Reconstructing B-spline Curves from Point Clouds – A Tangential Flow Approach Using Least Squares Minimization

Yang Liu Huaiping Yang Wenping Wang
Department of Computer Science
The University of Hong Kong
Pokfulam Road, Hong Kong SAR, P. R. China
yliu, hpyang, wenping@cs.hku.hk

Abstract

We present a novel algorithm based on least-squares minimization to approximate point cloud data in 2D plane with a smooth B-spline curve. The point cloud data may represent an open curve with self intersection and sharp corner. Unlike other existing methods, such as the moving least-squares method and the principle curve method, our algorithm does not need a thinning process. The idea of our algorithm is intuitive and simple — we make a B-spline curve grow along the tangential directions at its two endpoints following local geometry of point clouds. Our algorithm generates appropriate control points of the fitting B-spline curve in the least squares sense. Although presented for the 2D case, our method can be extended in a straightforward manner to fitting data points by a B-spline curve in higher dimensions.

1. Introduction

As a classical fitting problem in CAD/CAM, computer graphics, computer vision, image processing and statistics, fitting a smooth curve to a set of data points has been studied in the past thirty years. Many techniques and theories have been published and utilized. Many existing methods assume that the order of data points is known, so the fitting or interpolating curve can be obtained by minimizing an error function or computational geometry methods [1, 4]. For many practical problems, the data points are usually unordered and noisy. It still remains an active research problem as how to reconstruct a smooth curve from point cloud data with correct topology. Most existing methods use a thinning process to smooth noisy data points at first and then fit a curve to the smoothed data points, which are sometimes assumed to be ordered. Levin [13] proposes a moving least squares (MLS) method to reduce noisy data points to a thinner set by applying local weighted regressions twice. Lee [12] improves MLS by choosing appropriate neighborhoods for regressions. But the computation of MLS is rather costly, since regression is done twice for each data point; moreover, these methods cannot handle self-intersection cases. Another approach is mapping a point cloud to an image. Pottmann [16] maps data points to a binary image, then fits a smooth curve to the image's medial axis. Goshtasby [6] constructs a potential function based on the mapped points in an image to generate a new gray image and computes ridge contours. Then the mapped points are ordered by their projections on the ridge contours and a fitting curve is reconstructed from the ordered points.

Clustering the data points is another commonly used approach to fitting a curve to point data. Yan [20] presents a fuzzy curve-tracing algorithm. Data points are divided to several clusters by fuzzy algorithms and the center points of all the clusters are connected. After reordering and eliminating loops, the connected poly-line is output as a fitting curve. This method again cannot handle a point cloud with self-intersections and sharp corners. Other more complex approaches have been developed in recent years, including the principle curve method [10] and self-organizing maps [11]. But these methods also cannot reconstruct the curve with self-intersection. Implicit curve fitting [18] is another effective technique, but it can only deal with a point cloud representing a closed curve, and has considerable difficulty in capturing self-intersection points and sharp corners.

Our new algorithm is based on the simple idea that a spline curve can be made to crawl and stretch along the curve shape defined by a point cloud. More specifically, we place a short smooth curve segment somewhere on the point cloud and let it grow along the tangential directions at its two endpoints following the local geometry of the point cloud. The two endpoints of the curve serve as detectors and new points in neighborhoods of the two detectors are used to pull the fitting curve to grow via solving a least squares problem (see Figure 1). In this paper we choose a B-spline curve as the fitting model. Note that a B-spline curve can also express a poly-line when its degree is 1.

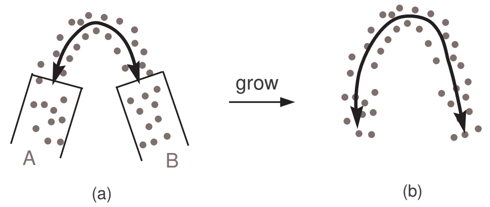


Figure 1. Intuitive explanation. (a) the two endpoints of the curve detect the new points in A and B; (b) the grown curve

A similar idea in surface reconstruction has been implemented by N.S. Sapidis and P.J. Besl [17]. They developed a region growing technique and approximated 3D points with a functional surface. The main disadvantage of region growing is that it is only compatible with simple shape. Recently Duan and Qin [5] also present an algorithm which reconstructs a point cloud with triangle meshes, where the final mesh is grown from a single seed triangle. The initial triangle grows in different tangential directions by appending to it new triangles, which are projected to back to the point cloud to preserve the model shape. Their growth mechanism is based on an evolutionary system of differential equations, and assumes noiseless data points representing a manifold. In contrast, our algorithm uses an iterative least squares approach for curve reconstruction and is capable of dealing with a curve with self-intersection, i.e. non-manifold data. Redistribution of the control points of the fitting curve is naturally achieved during the growth of the fitting curve via least squares minimization.

The remainder of the paper is organized as follows. Section 2 introduces preliminaries of B-spline curves and a local fitting algorithm using least squares. Section 3 presents our new algorithm and implementation details. In Section 4 several test examples are presented to demonstrate the effectiveness of our algorithm.

