Computer Aided Geometric Design 17 (2000) 161–177

COMPUTER AIDED GEOMETRIC DESIGN

www.elsevier.com/locate/comaid

Curve reconstruction from unorganized points

In-Kwon Lee [1]

POSTECH Information Research Laboratories, San 31 Hyoja Dong, Pohang 790-784, South Korea

Received September 1998; revised June 1999

Abstract

We present an algorithm to approximate a set of unorganized points with a simple curve without self-intersections. The moving least-squares method has a good ability to reduce a point cloud to a thin curve-like shape which is a near-best approximation of the point set. In this paper, an improved moving least-squares technique is suggested using Euclidean minimum spanning tree, region expansion and refining iteration. After thinning a given point cloud using the improved moving least-squares technique we can easily reconstruct a smooth curve. As an application, a pipe surface reconstruction algorithm is presented. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Curve reconstruction; Reverse engineering; Moving least-squares; Unorganized points; Pipe surface

1. Introduction

Reconstructing a curve or a surface from a point set is one of the most important problems in the reverse engineering of geometric models. In some cases curve reconstruction plays an important role in the surface reconstruction problem (Chen et al., 1999; Pottmann and Randrup, 1998; Pottmann et al., 1998a, 1998b). In this paper, we focus on the reconstruction of a curve from an *unorganized* point cloud having no ordering of the point elements. An attractive feature of our solution is that the algorithm can be extended to any dimension.

The motivation of this work comes from recent research attempting to reconstruct surfaces of revolution, helical surfaces, spiral surfaces, profile surfaces and developable surfaces from a set of points (Chen et al., 1999; Pottmann and Randrup, 1998; Pottmann et al., 1998a, 1998b). In that research, principal motion parameters, such as motion axis and pitch, are computed using line geometry. Using motion trajectories, the data points are

[1] E-mail: iklee@postech.ac.kr.

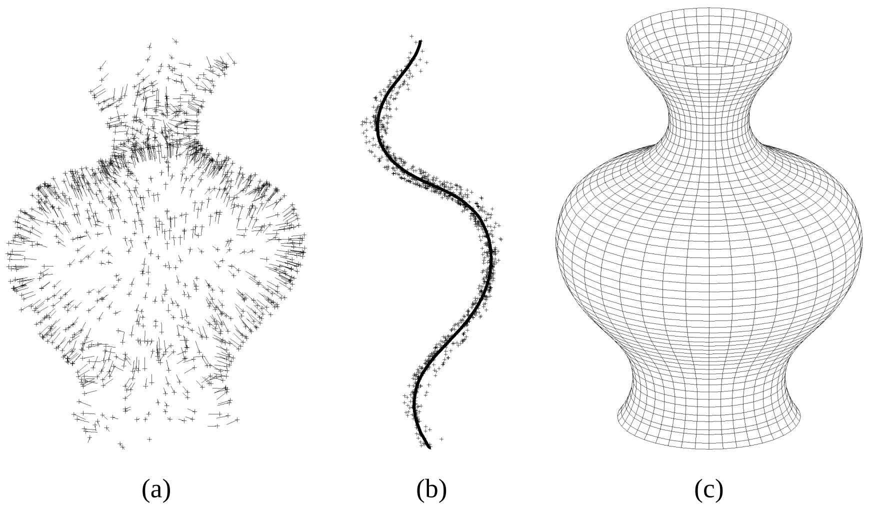


Fig. 1. Reconstruction of a surface of revolution: (a) data points and estimated normal vectors, (b) projected points and approximating curve, and (c) reconstructed surface of revolution.

projected onto an appropriate plane and approximated with a curve which is the profile curve of the reconstructed surface. Fig. 1 shows an example of the reconstruction of a surface of revolution. After a rotation axis of the surface of revolution is computed, a profile curve can be constructed by

(i) rotating all points with respect to the rotation axis into a plane through the rotation axis, and
(ii) approximating the projected point set with a curve.

Pottmann and Randrup (1998) used a method based on *image thinning* to approximate a profile curve. After defining an appropriate grid on the plane, pixels including one or more points are filled with black creating a binary image. Using conventional image thinning algorithms, the medial axis of this binary image can be computed easily. Finally, we can approximate the medial axis with a smooth curve. However, this method has some disadvantages:

- It is not easy to determine the size of pixel of the image. If the pixel is too large, the medial axis may not represent the best approximation curve of the point set. If the pixel is too small, it is possible for the point set to be separated into several different components.
- For open curves, the medial axis does not represent the shapes of the point set near the end points of the curve.

Many approaches have been suggested for the reconstruction of surfaces from unorganized point sets (Amenta et al., 1998; Bajajet et al., 1997; Hoppe et al., 1992, 1993; Mencl, 1995). For curve reconstruction, there are many solutions for cases where the ordering of points is known. However, there has been little research for curve reconstruction from an unorganized point set. Fang et al. (1992) used a method based on spring energy minimization to approximate an unorganized point set with a curve,

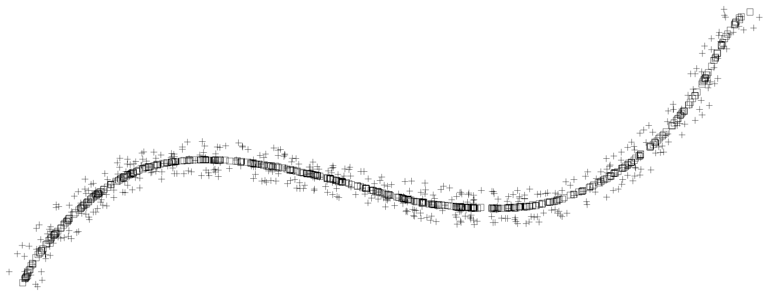


Fig. 2. Thinning a 2D point cloud using moving least-squares: the points represented with boxes are the result of moving least-squares.

which needs a good initial guess of the solution. Dedieu and Favardin (1994) presented an algorithm for ordering unorganized points assuming that all points are on the reconstructed curve; thus, this method is not appropriate for a point cloud. Taubin and Ronfard (1996) reconstructed a planar curve from unorganized data points using an implicit simplicial curve, which is defined by a planar triangular mesh and the values at the vertices of the mesh.

One simple solution of the curve reconstruction problem is finding a polygonal boundary such as an α-shape (Edelsbrunner and Mücke, 1994) which represents the shape of the point set. Then, the medial axis, representing the skeleton of the point set of this polytop is computed. However, the medial axis of a general polytop may have several branches (Ferley et al., 1997) that are difficult to transform into a single curve. The computation of the medial axis of a polytop would be burdened by the numerical problems, especially when the polytop is very complex such as an α-shape in higher dimension. Furthermore, like image thinning, this approach cannot reflect the shape near the end points of open curves.

