t), \quad s \in I_1,\ t \in I_2,$$

where for a fixed $t_* \in I_2$, $\mathbf{N}(s,t_*)$ represents the unit circle on the normal plane of the spine curve. The center of $\mathbf{N}(s,t_*)$ is $\mathbf{C}(t_*)$. Thus, a pipe surface is uniquely defined by a spine curve $\mathbf{C}(t)$ and a radius r.

Let $S = \{\mathbf{P}_i \mid i = 1,...,n\}$ denote a given set of points that is likely to be on a pipe surface. We assume that the unit normal vectors, $N = \{\mathbf{N}_i \mid i = 1,...,n\}$, for the points of S were already estimated. The estimation of point normals is a common subject in many reverse engineering problems; thus, various methods are already known [13, 14]. We have used a method to estimate $\mathbf{N}_i$ of $\mathbf{P}_i$: i) collecting a set T_i of the neighboring points of $\mathbf{P}_i$, ii) computing the regression plane of T_i by the least-squares method, and iii) taking the normal of the plane as $\mathbf{N}_i$. Without loss of generality, we assume that the normal vectors point to the outward direction of a pipe surface. It is also assumed that S represents a regular pipe surface without any self intersection [1]. This assumption is feasible because we only consider the case where an input point set is scanned from a real-world pipe object.

Our algorithm is sketched in Figure 3. The algorithm consists of three phases: *shrinking*, *moving least-squares*, and *spine curve approximation*. We describe the details of each stage in the rest of this section.

2.1. Shrinking

Basic Algorithm

The basic idea of this algorithm is that for a point $\mathbf{P}_i$ there exist neighboring points having almost equal normal directions to the normal direction $\mathbf{N}_i$ of $\mathbf{P}_i$. Let T_i be a set of

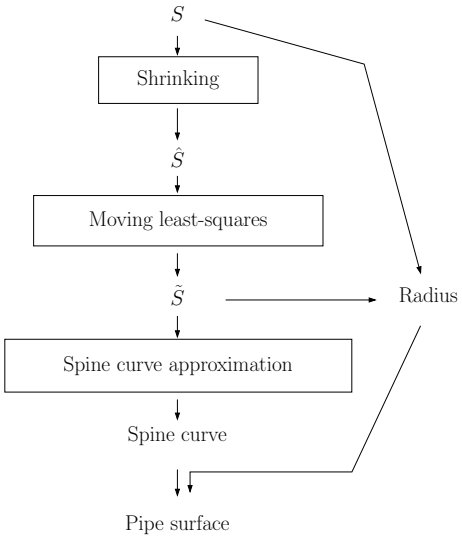

Figure 3. Pipe surface reconstruction algorithm.

neighboring points of $\mathbf{P}_i$ defined by

$$T_i = \{\mathbf{P}_j \mid \|\mathbf{P}_i - \mathbf{P}_j\| < R,\ \mathbf{P}_j \in S\},$$

where R is a prescribed small constant. With an appropriate small R, a point $\mathbf{P}_i$ in S probably has a neighboring point set T_i such that

$$\langle \mathbf{N}_j, \mathbf{N}_i \rangle > 0, \text{ for all } \mathbf{P}_j \in T_i, \tag{1}$$

where $\langle ., . \rangle$ denotes the scalar product of two vectors.

Consider a situation where S shrinks to a spine curve gradually by translating each point $\mathbf{P}_i$ of S along the $\mathbf{N}_i$ direction. R is fixed and the neighboring set T_i of $\mathbf{P}_i$ is recomputed at each stage of the shrinking.

Then, more and more T_is including the points of (almost) the opposite normal directions, that break the condition in Equation (1) will appear by stages.

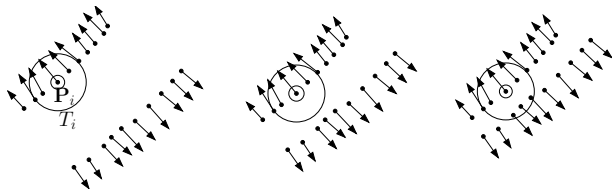

Figure 4. The basic idea of the shrinking method.