2. B-spline curve

An open B-spline curve of degree d in $\mathbb{R}^2$ is defined as [9, 15]

$$\mathcal{C}(t) = \sum_{i=0}^{n} B_{i,d}(t) \cdot P_i \tag{1}$$

where $B_{i,d}$ are the B-spline basis functions defined on the knot vectors

$$U = \{\underbrace{0, ..., 0}_{d+1}, u_{d+1}, ..., u_n, \underbrace{1, ..., 1}_{d+1}\} \tag{2}$$

and $\mathcal{P} = \{P_i \in \mathbb{R}^2, i = 0, \ldots, n\}$ are the control points of $\mathcal{C}(t)$.

The problem of fitting a B-spline curve to a point cloud can be formulated as follows. Given a point cloud $\mathcal{X} = \{X_k, k = 0, \ldots, N\}$, representing a *model shape*, find a B-spline curve $\mathcal{C}(t)$ with a fixed knot vector such that

$$\min_{\mathcal{P}} \left[\frac{1}{N+1} \sum_{k=0}^{N} d^2\left(X_k, \mathcal{C}(t)\right) + \frac{1}{n+1} \lambda \cdot f_s \right] \tag{3}$$

where $d(X_k, \mathcal{C}(t))$ is the minimal Euclidean distance from X_k to $\mathcal{C}(t)$ and f_s is a smoothing (regularization) term. λ is a coefficient which adjusts the ratio between distance errors and f_s. We use the smoothing term,

$$f_s = \alpha \int_0^1 ||\mathcal{C}'(t)||^2 \, dt + (1-\alpha) \int_0^1 ||\mathcal{C}''(t)||^2 \, dt$$

where $0 \leq \alpha \leq 1$. We also denote f_1 and f_2 as $\int_0^1 ||\mathcal{C}'(t)||^2 \; dt$ and $\int_0^1 ||\mathcal{C}''(t)||^2 \; dt$ respectively. Since the minimization of (3) is a nonlinear least squares problem, we use a Gauss-Newton algorithm [2, 14]. Firstly we derive a quadratic approximation to the original objective function. Since f_s is quadratic, we only consider the individual terms $d^2\left(X_k, \mathcal{C}(t)\right)$. For clarity, some symbols need to be defined firstly.

- Let $\mathcal{C}(t_k)$ denote the **projection point** of X_k on $\mathcal{C}(t)$, i.e. $t_k = \arg\min_t d(X_k, \mathcal{C}(t))$. t_k is called as the **projection parameter** of X_k.
- $V_k = X_k - \mathcal{C}(t_k)$.
- The local Frenet frame at $\mathcal{C}(t_k)$ on $\mathcal{C}(t)$ is $\{T_k, N_k\}$ where T_k is a unit tangential vector and N_k is a unit normal vector of the fitting curve at $C(t_k)$.

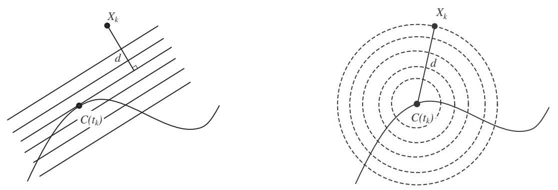
A quadratic approximation of $d^2\left(X_k, \mathcal{C}(t)\right)$ can be obtained as

$$d^2\left(X_k, \mathcal{C}(t)\right) \approx \gamma_k \cdot (V_k^T T_k)^2 + (V_k^T N_k)^2 \tag{4}$$

Then our local quadratic model is

$$\begin{aligned} \widetilde{F} \quad = \quad & \frac{1}{N+1} \sum_{k=0}^{N} \left(\gamma_k \cdot (V_k^T T_k)^2 + (V_k^T N_k)^2 \right) \\ & + \frac{1}{n+1} \lambda \cdot f_s \end{aligned} \tag{5}$$

It is known [19] that, when $\gamma_k = 0$, the above approximation leads to the Gauss-Newton method; when $\gamma_k = 1$, the above approximation amounts to the alternating method, also called the *intrinsic parametrization* by Hoschek [8]. The iso-values curves of the two approximations with $\gamma_k = 0$ and $\gamma_k = 1$ are shown in Figure 2. We use $\gamma_k = 0$ for the other X_k whose nearest points are their orthogonal-projection points on $\mathcal{C}(t)$, and $\gamma_k = 1$ for those data points X_k whose closest points on $\mathcal{C}(t)$ are the endpoints of $\mathcal{C}(t)$. In this way, the data points X_k in the former group will have little resistance to the tangential motion of points on the fitting curve, and the points X_k in the latter group will exert an attractive force to the two ends of the fitting curve to facilitate its growth.



(a) $\gamma_k = 0$ (b) $\gamma_k = 1$

Figure 2. Iso-values curves of distance error terms

3. Algorithm and Implementation

We list all parameters to be used in our algorithm:
PARAMETERS LIST

- $\mathcal{X}$: a point cloud.
- $\mathcal{C}(t)$: the B-spline curve.
- Q_0 and Q_1: the beginning and ending points of $\mathcal{C}(t)$.
- T_0 and T_1: the two unit tangential direction at Q_0 and Q_1.
- ξ: fitting tolerance.
- ϕ: converge speed tolerance.
- w: thickness.
- ρ: sampling density of the point cloud.
- δ: the parameter for correcting projection.
- $Head$ and $Tail$: growing sets at the two ends of the fitting curve.
- $\mathcal{A}$: the point set for current approximation.