To solve the curve reconstruction problem, we focus on the problem of making the point cloud as thin as possible with respect to the density and shape of the point set. After the original point set is reduced to a sufficiently thin cloud, ordering the points in this thin cloud is not difficult as we will see later. Levin (1998b) used a method called *moving least-squares* to thin a point cloud. The moving least-squares method was developed by McLain (1974, 1976) and used in many applications such as scattered data interpolation and smoothing (Lancaster, 1979; Levin, 1998a; McLain, 1974, 1976).

The basic idea of moving least-squares method is to compute a simple regression curve/surface $\mathbf{C}_i$ for each data point $\mathbf{P}_i$ which locally fits a certain neighborhood of $\mathbf{P}_i$ using a weighted regression scheme. Then $\mathbf{P}_i$ is moved to a new position $\mathbf{P}'_i$ on $\mathbf{C}_i$. Clearly, moving least-squares is near-best in the sense that the local error is bounded with the error of a local best approximation. This simple idea works well in many cases. For a thin cloud of points, moving least-squares generates very good results (see Fig. 2). However, there are some serious difficulties associated with the moving least-squares technique for some cases, such as varying thickness of the point set and the effects from unwanted neighboring points (see Fig. 3). In this paper, we suggest an improved version of moving least-squares to overcome these difficulties. Experimental results show that great improvements are

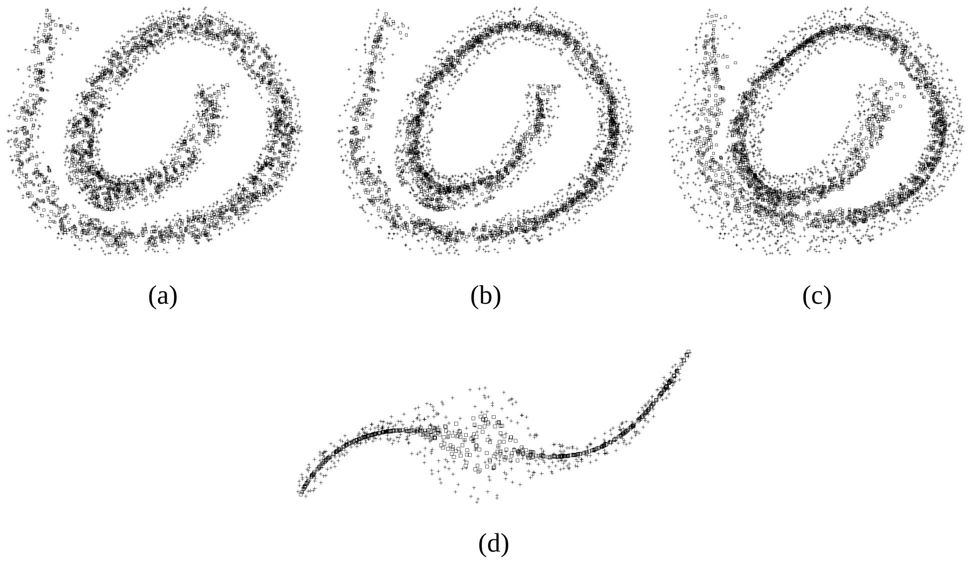


Fig. 3. Difficulties associated with moving least-squares.

achieved by our simple modification of moving least-squares. In this paper, we assume that the point cloud may represent a simple smooth curve without self-intersections.

This paper is organized as follows. In Section 2 we introduce the moving least-squares technique and describe some difficulties associated with this method. Section 3 presents an improved version of the moving least-squares, which overcomes the difficulties mentioned in Section 2. The extension to 3D or higher dimension is presented in Section 4. In Section 5, an algorithm is described to compute an approximation curve of a thin point cloud which is generated by moving least-squares. In Section 6, we present an interesting application of the curve approximation—the reconstruction of pipe surfaces. Finally, in Section 7 we conclude this work and suggest some future research directions.

2. Background

The concept of moving least-squares can be used to thin a point cloud (Levin, 1998b). 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 (see (Levin, 1998a, 1998b) for detailed error analyses of moving least-squares).

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

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

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

$$w_i = \mathrm{e}^{-r^2/H^2}, \tag{2}$$

where $r = \|\mathbf{P}_i - \mathbf{P}_*\|^2$, and H is a prescribed real constant. 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 the line $\mathbf{L}_*$, and $\mathbf{P}_*$ is a new origin. Let $\hat{\mathcal{S}} = \{\hat{\mathbf{P}}_i = (\hat{x}_i, \hat{y}_i) \mid i = 1, \ldots, N\}$ be the transformed point set. The local quadratic regression curve

$$\mathbf{Q}_*:\ \hat{y} = a\hat{x}^2 + b\hat{x} + c \tag{3}$$

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

$$D_q = \sum_{i=1}^{N} \left(a\hat{x}_i^2 + b\hat{x}_i + c - \hat{y}_i\right)^2 w_i. \tag{4}$$

Note that the projection of $\hat{\mathbf{P}}_*$ onto $\mathbf{Q}_*$ is $(0, c)$. Finally, $\mathbf{P}_*$ is moved to a new position computed by the inverse-transformation, M^{-1}, of $(0, c)$.

Instead of using the whole point set for each local regression, we can restrict the neighborhood of each point by introducing a cubic function (Wyvill et al., 1986):

$$w_i = \begin{cases} 2\dfrac{r^3}{H^3} - 3\dfrac{r^2}{H^2} + 1 & \text{if } r < H, \\ 0 & \text{if } r \geqslant H, \end{cases} \tag{5}$$

where $r = \|\mathbf{P}_i - \mathbf{P}_*\|^2$. The weighting function in Eq. (5) 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.

This simple method works well in many cases to reduce a point set to a thin cloud, as we saw in Fig. 2. However, the pure moving least-squares method does not work well in some cases:

- If H is too small, the local regressions do not reflect the thickness of the point cloud. Thus, the resulting points are scattered (Fig. 3(a)).
- Increasing H may help to make the thin point cloud. However, with a large H, the local regression may include some unwanted points, which may cause the algorithm to fail (Fig. 3(b)(c)).
- A fixed H cannot be applied to the whole point cloud that has a varying thickness (Fig. 3(d)).