The shrinking stops when a large enough portion of the point set breaks the condition in Equation (1). Figure 4 shows the basic idea of shrinking: as the point set shrinks,

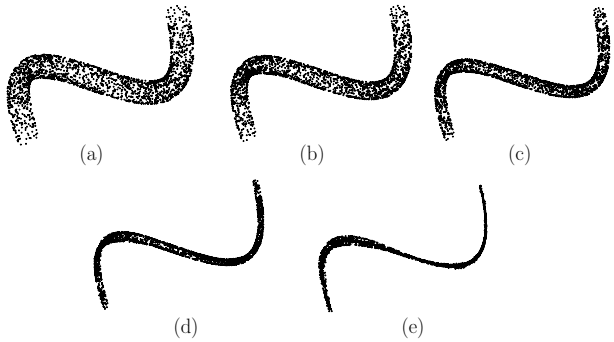

Figure 5. Shrinking S of 2000 points: (a) 0, (b) 0, (c) 63, (d) 672, and (e) 1631 T_i break the condition of Equation (1).

Algorithm 1: Shrinking for pipe surface

1) for $i = 1$ to n do $\hat{\mathbf{P}}_i \Leftarrow \mathbf{P}_i$;
2) BreakCount $\Leftarrow$ 0;
3) for $i = 1$ to n do
    for $j = 1$ to n do
        if $\|\hat{\mathbf{P}}_j - \hat{\mathbf{P}}_i\| < R$ and $\langle \mathbf{N}_j, \mathbf{N}_i \rangle < 0$ then
            BreakCount $\Leftarrow$ BreakCount + 1;
4) if BreakCount $\geq \mu \cdot n$ then
    stop and return $\hat{S} = \{\hat{\mathbf{P}}_i \mid i = 1, ..., n\}$;
5) for $i = 1$ to n do $\hat{\mathbf{P}}_i \Leftarrow \hat{\mathbf{P}}_i - \delta \cdot \mathbf{N}_i$;
6) Goto step 3);

Figure 6. Shrinking algorithm for pipe surface (Algorithm 1)

each T_i may include the points having opposite normal directions. In Figure 5(a) we have 2000 input points and estimated normals. The number of T_is that include the points having an opposite normal direction is increasing gradually from 0 to 1631 as the input point set shrinks.

The shrinking algorithm is summarized in Algorithm 1 (see Figure 6). S shrinks to a point set

$$\hat{S} = \{\hat{\mathbf{P}}_i \mid i = 1, ..., n\},$$

whose shape represents a spine curve of a pipe surface. In Algorithm 1, δ denotes a prescribed shrinking step that can be set as a small constant. μ is a real constant between 0 to 1 denoting a tolerance of the portion of the point set that breaks the condition in Equation (1). We used $\mu \geq 0.7$ for most of examples shown in this paper. Thus, R is the only sensitive parameter that must be given by users interactively.

Grouping

In Algorithm 1, for a point $\hat{\mathbf{P}}_i$, we checked all points $\hat{\mathbf{P}}_j$ in S to see if $\hat{\mathbf{P}}_j$ satisfies with the condition:

$$\|\hat{\mathbf{P}}_j - \hat{\mathbf{P}}_i\| < R \text{ and } \langle \mathbf{N}_j, \mathbf{N}_i \rangle < 0.$$

For the point $\hat{\mathbf{P}}_i$, if we know which points in S have opposite normal directions to $\mathbf{N}_i$ in advance, we may improve the time efficiency of shrinking algorithm.

We subdivide the given point set S into subsets $S_{k,l}$, ($-a \leq k < a$ and $-b \leq l < b$), to group the points with neighboring normal vectors. Subdivision procedure uses Gauss map, where Gauss map is a mapping of unit normal vectors onto a unit sphere. The unit sphere is subdivided into a set of subregions $G_{k,l}$ ($-a \leq k < a$ and $-b \leq l < b$) along its latitude and longitude. Then, we may hash each point in S according to the affiliation of its normal vector in the subregion of unit sphere; i.e., if a normal vector $\mathbf{N}_i$ is mapped onto the unit sphere subregion $G_{k,l}$, we add point $\hat{\mathbf{P}}_i$ to the group $S_{k,l}$.

Now, for a point $\hat{\mathbf{P}}_i$, we do not have to check all other points in the set S to find one with opposite normal direction to $\hat{\mathbf{P}}_i$. When $\hat{\mathbf{P}}_i$ is in the group $S_{k,l}$, we only check the points in the group $S_{-k,-l}$.