The basic algorithm flow is as follows:

1. *Input*: a point cloud $\mathcal{X} = \{X_k, k = 0, \ldots, N\} \subset \mathbb{R}^2$.
2. *Data Analysis*: Compute the Euclidean Minimum Spanning Tree (EMST) and a cell partition of $\mathcal{X}$. Then estimate thicknesses of $\mathcal{X}$.
3. *Initial curve*: An empty set $\mathcal{A}$ is initialized. Pick a seed point S in $\mathcal{X}$ randomly. Add S and the neighboring points of S in EMST to $\mathcal{A}$. Fit a line L to $\mathcal{A}$ with orthogonal regression. Convert L to a B-spline curve $\mathcal{C}(t)$.
4. *Approximation*: Fit $\mathcal{C}(t)$ to $\mathcal{A}$ using knot insertion until a fitting tolerance is satisfied.
5. *Growing*: Use two tangential directions at two endpoints of $\mathcal{C}(t)$ to search for more new points in $\mathcal{X}$. If there are such new points, add them to $\mathcal{A}$ and go to "Approximation"; otherwise go to "Refinement".
6. *Refinement*: Project all the points of $\mathcal{A}$ to $\mathcal{C}(t)$ and use the resulting parameter values t_k to get a better fit of $\mathcal{C}(t)$ to the point set $\mathcal{A}$; knot insertion and fairing is used here to improve the shape of $\mathcal{C}(t)$. Should data shape be closed, a periodic B-spline curve can be computed to yield a closed fitting curve $\mathcal{C}(t)$.
7. *Output*: the B-spline fitting curve $\mathcal{C}(t)$.

3.1. Input

In our algorithm, we suppose that (1) $\mathcal{X}$ represents a single smooth curve, possibly with noise, self-intersection and sharp corners; (2) there are not many outliers, i.e. points far away from the medial axis of $\mathcal{X}$; and (3) the density of the point cloud is nearly uniform at different locations of the model shape. Figure 3 shows an unacceptable set of data points.

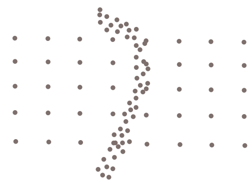


Figure 3. Unacceptable point cloud.

3.2. Data Analysis

Data Structure: The Euclidean Minimum Spanning Tree (EMST) has proven a useful data structure in curve reconstruction [12]. Lee uses the EMST to avoid clustering wrong points which should not belong to the current cluster although those points are not far away from it. We use the EMST to avoid adding wrong points to $\mathcal{A}$ in the growing phase. We use Prim's algorithm for computing the EMST of $\mathcal{X}$. The average, minimum and maximum edge length in the EMST are denoted as e_{ave}, e_{min}

and e_{max}. We use e_{max} as the estimated sampling density ρ of the point cloud if ρ is not provided.

Estimating Thickness: In most cases the thickness of the point cloud is not known. With the aid of the EMST we can estimate the thickness at different cell as follows.

1. Compute the bounding box of $\mathcal{X}$ in $\mathbb{R}^2$ and partition it into uniform cells $\{B_m, m = 0, \ldots, M\}$ (the size of every cell is $e_{max} \times e_{max}$). Suppose that the cell B_m contains n_{B_m} data points.

2. Estimate thickness ω_m for each cell B_m which contains data points in $\mathcal{X}$: (a) count the number of its nonempty neighboring cells, including B_m itself, in the eight directions (see Figure 4) $\vec{v}_{m,1} = (1,0), \vec{v}_{m,2} = (-1,0), \vec{v}_{m,3} = (0,1), \vec{v}_{m,4} = (0,-1), \vec{v}_{m,5} = (1,1), \vec{v}_{m,6} = (-1,-1), \vec{v}_{m,7} = (1,-1), \vec{v}_{m,8} = (-1,1)$. Denote $r_l, l = 1, \ldots, 8$ as the corresponding numbers. Let $Y_{i,j,k} = \min\{r_i, r_j, r_k\}$. Then $Y_{1,3,5}, Y_{1,4,7}, Y_{2,3,8}, Y_{2,4,6}$ are the thickness in terms of the number of cells in the four quadrants, each spanned by three direction vectors. Denote

$$Z = \{Y_{1,3,5}, Y_{1,4,7}, Y_{2,3,8}, Y_{2,4,6}\}$$

Then the thickness ω_m is defined to be

$$\omega_m = \begin{cases} e_{max} \times \max\limits_{Z_i \in Z} Z_i & , \quad \min\limits_{Z_i \in Z} Z_i = 1 \\ e_{max} \times \left(2 \max\limits_{Z_i \in Z} Z_i - 1\right) & , \quad \min\limits_{Z_i \in Z} Z_i > 1 \end{cases} \tag{6}$$

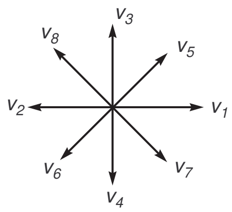


Figure 4. Eight directions

Fitting tolerance: There are many criterions for determining the fitting error of a B-spline curve. We use the average distance $E_{ave} = \left(\frac{1}{N+1}\sum\limits_{k=0}^{N} \|V_k\|^2\right)^{1/2}$, and check whether $E_{ave} < \xi$, a pre-specified tolerance. The user can set ξ. But sometimes it is preferably to set it automatically; in this case we set it relative to thicknesses of the point cloud letting $\xi = \left(\frac{1}{N+1}\sum\limits_{m=0}^{M}\left(\frac{\omega_m}{2}\right)^2 \cdot n_{B_m}\right)^{1/2}$.