3. Improved moving least-squares

To prevent the effects from unwanted points in the local regressions, we need to make a certain structure (as simple as possible) for the point set to define the connectivity of the point elements. The *Euclidean Minimum Spanning Tree* (EMST) is a good candidate. Let $\mathcal{S} = \{\mathbf{P}_i = (x_i, y_i) \mid i = 1, \ldots, N\}$ be a point set. Consider a graph $\mathcal{G} = (\mathcal{S}, \mathcal{E})$ having total connectivity among all point elements, i.e., $\mathcal{E} = \{(\mathbf{P}_i, \mathbf{P}_j) \mid i, j = 1, \ldots, N,\ i \neq j\}$. The EMST of $\mathcal{G}$ is a tree (thus, having no cycle) connecting all points in $\mathcal{S}$ so that the sum of its edge lengths is minimum. EMST of $\mathcal{G}$ can be computed by the well-known Kruskal's or Prim's algorithm (Sedgewick, 1988).

Instead of using a total connectivity graph, we can also use *Delaunay triangulation* (DT) to compute EMST due to the well-known fact that EMST is a subgraph of DT (Preparata and Shomos, 1985). Fig. 4 shows an example of DT and EMST of a point set in 2D. Note that the α-shape of Edelsbrunner and Mücke (1994) is another subset of DT which can represent the appropriate connectivity of the point elements though the parameter α is usually chosen interactively (Amenta et al., 1998; Guo et al., 1997).

Algorithm 1.

```
Collect1($\mathbf{P}$, $\mathbf{P}_*$, $H$, $\mathcal{A}$)
  begin
    $\mathcal{A} \Leftarrow \mathcal{A} \cup \{\mathbf{P}\}$
    for each edge $(\mathbf{P}, \mathbf{P}_j)$ in EMST do
      if $\mathbf{P}_j \notin \mathcal{A}$ and $\|\mathbf{P}_* - \mathbf{P}_j\| < H$
        then Collect1($\mathbf{P}_j$, $\mathbf{P}_*$, $H$, $\mathcal{A}$)
  end
```

Now, in the local regression for $\mathbf{P}_*$, we use the EMST to collect the neighboring points. Let $\mathcal{A}$ denote a set of neighboring points of $\mathbf{P}_*$ with respect to a distance H. The recursive

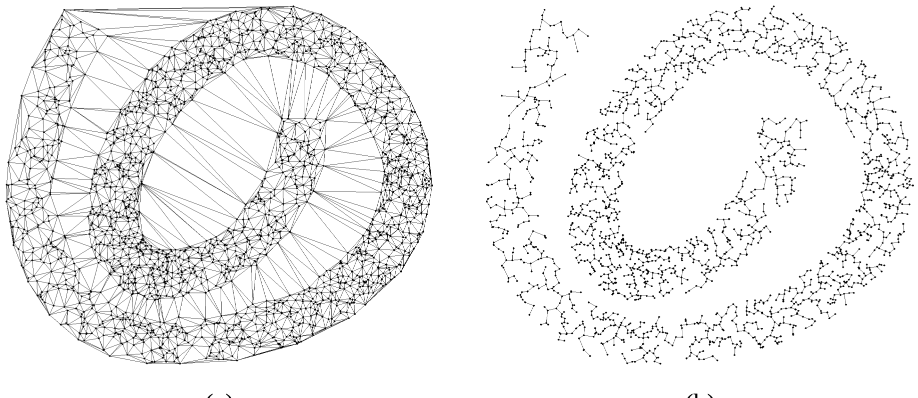

(a) (b)

Fig. 4. An example of (a) Delaunay triangulation and (b) Euclidean minimum spanning tree.

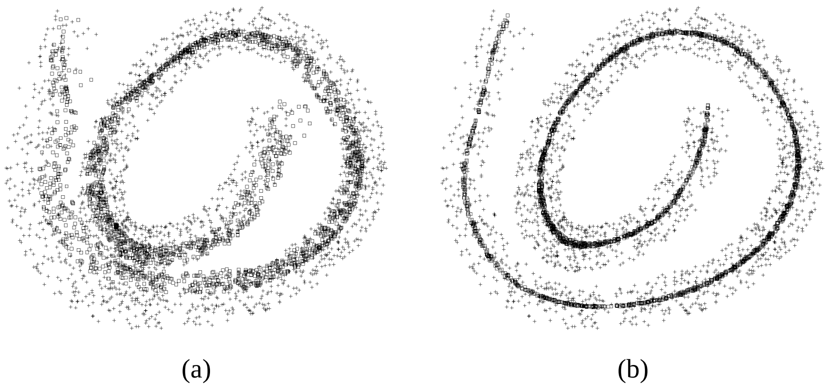


Fig. 5. The result of moving least-squares (a) without EMST and (b) with EMST.

algorithm in Algorithm 1 is used to compute $\mathcal{A}$ by initially assigning an empty set to $\mathcal{A}$ and calling **Collect1**($\mathbf{P}_*$, $\mathbf{P}_*$, H, $\mathcal{A}$). Fig. 5 shows the effect of the collecting algorithm applied to moving least-squares by comparing two results with and without the EMST structure.

The size of H must be determined to reflect the thickness of the point cloud. We introduce the concept of *correlation* (Pitman, 1992) developed in probability theory, which can be used to compute the size of an appropriate H. Let X and Y be two random variables. The *covariance* of X and Y is defined by

$$Cov(X,Y) = E\big[(X-E(X))(Y-E(Y))\big] = E(XY) - E(X)E(Y), \tag{6}$$

where $E[\circ]$ denotes an expectation (average) of a random variable $\circ$. Correlation, denoted as $\rho(X,Y)$, is a standardized covariance which is easier to interpret:

$$\rho(X,Y) = \frac{Cov(X,Y)}{SD(X)SD(Y)}, \tag{7}$$

where $SD(\circ)$ represents a standard deviation of a random variable $\circ$. $\rho(X,Y)$ has a value between -1 and $+1$ representing the degree of linear dependence between X and Y. For example, if $Y = X + b$ (b is a constant) exactly, then $\rho(X,Y) = 1$. Conversely, if $Y = -X + b$ exactly, then $\rho(X,Y) = -1$. Fig. 6 shows some correlations $\rho(X,Y)$ of various sets of points.