2.2. Moving least-squares

$\hat{S}$, the output of the shrinking algorithm, is now processed by a method called *moving least-squares*. The moving least-squares method was developed by McLain [16, 17] and used to transform a point cloud into a very thin curve-like shape [11, 15]. In this subsection, we briefly introduce the moving least-squares method described in [11].

For each data point, a simple curve or surface that fits some neighborhood of the data point is computed using a weighted regression scheme. Then, the data point is moved to a new position on this approximated curve or surface. The moving least-squares method is near-best in the sense that the local error is bounded with the error of a local best polynomial approximation. Once a point set is reduced to a very thin curve-like shape, ordering the point set and computing a smooth approximation curve that fits the point set are not difficult.

Let $S = \{\mathbf{P}_i = (x_i, y_i) \mid i = 1, ..., n\}$ be a point set in 2D. For a point $\mathbf{P}_* \in S$, a local regression line, $\mathbf{L}_* : y = ax + b$, can be computed by minimizing a quadratic function:

$$D_l = \sum_{i=1}^{n} (ax_i + b - y_i)^2 w_i, \tag{2}$$

where w_i is a nonnegative weight for each point $\mathbf{P}_i$ computed by an appropriate weighting function. We can choose any weighting function which generates larger penalties for the points far from $\mathbf{P}_*$. One of our choices is

$$w_i = \begin{cases} 2\frac{r^3}{H^3} - 3\frac{r^2}{H^2} + 1 & \text{if } r < H \\ 0 & \text{if } r \geq H, \end{cases} \tag{3}$$

where $r = \|\mathbf{P}_i - \mathbf{P}_*\|$ and H denotes a prescribed weighting radius [18]. The weighting function in Equation (3) forces the weights of the points that are outside of the open circle of radius H with the center $\mathbf{P}_*$ to vanish. Thus, we can compute a regression line or quadratic curve using only the set of points whose distances from $\mathbf{P}_*$ are less than H.

From this weighted regression, we can compute the local best regression line $\mathbf{L}_*$ for $\mathbf{P}_*$. Consider a transformation M that transforms a whole point set into a new coordinate system, where the x axis is parallel to line $\mathbf{L}_*$, and $\mathbf{P}_*$ is a new origin. Let $\bar{S} = \{\bar{\mathbf{P}}_i = (\bar{x}_i, \bar{y}_i) \mid i = 1, ..., n\}$ be the transformed point set. The local quadratic regression curve

$$\mathbf{Q}_* : \bar{y} = a\bar{x}^2 + b\bar{x} + c \tag{4}$$

for $\bar{\mathbf{P}}_*$ can be computed by minimizing

$$D_q = \sum_{i=1}^{n} (a\bar{x}_i^2 + b\bar{x}_i + c - \bar{y}_i)^2 w_i. \tag{5}$$

Note that the projection of $\bar{\mathbf{P}}_*$ onto $\mathbf{Q}_*$ is $(0, c)$. Finally, $\mathbf{P}_*$ is moved to a new position computed by the inverse-transformation, M^{-1}, of $(0, c)$. Figure 7 shows an example of thining a 2D point set.

Figure 7. Thining a 2D point set using the moving least-squares method: original point set (left), and thin point set after the moving least-squares method (right).

The moving least-squares technique cannot be directly extended into 3D. Although we can easily compute a 3D regression line, computing the 3D quadratic regression curve is difficult. We suggest a local regression algorithm for each point $\mathbf{P}_*$ in 3D as follows:

1. Let $\mathcal{A}$ be a set of neighboring points of $\mathbf{P}_*$, i.e., $\mathcal{A} = \{\mathbf{P}_j \mid \|\mathbf{P}_j - \mathbf{P}_*\| < H\}$.

2. Compute a regression plane $\mathbf{K}$: $z = Ax + By + C$ by minimizing the quadratic function

$$D_k = \sum_{\mathbf{P}_j \in \mathcal{A}} (Ax_j + By_j + C - z_j)^2 w_j.$$

3. Project the points in $\mathcal{A}$ onto $\mathbf{K}$.

4. Solve the 2D moving least-squares problem on $\mathbf{K}$.

The above algorithm is based on the fact that a point on a regular space curve locally has an osculating plane. Basically, the above algorithm can be extended to any higher dimension by the projection of the data points in d dimensional space onto a $d-1$ dimensional hyperplane repeatedly until the problem is reduced to a 2D problem.