3.3. Initial curve

We pick a point S as a seed in $\mathcal{X}$ randomly. If the point cloud has self-intersections or sharp corners, the user should avoid placing S in those areas. Choose those points which are neighbors of S in the EMST and the shortest distance from them to S in the EMST are less than a user-defined value l_S (We set $l_S = 10 \cdot \omega_i$, for $S \in B_i$ in our implementation). Then group them as a set $\mathcal{I}$, representing a neighborhood of the point S. We fit a straight line L to the point set $\mathcal{I}$, using orthogonal regression, which is done in 2D plane using PCA (principal component analysis); (this is a simple constrained optimization problem even in higher dimensions). Then we can convert L to a B-spline curve, using $d+1$ uniformly distributed control points on the line L. We project all the points in $\mathcal{I}$ to L orthogonally to obtain their corresponding projection points and parameters. The projection points which correspond to the minimum and maximum parameter will serve as the two endpoints of the initial B-spline curve. The knot vector is also set to be equal-spaced. After obtaining the initial B-spline curve, we compute the projection points and parameters for all points in $\mathcal{I}$. Record them for approximation. Two sets $Head$ and $Tail$ are initialized for growing: $Head = Tail = \mathcal{I}$ and $\mathcal{A} = \mathcal{I}$.

3.4. Approximation

Since the projection point and parameter of every data point in the set $\mathcal{A}$ are known, we may compute new control points of $\mathcal{C}(t)$ by solving the linear squares problem (5). If the projection point of $X_i \in \mathcal{A}$ is close to one endpoint of $\mathcal{C}(t)$, we set $\gamma_i = 1$ otherwise $\gamma_i = 0$. For preventing the linear system from becoming ill-posed, we add a regularization term $\tau \sum_{i=0}^{n}(P_i - P_{i,old})^2$ where P_i are variables and $P_{i,old}$ are current control points before optimization. There are many techniques for choosing an optimal τ such as general cross validation(GCV) and L-curve criterion [3,7]. But these techniques are too costly because sub-nonlinear and complex optimization problems need to be solved. We employ the Levenberg-Marquardt strategy [14] to adjust τ dynamically such that $\widetilde{F}$ in Eqn (5) is always decreased after every iteration of optimization. This strategy is used for all of our testing examples to be presented in Section 4. After each iteration of optimization, we update the projection points and parameters of $\mathcal{A}$ and check whether the tolerance is reached. The updated procedure will be presented in Section 3.6.

The integration of the smoothness term f_s's derivatives can be computed by a numerical integration method such as Gauss quadrature. To avoid undesirable large variations of $\partial f_s/\partial\mathcal{P}$, we normalize the matrices $M_1 = \partial f_1/\partial\mathcal{P}$ and $M_2 = \partial f_2/\partial\mathcal{P}$ to $\widetilde{M_1} = M_1/\max\limits_{i,j}\|M_{1i,j}\|$ and $\widetilde{M_2} = M_2/\max\limits_{i,j}\|M_{2i,j}\|$.

The tolerance can be checked as follows: let $Err_{ave,j}$ be current average error $\left(\frac{1}{n_A}\sum_{X_k\in\mathcal{A}}\|X_k-\mathcal{C}(t_k)\|^2\right)^{1/2}$ after the j-th iteration where $\mathcal{C}(t_k)$ is the projection point of X_k and n_A is the number of points in $\mathcal{A}$. If tolerance is not satisfied and the relative error reduction $\frac{(Err_{ave,j-1}-Err_{ave,j})}{Err_{ave,j-1}}$ is smaller than a threshold ϕ (we use $\phi = 0.1\%$), we insert a new knot to $\mathcal{C}(t)$.

3.4.1. Knot insertion We propose two knot insertion strategies for the following two cases.

Case 1 ($A = \mathcal{X}$): In this case, all the data points can be projected to $\mathcal{C}(t)$. Thus we improve the fitting accuracy by knot insertion. There are several published methods for control point insertion [9, 15, 21]. We add a knot at the place where the maximum error occurs.

Case 2 ($A \subset \mathcal{X}$): The B-spline curve is in its growing process. In this case, most knot insertion methods do not work well, since the B-spline curve will change a lot after the next iteration and the nonuniform knot vector will affect the curve's quality. Therefore we insert a knot u' at $[u_d, u_n]$ and redistribute all the knots and make them equally spaced.

3.5. Growing

The growing procedure is the most important process in our algorithm. Let T_0 and T_1 be two tangential directions at the endpoints Q_0 and Q_1 of the B-spline curve. Since $\mathcal{C}(t)$ approximates $\mathcal{A}$ in the least squares sense, T_0, T_1 represent the local geometry around the neighborhood of Q_0 and Q_1 in $\mathcal{X}$. We shall add new points to $\mathcal{A}$ along T_0 and T_1.