Algorithm 2.

```
Collect2($\mathbf{P}$, $\mathbf{P}_*$, $H_\circ$, $\rho_\circ$, $\varepsilon_h$, $\mathcal{A}$)
  begin
    $H \Leftarrow H_\circ$;
    repeat
      Collect1($\mathbf{P}$, $\mathbf{P}_*$, $H$, $\mathcal{A}$);
      $L \Leftarrow$ regression line of $\mathcal{A}$;
      $\theta \Leftarrow$ angle between $L$ and positive $x$ axis;
      $\hat{\mathcal{A}} \Leftarrow$ Rotate $\mathcal{A}$ by $\frac{\pi}{4} - \theta$;
```

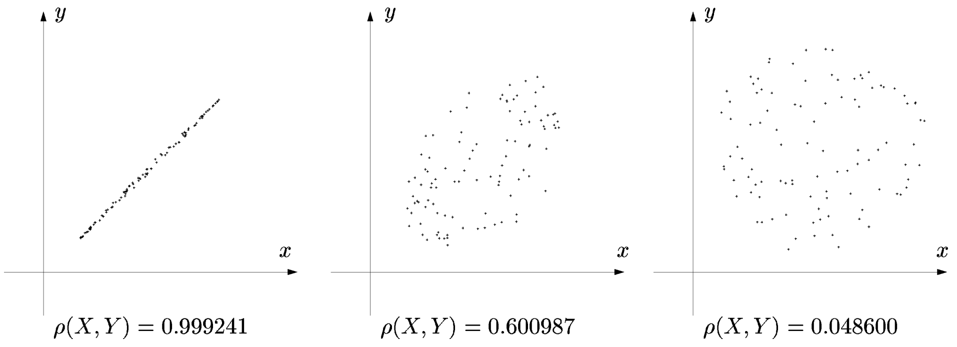


Fig. 6. Correlations of various point sets.

```
        $\rho \Leftarrow$ correlation of data points in $\hat{\mathcal{A}}$;
        $H \Leftarrow H + \varepsilon_h$;
    **until** $\rho < \rho_\circ$
**end**
```

The basic idea for determining the size of a region is to initially take a small H and expand the region until the region has a certain large ρ, which means the region includes enough points for a local regression. The modified version of the collection algorithm for each point $\mathbf{P}_*$ is sketched in Algorithm 2, where $\rho_\circ$ is a prescribed tolerance of correlation, $H_\circ$ is an initial H, and ε_h is an increment of H.

Fig. 7 shows the sequence of the region expansion to find an appropriate H for a local regression of a point. Note that, before the computation of each ρ, the set of points in the current region must be rotated by an angle of $\frac{\pi}{4} - \theta$, where θ is an angle between a local regression line of the current region and a positive x axis. Fig. 8(c) shows the result of Algorithm 2 with $H_\circ = 1$ and $\rho_\circ = 0.7$, which is better than the case using a fixed $H = 1$ (Fig. 8(b)).

However, as we see in Fig. 8(c), the difficulty caused by varying thickness of the point cloud still remains. We suggest the use of an iteration scheme for refining a point set. Let $\mathbf{Q}_*$ be a quadratic regression curve (in Eq. (3)) and $\mathcal{A}$ be a set of neighboring points of a point $\mathbf{P}_*$. During the computation of the local quadratic regression curve, we can easily compute the average approximation error

$$\varepsilon_* = \frac{\sum_{\mathbf{P}_j \in \mathcal{A}} \|\mathbf{P}_j - \mathbf{Q}_*\|}{|\mathcal{A}|}, \tag{8}$$

where $|\mathcal{A}|$ is the number of points in $\mathcal{A}$. If ε_* is larger than the prescribed tolerance, we apply moving least-squares for the point $\mathbf{P}_*$ repeatedly. Fig. 9 shows an example of the iteration.

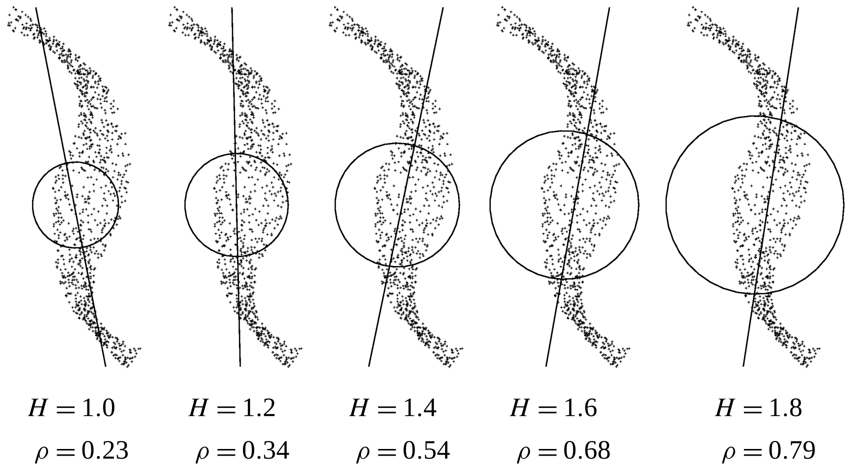


Fig. 7. Expanding a region to find an H using correlation computation: lines represent the local regression lines computed from the sets of collected points.

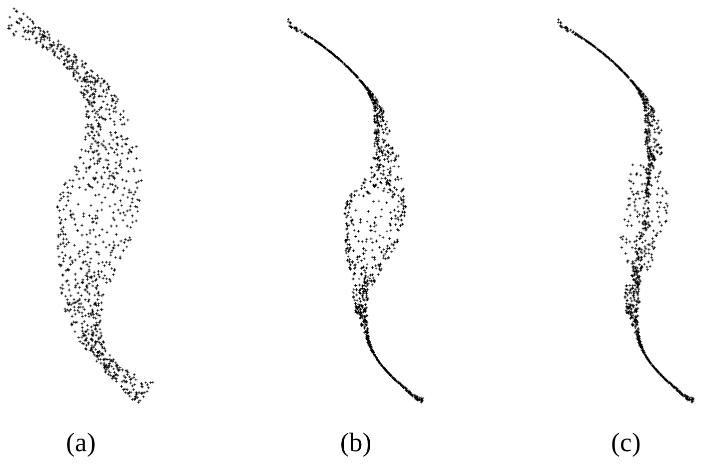


Fig. 8. The effect of using variable H based on correlation computation: (a) an original point set, (b) the output of moving least-squares method with Algorithm 1 using fixed size $H = 1.0$, and (c) the output of moving least-squares method with with Algorithm 2 using $H_\circ = 1.0$ and $\rho_\circ = 0.7$.