Lee [11] suggested some other techniques such as using EMST (Euclidean minimum spanning tree) for robust local regression and using correlation to estimate an appropriate H value for each data point. See [11] for more details.

By using the moving least-squares technique, we can make a very thin point cloud $\tilde{S}$ from $\hat{S}$ – the output of shrinking algorithm. Now, we have to order the points in the very thin point cloud $\tilde{S}$. The simplest ordering method is to connect two closest points each other recursively. After ordering the points, we approximate $\tilde{S}$ with a smooth spine curve using conventional curve approximation/interpolation techniques [13]. Figure 8 shows an example of $\hat{S}$, $\tilde{S}$ and an approximated spine curve.

2.3. Radius of a swept sphere

Let $\tilde{\mathbf{P}}_i$ and $\mathbf{P}_i$ be points in $\tilde{S}$ and S, respectively, with the same index i. $\tilde{S}$ is a discrete version of a spine curve; thus, we can compute the radius r of the swept sphere of a pipe surface by averaging the distance between $\tilde{\mathbf{P}}_i$ and $\mathbf{P}_i$:

$$r = \frac{\sum_{i=1}^{n} \left\| \tilde{\mathbf{P}}_i - \mathbf{P}_i \right\|}{n}. \qquad (6)$$

2.4. Example

Figure 9 shows an example of pipe surface reconstruction. 2000 points are sampled from an original pipe surface with some perturbation. The intermediate result $\hat{S}$ and $\tilde{S}$ of this example have already been shown in Figures 5 and 8.

3. Canal Surface Reconstruction

Canal surface is a generalization of a pipe surface. While a pipe surface has a constant radius r, a variable radius function $r(t)$ can be used for a canal surface. One of the parametrizations of a canal surface $\mathbf{A}(s,t)$ is given by

$$\mathbf{A}(s,t) = \mathbf{C}(t) + r(t)\mathbf{N}_A(s,t), \;\; s \in I_1, \; t \in I_2,$$

Figure 8. $\hat{S}$ (left), $\tilde{S}$ (middle) and an approximated spine curve (right).

Figure 9. Example of pipe surface reconstruction: an original pipe surface (left), 2000 sample points (middle), and a reconstructed pipe surface (right).

where $\mathbf{C}(t)$, $r(t)$, and $\mathbf{N}_A(s,t)$ are a spine curve, a radius function, and a unit normal vector of the canal surface, respectively. Peternell and Pottmann [2] showed that the canal surface has a rational parametrization when both $\mathbf{C}(t)$ and $r(t)$ are rational. Figure 10 shows that a canal surface is an envelope of spheres of variable radii moving along a spine curve.

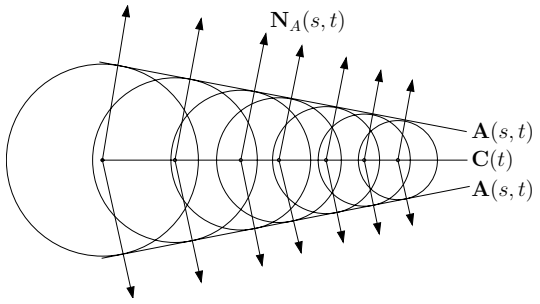

Figure 10. Symmetric normal vectors of a canal surface

Like the pipe surface case, we assume that an input point set represents a real and regular canal surface [2] without any self intersection. For a canal surface, we must consider the following two facts in the shrinking algorithm.

- Due to the variable radius, two symmetric normal vectors of a canal surface may not have exact opposite directions (see Figure 10). We must define a more strict condition than Equation (1) suggested for pipe surfaces. In general, the angle between two symmetric normal vectors can be in the interval $(0 : \pi]$. However, we assume that for a real object, the rate of the change

of radius $r'(t)$ is not too large.

- Unlike the pipe surface case, each data point must shrink according to a different radius from the others. Some points stop shrinking in the middle of the algorithm execution, while others continue to shrink until the points are satisfied with the stopping condition.

In Algorithm 2 (see Figure 11), we add an additional parameter τ to control the tolerance of direction deviation between two symmetric normal vectors. If $\tau = 0$ the condition is the same as that for the pipe surface case. We used $\tau = 0.1736 \simeq \cos(80°)$ for the examples in this paper assuming that $r'(t)$ is not too large. Furthermore, a tag, stop_i, for each $\mathbf{P}_i$ is used to record the information of whether $\mathbf{P}_i$ has stopped shrinking or not. Only non-stopped points continue to shrink towards a spine curve. Figure 12 shows an example of shrinking in canal surface reconstruction.