Construct Search Areas: (1) Get the thickness w_0 of the cell which contains Q_0 and the thickness w_1 of the cell which contains Q_1; (2) Construct two rectangles B_0 and B_1 attached to Q_0 and Q_1 along $-T_0$ and T_1 (see Figure 5); we shall call the rectangles *banded region*. The widthes of B_0 and B_1 are w_0 and w_1; (we use local thickness to avoid involving too many nonlocal points. Figure 6 shows an unacceptable case). (3) For each point $X_i \in Head$, check all the adjacent points X_p of X_i in the EMST. If $X_p \in B_0$, add it to $Head$ and set X_p's projection point as Q_0 and projection parameter as 0. X_i is called an **introducer** of X_p. This process is executed recursively until no more points are added to $Head$. Similarly, for each point $X_i \in Tail$, check all the adjacent points X_p of X_i in the EMST. If $X_p \in B_1$, add it to $Tail$ and set X_p's projection point to Q_1 and projection parameter to 1.

3.6. Finding projection points

In every optimization iteration, we need to find the projection point $\mathcal{C}(t_i^*)$ and parameter t_i^* of every point X_i in $\mathcal{A}$

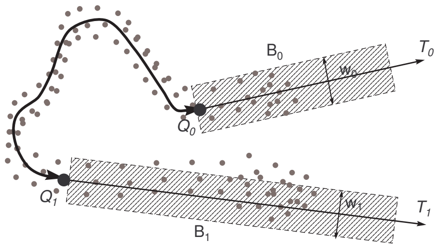


Figure 5. The growing set $\mathcal{A}$.

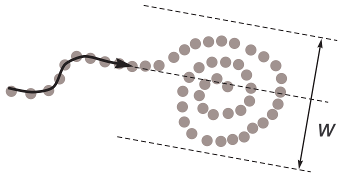


Figure 6. Choose thickness

for constructing the equation (3). We use the existing projection point and parameter as initial conditions and apply the Newton formula in Eqn. (7) [9, 15] to find the new projection points and parameters iteratively.

$$t_i^* = t_i - \frac{(X_i-\mathcal{C}(t_i))\cdot\mathcal{C}'(t_i)}{(X_i-\mathcal{C}(t_i))\cdot\mathcal{C}''(t_i)-\mathcal{C}'(t_i)^2} \qquad (7)$$

Filtering points: When $\mathcal{C}(t)$ marches through the self-intersection region, it is possible that $\mathcal{C}(t)$ is mislead by the data points belonging to another branch. We use a filtering process to reduce the influence of these wrongly included points. Since for each point X_i in $\mathcal{A}$, the thickness ω_i of its surrounding grid has been estimated in Section 3.2. Then the distance from X_i to a good fitting curve should be less than half of the thickness ω_i. Thus the filtering strategy is as follows: if $\|X_i - \mathcal{C}(t_i^*)\| \geq \omega_i/2$ and $\frac{|(X_i-\mathcal{C}(t_i^*))\cdot\mathcal{C}'(t_i^*)|}{\|X_i-\mathcal{C}(t_i^*)\|\|\mathcal{C}'(t_i^*)\|} < 10^{-8}$, then X_i will not be involved in Eqn (5). The latter condition is to ensure that the points which pull the endpoints of $\mathcal{C}(t)$ will not be filtered.

Handling sharp corners: In general, the projection point is the nearest point from X_i to $\mathcal{C}(t)$. But sometimes it will cause problems when the curve is passing through a sharp corner. In Figure 7, F_1 and F_2 are the projection points of X_1 and X_2 respectively and X_1 is the introducer of X_2. The projection points of $X_2, \ldots, X_7$ are at the left of F_1. If we approximate (4) directly, $\mathcal{C}(t)$ will be distorted because there is no pulling force for the end of the curve to pass through $X_2, \ldots, X_7$. Suppose that X_i is one of the current data points and X_p is the introducer of X_i, with projection points F_i and F_p, respectively. The nearest endpoint of the curve to X_i and X_p is P. We use the following strategy: compute the arc-length $l_i = \widehat{F_i P}$ and $l_p = \widehat{F_p P}$. If $l_i > \delta$ and $l_p < \delta$ ($\delta = e_{ave}$ in our experiments), we set the projection point of X_i to P with the projection parameter 0 or 1 (depended on the position of P at $\mathcal{C}(t)$). Therefore all the points like $F_2, \ldots, F_7$ will always pull the endpoints of the fitting curve to move towards them.

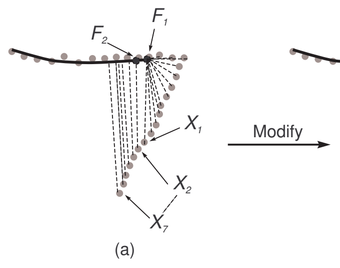


Figure 7. Sharp corner.

3.7. Other cases

In three other cases, no new data points can be added into B_0 or B_1 and the growing of $Head$ or $Tail$ will halt. These two cases are processed in our algorithm as follows.

1. The direction T_0 or T_1 does not respect the local geometric information of $\mathcal{X}$ due to the lack of degree of freedom in approximation (Figure 8). By inserting knots to $\mathcal{C}(t)$ and doing approximation again, the banded region will be able to contain new points in $\mathcal{X}$.