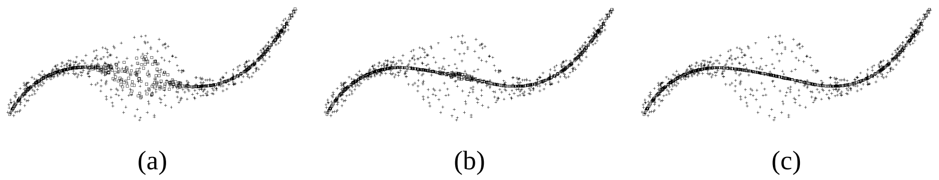


Fig. 9. An example of the iteration scheme for refining a point set ($H = 2$, $\varepsilon_* = 0.1$): (a) first iteration (500 points moved), (b) second iteration (500 points moved), and (c) third iteration (487 points moved).

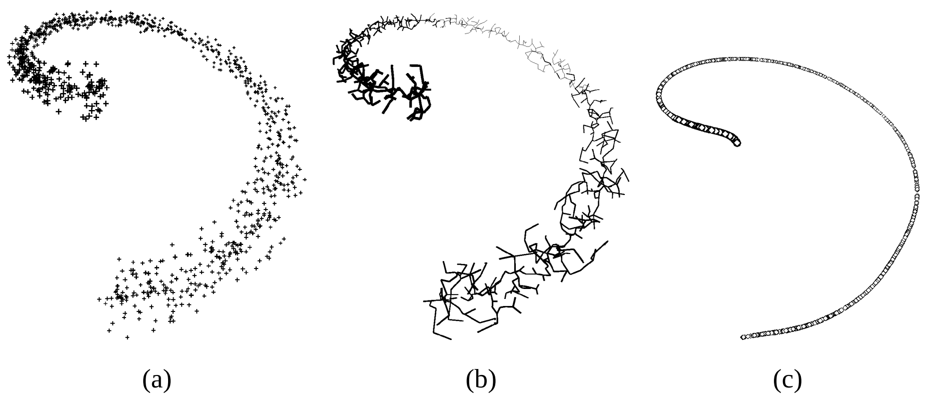


Fig. 10. 3D curve reconstruction: (a) 3D point set, (b) EMST of the point set, and (c) thin point cloud using $H = 2$ after two iterations.

4. 3D extension

The moving least-squares technique described in Section 2 cannot be directly extended into 3D. Although we can easily compute a 3D regression line, computing the 3D quadratic regression curve is not easy. We suggest a local regression algorithm for each point $\mathbf{P}_*$ in 3D as follows:

(1) Collect a set of neighboring points, $\mathcal{A}$, within H using EMST.
(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 the plane $\mathbf{K}$.
(4) If the correlation ρ of $\mathcal{A}$ on $\mathbf{K}$ is less than the prescribed tolerance then increase H and repeat steps (1)–(3).
(5) Solve the 2D moving least-squares problem on the plane $\mathbf{K}$.

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. Note that the weight values computed for 3D data in steps (1) and (2) can be used to solve the 2D problem in step (5) without any extra computations. Fig. 10 shows the moving least-squares method applied to a 3D point set.

5. Approximating a thin point cloud with a curve

Once the moving least-squares technique generates a sufficiently thin point cloud, we can approximate the thin point cloud with a smooth curve by ordering the points in the cloud after reducing the number of points if needed.

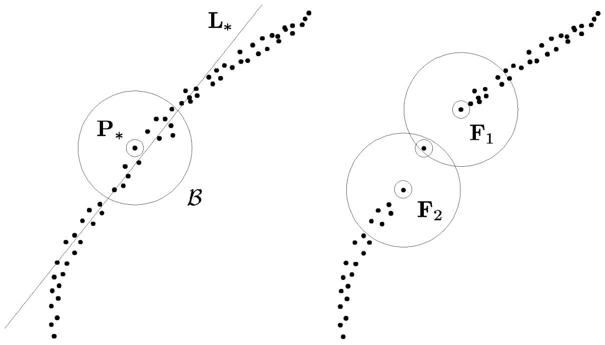


Fig. 11. Ordering points.

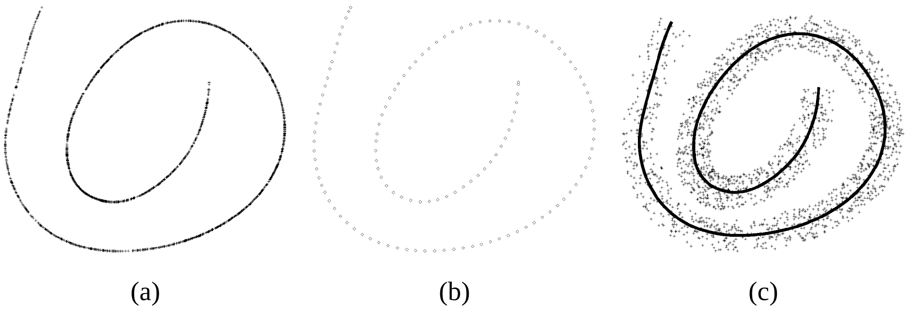


Fig. 12. Approximating a thin point cloud with a smooth curve: (a) thin point cloud with 2000 points, (b) 125 ordered points, and (c) quadratic B-spline curve approximation.

First, we take a random point $\mathbf{P}_*$ from the thin point cloud $\mathcal{S}$. The neighboring points of $\mathbf{P}_*$ are nearly on a line (having a direction $\mathbf{L}_*$) if $\mathcal{S}$ is sufficiently thin. Let h be a prescribed small value and $\mathcal{B}$ be a set of points having distances less than h from $\mathbf{P}_*$. We can easily divide $\mathcal{B}$ into two sets by inspecting the direction of vectors $\vec{v}_j$ which are from $\mathbf{P}_*$ to $\mathbf{P}_j \in \mathcal{B}$:

$$\mathcal{B}_1 = \{\mathbf{P}_j \mid \mathbf{P}_j \in \mathcal{B},\ \langle \vec{v}_j, \mathbf{L}_* \rangle \geqslant 0\},$$
$$\mathcal{B}_2 = \{\mathbf{P}_j \mid \mathbf{P}_j \in \mathcal{B},\ \langle \vec{v}_j, \mathbf{L}_* \rangle < 0\},$$

where $\langle \circ, \circ \rangle$ denotes the scalar product of two vectors. Recall that we already computed the local regression line $\mathbf{L}_*$ for the point $\mathbf{P}_*$ in the previous stage; thus, we can use $\mathbf{L}_*$ without any extra computation. (For a 3D point set, it is necessary to compute a 3D regression line.) We take two points, $\mathbf{F}_1$ and $\mathbf{F}_2$, which are furthest from $\mathbf{P}_*$ in both sets $\mathcal{B}_1$ and $\mathcal{B}_2$, respectively. From the two points $\mathbf{F}_1$ and $\mathbf{F}_2$, the step is repeated in both directions until we cannot continue. Fig. 11 shows the ordering process of the thin point cloud.