Algorithm 2: Shrinking for canal surface

```
1) for i = 1 to n do
     P̂_i ⇐ P_i;
     stop_i ⇐ false;
2) BreakCount ⇐ 0;
3) for i = 1 to n do
     for j = 1 to n do
         if stop_i = false and ‖P̂_j − P̂_i‖ < R
           and ⟨N_j, N_i⟩ < τ then
             BreakCount ⇐ BreakCount + 1;
             stop_i ⇐ true;
4) if BreakCount ≥ μ · n then
     stop and return Ŝ = {P̂_i | i = 1, ..., n};
5) for i = 1 to n do
     if stop_i = false
         P̂_i ⇐ P̂_i − δ · N_i;
6) Goto step 3);
```

Figure 11. Shrinking algorithm for canal surface (Algorithm 2)

For a canal surface, we also apply the moving least-squares technique to $\hat{S}$ to generate a very thin point set $\tilde{S}$ like the pipe surface reconstruction. Figure 13 shows $\hat{S}$, $\tilde{S}$ and a spine curve approximated from $\tilde{S}$.

The radius function $r(t)$ of a canal surface can be approximated from the discrete radii

$$r_i = \left\| \tilde{\mathbf{P}}_i - \mathbf{P}_i \right\|$$

using a conventional curve approximation technique that is also used in spine curve approximation. Figure 14 shows an example of a reconstructed canal surface. 2000 input points are sampled from an exact canal surface with some perturbation. The shrinking and moving least-squares steps

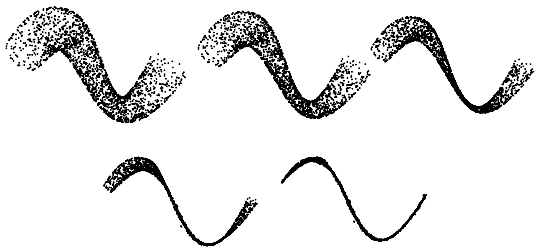

Figure 12. Shrinking 2000 data points for reconstruction of a canal surface: 0, 0, 8, 998, and 1609 points have stopped by stages (from left to right).

of this example have already been shown in Figures 12 and 13.

Figure 13. $\hat{S}$ (left), $\tilde{S}$ (middle), and an approximated spine curve (right).

Figure 14. Example of canal surface reconstruction: an original canal surface (left), 2000 sample points (middle), and a reconstructed canal surface (right).

4. Experimental Results

The original pipe surface $\mathbf{B}(s,t)$ of the example shown in Figure 9 is generated with a radius of 1, and a cubic B-spline spine curve with a knot vector (0, 0, 0, 0, 1, 2, 3, 3, 3, 3) and the six control points shown in Table 1. We take 2000 random sample points from $\mathbf{B}(s,t) + \epsilon(s,t)$, where $\epsilon(s,t)$ is a perturbation surface that has any random value between -0.1 and 0.1 for each sample point. For the shrinking algorithm, we used $\delta = 0.2$, $\mu = 0.7$, and $R = 0.2$ (see Algorithm 1). As shown in Figure 5, it took five iteration steps to generate $\hat{S}$, where 1631 points break the condition in Equation (1). The radius computed by Equation (6) is 0.984172. The maximal and average distance between the original spine curve $\mathbf{C}(t)$ and the approximated spine curve

$\mathbf{C}_a(t)$ are 1.94653324 and 1.0200849, respectively, where distance is defined by $\|\mathbf{C}(t) - \mathbf{C}_a(t)\|$.

x	y	z
-3.000	6.660	0.000
-0.337	5.660	-4.660
0.410	0.430	-8.052
-2.037	-7.906	-3.354
1.813	-6.807	-0.175
6.760	-3.150	0.000

Table 1. Control points of the spine curve of the original pipe surface in the example shown in Figure 9.

x	y	z
0.000	0.000	0.000
2.773	-0.234	0.159
3.091	0.058	-0.043
3.086	2.917	-0.365
2.939	3.054	0.247
0.066	3.159	-0.292
-0.279	2.959	0.377
0.411	3.108	2.569
-0.045	2.942	2.928
0.000	0.000	3.000

Table 2. Control points of the spine curve of the original pipe surface in the example shown in Figure 15.

x	y	z
-4.000	4.000	0.000
-0.478	4.547	0.000
3.335	2.618	0.000
1.428	-1.214	0.000
-3.543	-1.614	0.000
-4.257	-4.700	0.000

Table 3. Control points of the spine curve of the original canal surface in the example shown in Figure 16.