2. $Head$ or $Tail$ stops growing when fitting a point cloud with self-intersections (Figure 9). Although the EMST is connected, the topology may be not the same as that of the desired shape of $\mathcal{X}$. We need to fill the gap by adding edges. Suppose that $Tail$ is unchanged. Let S_1 denote all the data points in $(B_1 \cap \mathcal{X})/Tail$. Find the pair of points $X_p \in S_1$ and $X_q \in Tail$ with the minimum distance between S_1 and $Tail$. If their distance is less than the sampling density, i.e. $\|X_p - X_q\| \leq \rho$, connect X_p to X_q. This gap-filling procedure recovers the desired topology of the point cloud data by modifying the EMST structure.

3. A very sharp corner can stop the growing of the fitting curve, despite that we have used "introducers" to overcome the wrong projection problems in Section 3.6. Figure 10 shows a very sharp corner with its EMST. All the introducers of the points in C_1 and C_2 can be traced to P. But the projection points of the points in C_1 are far away from Q_1, the criterion in Section 3.6 will not satisfy. So the points in C_1 will not pull Q_1 to move to them. When this happens, we leave the current B-spline curve alone and generate a new seed (another initial B-spline curve) from the un-visited data points. The new curve is then growing up in the same way as before and finally the two B-spline curves are merged together to produce the desired result. This new seeding strategy can also be used to handle complex cases when the model shape is composed of multiple branches, as illustrated in Figure 11.

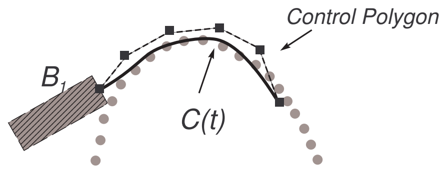


Figure 8. Less control points.

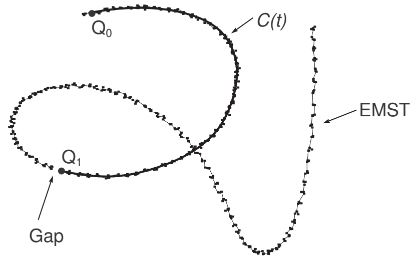


Figure 9. EMST with wrong topology.

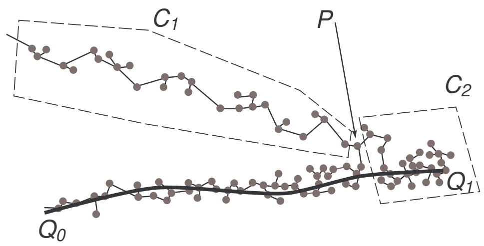
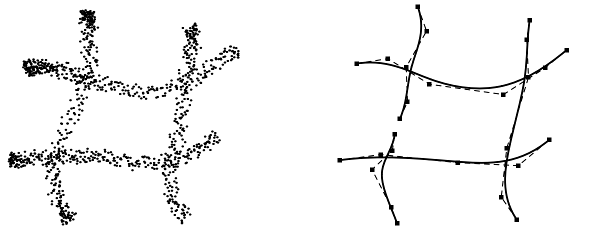


Figure 10. Very sharp corner.

4. Experiments

In this section, we present some test examples to demonstrate the effectiveness of our method. Several representa-



(a) point clouds (b) fitting curves without merging

Figure 11. Multiple branches.

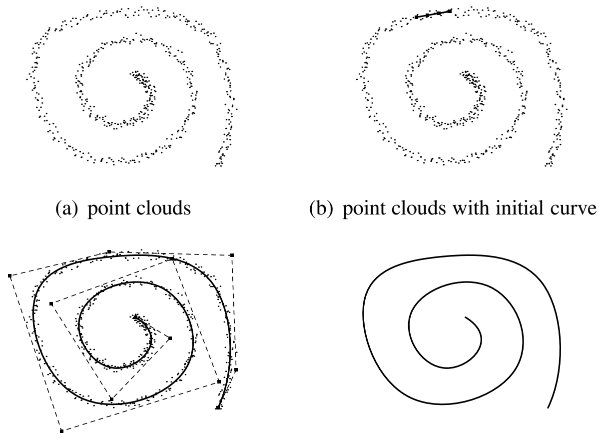
tive examples, including a swirl shape with varying thicknesses (Example 1), a shape with sharp corners(Example 2) and self-intersections (Example 3) are tested in our experiment. We shall also show that our method can be used to extract the skeleton of a shape by taking the uniformly sampled points inside the shape as the point cloud data to be fitted (Example 4). There are two parameters λ and α which control the smoothness of the fitting curve and are pre-specified by the user. In our test examples, we usually set $\alpha = 0.5$, which yields a balance between the stiffness and flexibility of the fitting B-spline curve. λ should be adjusted according to the distribution of the sampling density and thickness of the point cloud data. Larger values of λ should be used when the sample density is quite nonuniform and there is considerable thickness of data points. We also show a failed example(Example 5) which uses a very small λ. All the experiments were run on a PC with a Pentium 2.4GHz CPU and 512MB RAM.