After ordering the points, we can apply conventional curve approximation methods (Hoschek and Lasser, 1993) to the ordered points. Fig. 12 shows an example of the curve approximation from a thin point cloud with 2000 points generated from the moving least-squares technique with $H = 1$ and two iterations. To order the thin point cloud, we used

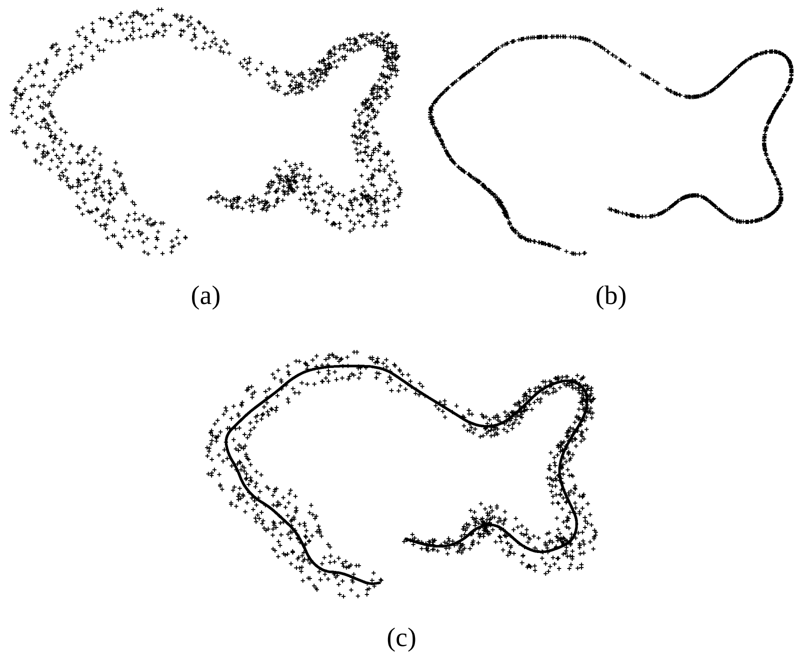


Fig. 13. 2D curve reconstruction: (a) 1000 points, (b) thin point cloud after two iterations, and (c) quadratic B-spline curve approximated from 76 ordered points.

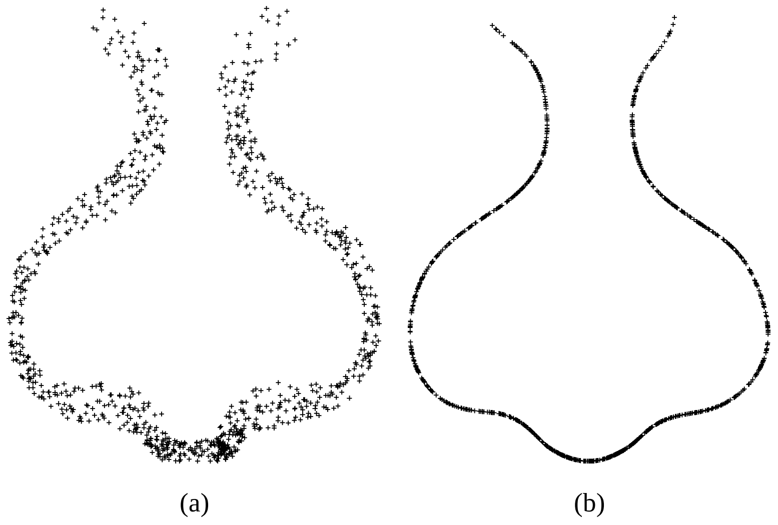
$h = 0.1$ and reduced the point set to 125 ordered points (Fig. 12(b)). Finally, the ordered points are approximated with a quadratic B-spline curve with 50 control points (Fig. 12(c)) using the chord length parametrization scheme (Hoschek and Lasser, 1993).

Figs. 13–16 illustrate some results of the curve reconstruction using the algorithm presented in this paper. 2D random data points were sampled from the area between two variable radius offset curves, $\mathbf{C}(t) + \mathbf{N}(t)r(t)$ and $\mathbf{C}(t) - \mathbf{N}(t)r(t)$, where $r(t)$ is a variable radius, and $\mathbf{N}(t)$ is a unit normal vector of a curve $\mathbf{C}(t)$ in the plane. Similarly, 3D random data points were sampled from the inside volume of a swept surface $\mathbf{S}(u, v) = \mathbf{C}(u) + \mathcal{F}(u)r(u)\mathbf{U}(v)$, where $r(u)$ is a variable radius, $\mathbf{U}(v)$ is a unit circle, and $\mathcal{F}(u)$ is the Frenet frame of a 3D spine curve $\mathbf{C}(u)$.

6. Pipe surface reconstruction

A *pipe surface* is generated as an envelope of a sphere with a constant radius whose center runs along a spine curve. Using the curve reconstruction algorithm, we can easily reconstruct the pipe surface from a given set of points. We assume that the (unit) normal vector at each data point has already been estimated (Hoschek and Lasser, 1993).

A pipe surface is defined by a *spine curve* and a *constant radius* of the swept sphere. The radius of the swept sphere can be computed in several different ways. A simple method is to collect some points from a point set, and then computing a torus which locally fits



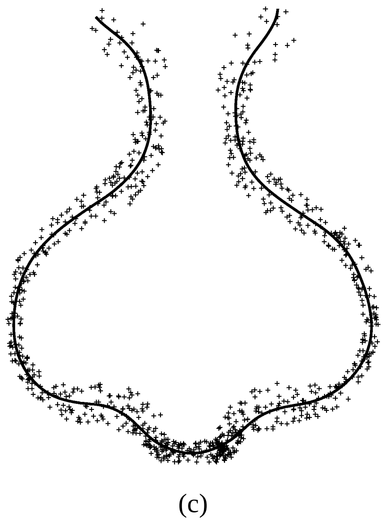


Fig. 14. 2D curve reconstruction: (a) 1000 points, (b) thin point cloud after two iterations, and (c) quadratic B-spline curve approximated from 87 ordered points.

the region. One can use the torus least-squares fitting (Lukács et al., 1997) or surface of revolution reconstruction (Pottmann et al., 1998b).