Figure 15 illustrates another example of pipe surface reconstruction. The original pipe surface $\mathbf{B}(s,t)$ has a radius of 3, and a cubic B-spline spine curve with a knot vector (0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 7, 7, 7, 7) and the 10 control points shown in Table 2. 2000 random sample points are taken from $\mathbf{B}(s,t) + \epsilon(s,t)$ such that $-0.05 \leq \epsilon(s,t) \leq 0.05$. For the shrinking algorithm, we used $\delta = 0.05$, $\mu = 0.95$, and $R = 0.1$. It took six iteration steps to generate $\hat{S}$, where 1934 points break the condition in Equation (1). The radius computed by Equation (6) is 0.298778. The maximal and average distance between the original spine curve and the approximated spine curve are 1.60093152 and 0.814391259, respectively.

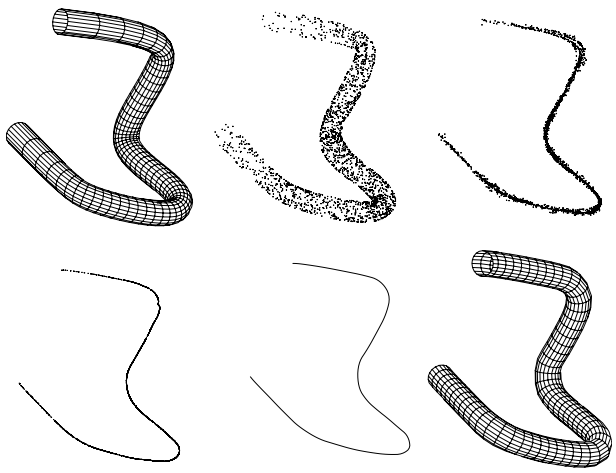

Figure 15. Another example of pipe surface reconstruction: original surface, S (2000 sample points), $\hat{S}$, $\tilde{S}$, approximated spine curve, and reconstructed pipe surface (from top–left to bottom-right).

The original canal surface $\mathbf{A}(s,t)$ for the example in Figure 14 has the same spine curve described in Table 1. The radius function $r(t)$ is a one-dimensional quadratic B-spline curve with a knot vector (0, 0, 0, 1, 2, 3, 4, 4, 4) and six control points (2, 1.5, 1, 1.5, 2.5, 2). 2000 sample points are taken from $\mathbf{A}(s,t) + \epsilon(s,t)$ such that $-0.05 \leq \epsilon(s,t) \leq 0.05$. The parameters of the shrinking algorithm are $\delta = 0.1$, $\mu = 0.8$, $R = 0.1$, and $\tau = \cos(80^\circ)$. $\hat{S}$ is computed after 20 iteration steps of shrinking, where 1609 points finally stopped shrinking. The maximal and average distance between the original spine curve and the approximated spine curve are 1.772888 and 1.0936823, respectively. The maximal and average distance between the original and the approximated radius functions are 0.14423506 and 0.07538086, respectively.

Figure 16 illustrates another example of canal surface reconstruction. The original canal surface $\mathbf{A}(s,t)$ has a cubic B-spline spine curve with a knot vector (0, 0, 0, 0, 1, 2, 3, 3, 3, 3) and the six control points shown in Table 3. The radius function is a one-dimensional cubic B-spline curve with a knot vector (0, 0, 0, 0, 1, 2, 3, 3, 3, 3) and six control points, (1.9999, 1.6666, 1.103463, 1.037336, 1.833333, 2.5). 2000 sample points are taken from $\mathbf{A}(s,t)+\epsilon(s,t)$ such that $-0.05 \leq \epsilon(s,t) \leq 0.05$. The parameters of the shrinking algorithm are $\delta = 0.1$, $\mu = 1.0$, $R = 0.1$, and $\tau = \cos(80^\circ)$. $\hat{S}$ is computed after 26 itera-