Example 1 Swirl Shape : (Figure 12) The number of points is 1018*. Parameters:* $\lambda = 0.001, \alpha = 0.5$*. The number of control points is* 12 *and the degree of* $\mathcal{C}(t)$ *is* 3*. The time for generating EMST and spacial partition is 0.02 seconds, and the total computation time with 142 iterations is 0.58 seconds.*

Example 2 Sharp corner: (Figure 13) The number of points is 842*. Parameters:* $\lambda = 0.01, \alpha = 0.5$*. The number of control points is* 13 *and the degree of* $\mathcal{C}(t)$ *is* 3*. The time for generating EMST and spacial partition is 0.04 seconds, and the total computation time with 152 iterations is 1.27 seconds.*

Example 3 self-intersection shape with large noisy: (Figure 14) The number of points is 2000*. Parameters:* $\lambda = 0.02, \alpha = 0.5$*. The number of control points is* 12 *and the degree of* $\mathcal{C}(t)$ *is* 3*. The time for generating EMST and spacial partition is 0.32 seconds, and the total computation time with 68 iterations is 2.37 seconds.*

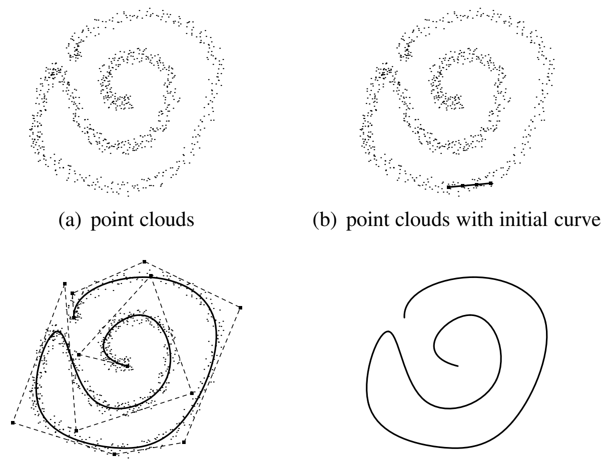
Example 4 ξ *shape: (Figure 15)* 666 *points are uniformly sampled inside the character* ξ*. Parameters:* $\lambda = 0.06, \alpha =$



(a) point clouds (b) point clouds with initial curve

(c) fitting result (d) fitting curve

Figure 12. Swirl shape



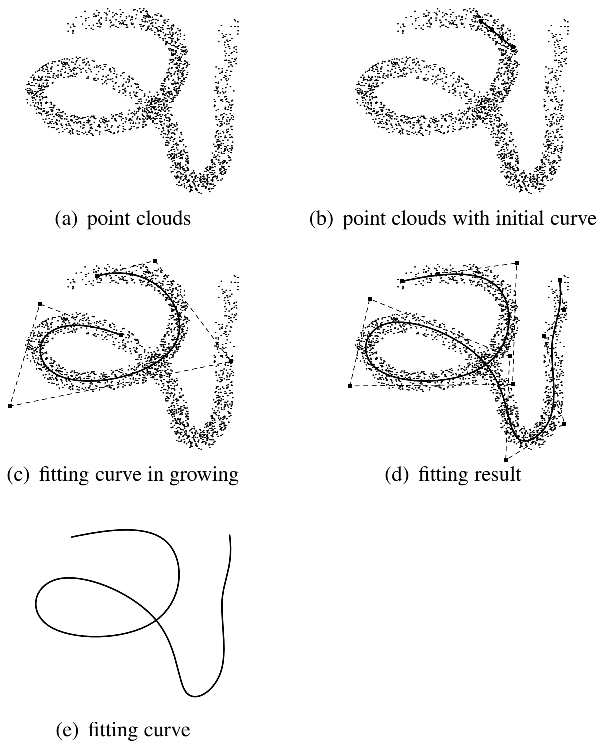
(a) point clouds (b) point clouds with initial curve

(c) fitting result (d) fitting curve

Figure 13. Sharp corner

0.5*. The number of control points is* 21 *and the degree of* $\mathcal{C}(t)$ *is* 3*. The time for generating EMST and spacial partition is 0.028 seconds, and the total computation time with 40 iterations is 0.59 seconds.*

Example 5 The data points and the initial curve are same as them in Example 2. Parameters: $\lambda = 0.000001, \alpha = 0.5$*. After* 196 *iterations, the fitting curve cannot grow since one end of the curve doesn't follow the local geometry and no more points can be added to this end(Figure 16).*