Once the radius r is found, each data point $\mathbf{P}_i$ is translated by a vector $r\mathbf{N}_i$, where $\mathbf{N}_i$ is a unit normal vector of $\mathbf{P}_i$. If a given point set is likely to represent a pipe surface, the translations generate a reasonably thin point cloud corresponding to a spine curve. Using the curve reconstruction algorithm that we presented in this paper, we can easily compute the spine curve.

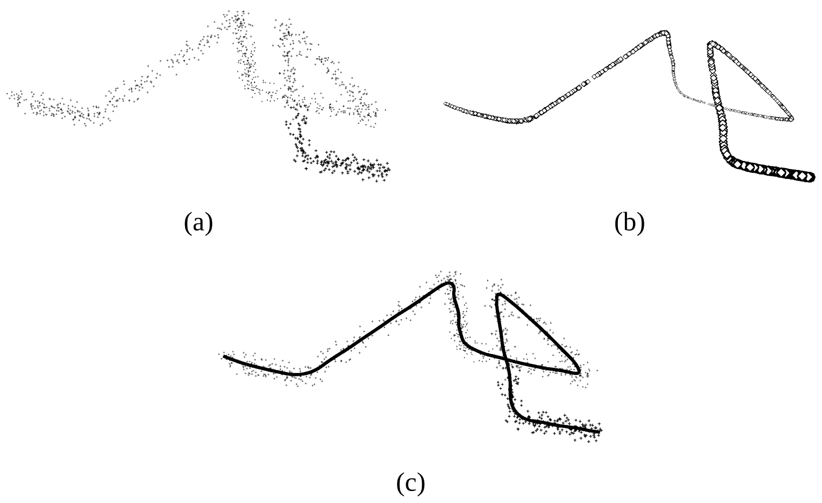


Fig. 15. 3D curve reconstruction: (a) 1000 points, (b) thin point cloud after two iterations, and (c) quadratic B-spline curve approximated from 83 ordered points.

Fig. 17 shows an example of the pipe surface reconstruction. After adding a perturbation factor $e(u, v)$ to an exact pipe surface $\mathbf{S}(u, v)$ (Fig. 17(a)), 1000 sample points and normal vectors are taken from the perturbed pipe surface (Fig. 17(b)). In Fig. 17(c), the original data points are translated towards the spine curve using unit normal vectors and the radius which is computed by the method in (Pottmann et al., 1998b). The moving least-squares technique is applied to the point set (Fig. 17(d)), and the spine curve is approximated (Fig. 17(e)).

7. Conclusion

In this paper, we presented a method to reconstruct a curve from an unorganized point set. Using the power of the moving least-squares technique, the point set can easily be reduced to a thin point cloud which is a near-best approximation in the sense of the local regression. We suggested some techniques to overcome the difficulties arising from the pure moving least-squares, from which we have achieved great improvements of the results.

In this paper, we used a simple method based on the correlation of the point set to find the weighting parameter H. For estimating an optimal weighting parameter H for each local regression, it is necessary to measure the width of the point cloud locally using some polygonal structures such as an α-shape (Edelsbrunner and Mücke, 1994) as well as the curvature analysis of the point set. It will also be interesting to consider non radial penalty functions. After computing the local regression line, we can modify the weighting function to give less penalties to the points in the tangent direction of the curve and more penalties to the points in the orthogonal direction. Nevertheless, we must keep in mind that the local

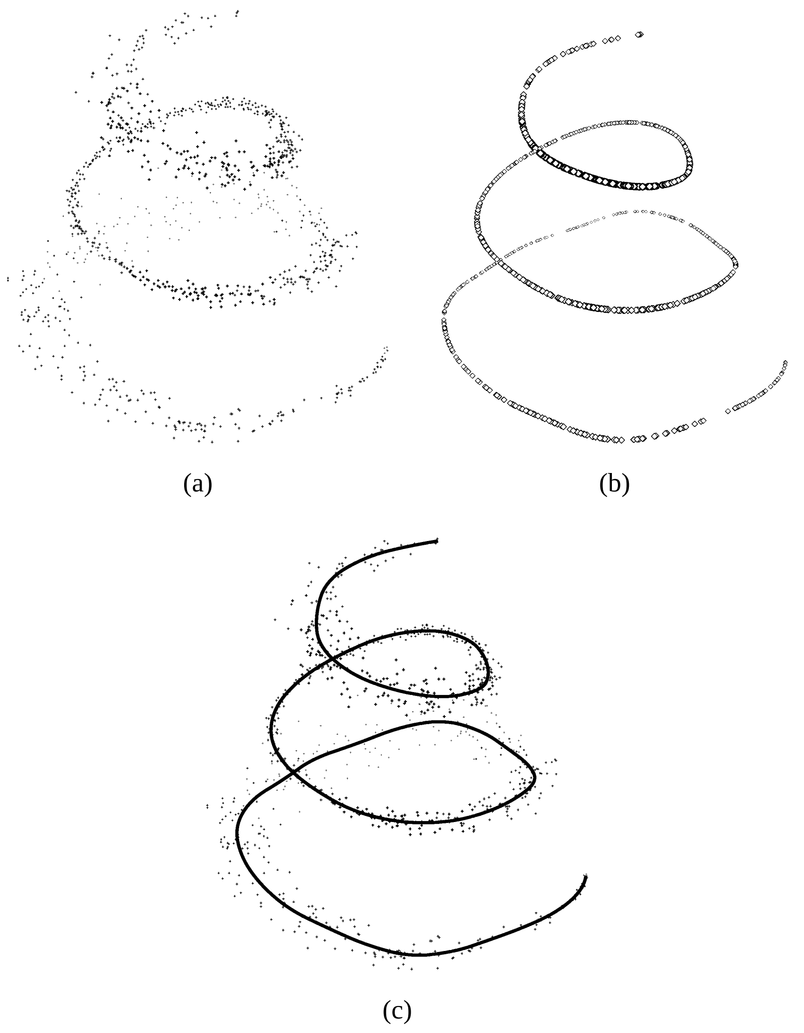


Fig. 16. 3D curve reconstruction: (a) 1000 points, (b) thin point cloud after two iterations, and (c) quadratic B-spline curve approximated from 67 ordered points.

regression must be as simple as possible because the local regression will be applied to every point in the set.