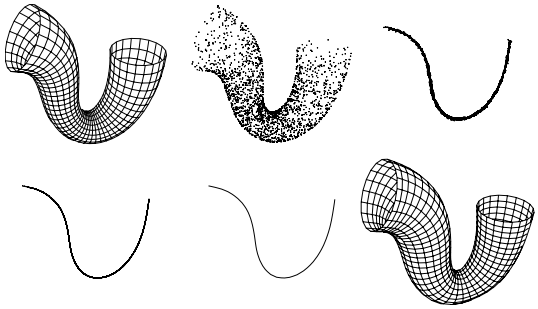

Figure 16. Another example of canal surface reconstruction: original surface, S (2000 sample points), $\hat{S}$, $\tilde{S}$, approximated spine curve, and reconstructed canal surface (from top-left to bottom-right).

tion steps of shrinking, at which 1995 points have stopped shrinking. The maximal and average distance between the original spine curve and the approximated spine curve are 1.4648859 and 1.023708, respectively. The maximal and average distance between the original and the approximated radius functions are 0.5901724 and 0.2694635, respectively.

5. Conclusion

In this paper, we have suggested methods to reconstruct a pipe and a canal surface from an unorganized point set. The shrinking method and moving least-squares methods were used effectively to extract a spine curve and a constant radius (or a radius function) from a given point set. We expect that this approach can be extended to more complex problems. Currently, the sweep surface reconstruction problem is under investigation.

References

[1] T. Maekawa, N. M. Patrikalakis, T. Sakkalis, and G. Yu. Analysis and applications of pipe surfaces. *Computer Aided Geometric Design*, 15(5):437–458, 1998.

[2] M. Peternell and H. Pottmann. Computing rational parametrizations of canal surfaces. *Journal of Symbolic Computation*, 23:255–266, 1997.

[3] H. Pottmann and T. Randrup. Rotational and helical surface approximation for reverse engineering. *Computing*, 60:307–322, 1998.

[4] H. Pottmann, H.-Y. Chen, and I.-K. Lee. Approximation by profile surfaces. in A. Ball et al. Eds., *The Mathematics of Surfaces VIII*, Information Geometers, pages 17–36, 1998.

[5] H.-Y. Chen, I.-K. Lee, S. Leopoldseder, H. Pottmann, T. Randrup, and J. Wallner. On surface approximation using developable surfaces. *Graphical Models and Image Processing*, 61:110–124, 1999.

[6] M. Peternell and H. Pottmann. Approximation in the space of planes: applications to geometric modeling and reverse engineering. *Rev.R.Acad.Cien.Serie A.Mat*, 96(2):243–256, 2002.

[7] H. Pottmann and J. Wallner. *Computational line geometry*. Springer-Verlag, 2001.

[8] H. Pottmann, S. Leopoldseder, J. Wallner, and M. Peternell. Recognition and reconstruction of special surfaces from point clouds. *Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. XXXIV, Part 3A, Commisiion III, pp. 271-276, 2002.

[9] W.-D. Ueng, J.-Y. Lai, and J.-L. Doong. Sweep-surface reconstruction from three-dimensional measured data. *Computer-Aided Design*, 30(10):791–805, 1998.

[10] R. Ramamoorthi and J. Arvo. Creating generative models from range images. *SIGGRAPH '99 proceedings*, 1999.

[11] I.-K. Lee. Curve reconstruction from unorganized points. *Computer Aided Geometric Design*, 17(2):161–177, 2000.

[12] I.-K. Lee, J. Wallner, and H. Pottmann. Scattered data approximation with kinematic surfaces. *Proceedings of SAMPTA'99*, Loen, Norway, pp. 72–77, 1999.

[13] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*, A. K. Peters, 1993.

[14] P. Krsek, G. Lukás, and R. R. Martin. Algorithms for computing curvatures from range data. in A. Ball et al. Eds., *The Mathematics of Surfaces VIII*, Information Geometers, pages 1–16, 1998.

[15] D. Levin, Mesh-independent surface interpolation, *private communication*.

[16] D. McLain, Drawing contours from arbitrary data points, *The Computer Journal*, 17:318–324, 1974.

[17] D. McLain, Two dimensional interpolation from random data, *The Computer Journal*, 19:178–181, 1976.

[18] G. Wyvill, C. McPheeters, and B. Wyvill, Data structure for soft objects, *Visual Computer* 2:227–234, 1986.