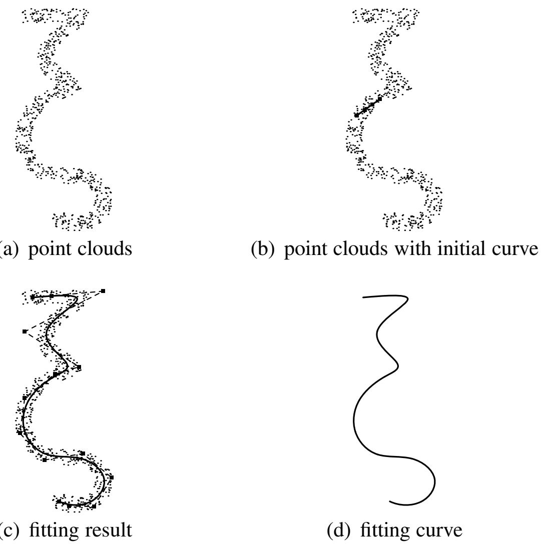


(a) point clouds (b) point clouds with initial curve

(c) fitting curve in growing (d) fitting result

(e) fitting curve

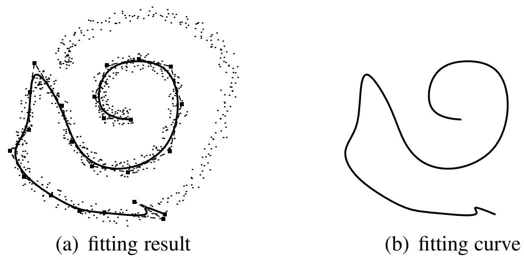
Figure 14. Self-intersection



(a) point clouds (b) point clouds with initial curve

(c) fitting result (d) fitting curve

Figure 15. Skeleton of the character ξ



(a) fitting result (b) fitting curve

Figure 16. failed example

5. Conclusion

We have presented a novel algorithm based on least-squares method to approximate noisy/non-noisy point clouds with smooth B-spline curves. Unlike the existing methods our algorithm does not need a thinning process. The basic idea of our algorithm is encouraging the B-spline curve to grow along tangential directions which respect local geometric changes of the model shape. Our method generates appropriate control points for the B-spline curve in the least squares, due to the use of the quadratic error term that does not inhibit tangential flow. We have also applied this algorithm successfully to point clouds which contain self-intersections and/or sharp corners. Our method can be extended to computing a B-spline fitting curve from a data cloud in higher dimensions.

There are several issues with our method that should be studied in further research. Our method is based on tangential direction detection and lease squares fitting. When a point cloud is sampled non-uniformly and has large thicknesses variation, the least squares method without weights may not produce desired results, and the EMST may not faithfully represent the correct topology of the point cloud. In such a case, the local regression approach in Lee's approach [12] may produce a statistical more meaningful result.

We have assumed implicitly that the two branches at an intersection have rather different tangential directions, since the tangential direction detection mechanism will have difficulty in distinguishing two branches with similar directions. From our experiences, relatively narrow band searching regions can help find the right branch if the local geometry at self-intersections changes slowly. Note that we do not explicitly detect intersections in the model shape. Thus our further work will include explicit feature detection to facilitate the fitting curve marching through intersections and adjusting the smoothing term automatically.

6. Acknowledgements

This research is supported by a HKU CRCG Grant on Basic Research. The authors wish to thanks the anonymous reviewers for their careful reading and suggestions.

References

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

[2] A. Atieg and G. A. Watson. A class of methods for fitting a curve or surface to data by minimizing the sum of squares of orthogonal distances. *Journal of Computational and Applied Mathematics*, 158:227–296, 2003.

[3] A. Bjorck. *Numerical Methods for Least Squares Problems*. Mathematics Society for Industrial and Applied Mathematics, Philadelphia, 1996.

[4] T. K. Dey, K. Mehlhorn, and E. Ramos. Curve reconstruction: connecting dots with good reason. *Comput. Geom. Theiry & Appl.*, 15:229–244, 2000.

[5] Y. Duan and H. Qin. 2.5d active contour for surface reconstruction. In *Proceedings of 8th international workshop on Vision, Modeling and Visualization*, pages 431–439, Munich, Germany, November 2003.

[6] A. A. Goshtasby. Grouping and parameterizing irregularly spaced points for curve fitting. *ACM Transaction on Graphics*, 19(3):185–203, 2000.

[7] P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems*. Society for Industrial and Applied Mathematics, Philadelphia, 1998.

[8] J. Hoschek. Intrinsic parameterization for approximation. *Computer Aided Geometric Design*, 5:27–31, 1988.

[9] J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. AK Peters, 1993.

[10] B. Kégl, A. Krzyzak, T. Linder, and K. Zeger. Learning and design of pricipal curves. *IEEE Transactionon Pattern Analysis and Machine Intelligence*, 22(3):281–297, 2000.

[11] G. S. Kumar, P. K. Kalra, and S. G. Dhande. Curve and surface reconstruction from points: an approach based on self-organizing maps. *Applied Soft Computing*, 5(1):55–66, 2004.

[12] I.-K. Lee. Curve reconstructionfrom unorganized points. *Computer Aided Geometric Design*, 17:161–177, 2000.

[13] D. Levin. The approximation power of moving least-squares. *Mathematics of Computation*, 67(224):1517–1531, 1998.

[14] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer Verlag, 1999.

[15] L. Piegl and W. Tiller. *The NURBS book*. Springer, New York, 2nd edition, 1997.

[16] H. Pottmann and T. Randrup. Rotational and helical surface approximation for reverse engineering. *Computing*, 60(4):307–322, 1998.

[17] N. S. Sapidis and P. J. Besl. Direct construction of polynomial surfaces from dense range images through region growing. *ACM Trans. Graph.*, 14(2):171–200, 1995.

[18] G. Taubin. An improved algorithm for algebraic curve and surface fitting. In *4th Int. Conf. on Computer Vision*, pages 658–665, 1993.

[19] W. Wang, H. Pottmann, and Y. Liu. Fitting b-spline curves to point clouds by squared distance minimization. Technical Report TR-2004-11, Dept. of Computer Science, The University of Hong Kong, 2004. http://www.cs.hku.hk/research/techreps/document/TR-2004-11.pdf.

[20] H. Yan. Fuzzy curve-tracing algorithm. *IEEE Transactions on Systems, Man,and Cybernetics–Part B: Cybernetics*, 31(5):768–780, 2001.

[21] H. P. Yang, W. Wang, and J. G. Sun. Control point adjustment for b-spline curve approximation. *Computer-Aided Design*, 36:639–652, 2004.