Acknowledgement

The author would like to thank Professor David Levin at Tel-Aviv University and Professor Helmut Pottmann at Vienna University of Technology for pointing the author to the moving least-squares method and giving many pieces of valuable advice. The

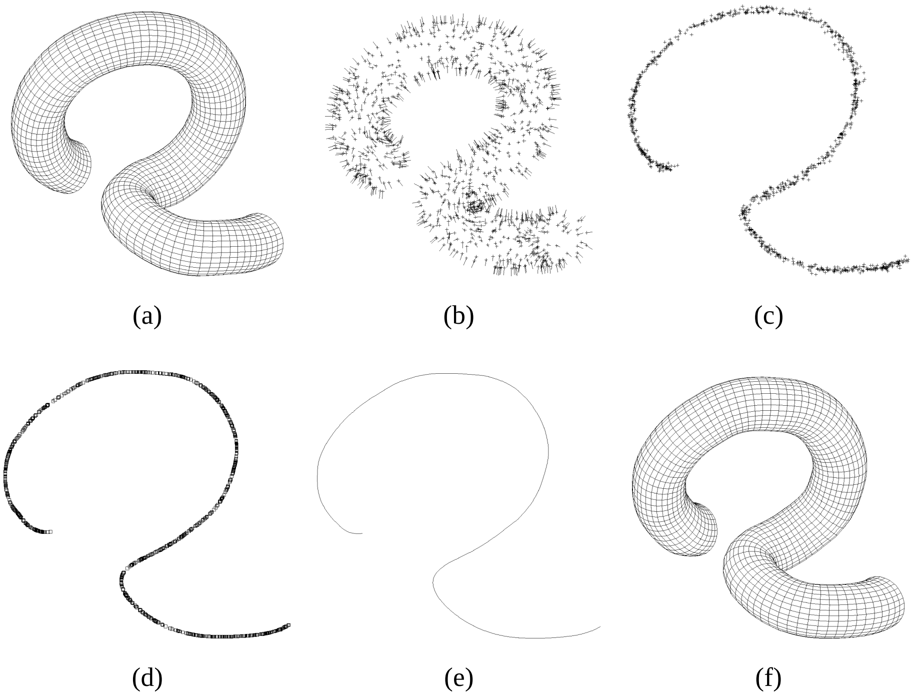


Fig. 17. Pipe surface reconstruction: (a) original pipe surface, (b) 1000 points and normal vectors, (c) data points are translated by the radius of the swept sphere, (d) thin point cloud after moving least-squares, (e) spine curve reconstructed, and (f) reconstructed pipe surface.

reviewers' comments are gratefully acknowledged. Much of this work was done while the author stayed at the Institute of Geometry, Vienna University of Technology, Austria. The research was supported in part by the Korea Science and Engineering Foundation (KOSEF) through its overseas postdoctoral fellowship program in the second half of the year 1997 and by grant No. 2UD99520 of the POSTECH Information Research Laboratories.

References

Amenta, N., Bern, M. and Kamvysselis, M. (1998), A new Voronoi-based surface reconstruction algorithm, in: SIGGRAPH 98 Conference Proceedings, 415–422.

Bajaj, C., Bernardini, F. and Xu, G. (1997), Reconstructing surfaces and functions on surfaces from unorganized 3D data, Algorithmica 19, 243–261.

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

Dedieu, J.-P. and Favardin, Ch. (1994), Algorithms for ordering unorganized points along parametrized curves, Numerical Algorithms 6, 169–200.

Edelsbrunner, H. and Mücke, E.P. (1994), Three-dimensional alpha shapes, ACM Transactions on Graphics 13, 43–72.

Fang, L. and Gossard, D.C. (1992), Fitting 3D curves to unorganized data points using deformable curves, in: Visual Computing (Proceedings of CG International ’92), Springer, Berlin, 535–543.

Ferley, E. and Cani-Gauscuel, M. and Attali, D. (1997), Skeletal reconstruction of branching shapes, Computer Graphics Forum 16 (5), 283–294.

Guo, B., Menon, J. and Willette, B. (1997), Surface reconstruction using alpha shapes, Computer Graphics Forum 16 (4), 177–190.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. and Stuetzle, W. (1992), Surface reconstruction from unorganized points, Computer Graphics 26, 71–78.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. and Stuetzle, W. (1993), Mesh optimization, Computer Graphics 27, 19–26.

Hoschek, J. and Lasser, D. (1993), Fundamentals of Computer Aided Geometric Design, A.K. Peters.

Lancaster, P. (1979), Moving weighted least-squares methods, in: Polynomial and Spline Approximation, Reidel, 103–120.

Levin, D. (1998a), The approximation power of moving least-squares, Mathematics of Computation 67, 1517–1531.

Levin, D. (1998b), Mesh-independent surface interpolation, private communication.

Lukács, G., Marshall, A.D. and Martin, R.R. (1997), Geometric least-squares fitting of spheres, cylinders, cones, and tori, in: Martin, R.R. and Varady, T. eds., RECCAD Deliverable Documents 2 and 3, Copernicus Project No. 1068, Report GML 1997/5, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest.

McLain, D. (1974), Drawing contours from arbitrary data points, The Computer Journal 17, 318–324.

McLain, D. (1976), Two dimensional interpolation from random data, The Computer Journal 19, 178–181.

Mencl, R. (1995), A graph-based approach to surface reconstruction, Computer Graphics Forum 14, 445–456.

Pitman, J. (1992), Probability, Springer, Berlin.

Pottmann, H. and Randrup, T. (1998), Rotational and helical surface approximation for reverse engineering, Computing 60, 307–322.

Pottmann, H., Lee, I.-K. and Randrup, T. (1998a), Reconstruction of kinematic surface from scattered data, in: Proceedings of Symposium on Geodesy for Geotechnical and Structural Engineering, Eisenstadt-Austria, 483–488.

Pottmann, H., Chen, H.-Y. and Lee, I.-K. (1998b), Approximation by profile surfaces, in: A. Ball et al., eds., The Mathematics of Surfaces VIII, Information Geometers.

Preparata, F.P. and Shomos, M.I. (1985), Computational Geometry: An Introduction, Springer, Berlin.

Sedgewick, R. (1988), Algorithms, 2nd edn., Addison-Wesley, Reading, MA.

Taubin, G. and Ronfard, R. (1996), Implicit simplicial models for adaptive curve reconstruction, IEEE Transactions on Pattern Analysis and Machine Intelligence 18, 321–325.

Wyvill, G. McPheeters, C. and Wyvill, B. (1986), Data structure for soft objects, Visual Computer 2, 227–234.