HKU CS Tech Report TR-2004-11

Fitting B-Spline Curves to Point Clouds by Squared Distance Minimization

Wenping Wang[1], Helmut Pottmann[2], Yang Liu[1]
[1] Department of Computer Science, The University of Hong Kong, Hong Kong SAR, China
[2] Geometric Modeling and Industrial Geometry, Vienna University of Technology, Austria

Abstract

Computing a curve to approximate data points is a problem encountered frequently in many applications in computer graphics, computer vision, CAD/CAM, and image processing. We present a novel and efficient method, called *squared distance minimization* (SDM), for computing a planar B-spline curve, closed or open, to approximate a target shape defined by a *point cloud*, i.e., a set of unorganized, possibly noisy data points. We show that SDM outperforms significantly other optimization methods used currently in common practice of curve fitting. In SDM a B-spline curve starts from some properly specified initial shape and converges towards the target shape through iterative quadratic minimization of the fitting error. Our contribution is the introduction of a new fitting error term, called the *squared distance (SD) error term*, defined by a quadratic approximant of squared distances from data points to a fitting curve. The SD error term measures faithfully the geometric distance between a fitting curve and a target shape, thus leading to faster and more stable convergence than the point distance (PD) error term, which is commonly used in computer graphics and CAGD, and the tangent distance (TD) error term, which is adopted in the computer vision community. To provide a theoretical explanation of the superior performance of SDM, we formulate the B-spline curve fitting problem as a nonlinear least squares problem and conclude that SDM is a quasi-Newton method, which employs a carefully chosen positive definite approximant to the true Hessian of the objective function. Furthermore, we show that the method based on the TD error term is a Gauss-Newton iteration which is unstable for target shapes with corners, whereas optimization based on the PD error term is the alternating method that is known to have linear convergence.

Keywords: Curve fitting, Squared distance minimization, nonlinear least squares, B-Spline curve, unorganized point cloud

1 Introduction

We consider the following problem: Given a set of unorganized data points X_k, $k = 1, 2, \ldots, n$, in the plane, compute a planar B-spline curve to approximate the points X_k. The data points X_k are assumed to represent the shape of some unknown planar curve, which can be open or closed, but not self-intersecting; this curve is called a *target curve* or *target shape*. We suppose that unorganized

data points, often referred to as a *point cloud* or *scattered data points* in the literature, may have non-uniform distribution with considerable noise; this assumption makes it difficult or impossible to order data points along the target curve. Hence, we assume that such an ordering is not available.

The above problem can be formulated as a nonlinear optimization problem as follows. Consider a B-spline curve $P(t) = \sum_{i=1}^{m} P_i B_i(t)$ with control points P_i. We suppose that the order and the knots of the B-spline curve are fixed, so they are not subject to optimization. Given data points X_k, $k = 1, 2, \ldots, n$, we want to find the control points P_i, $i = 1, 2, \ldots, m$, such that the objective function

$$f = \frac{1}{2} \sum_{k=1}^{n} d^2(P(t), X_k) + \lambda f_s \tag{1}$$

is minimized, where $d(P(t), X_k) = \min_t ||P(t) - X_k||$ is the distance of the data point X_k to the curve $P(t)$, f_s is a regularization term to ensure a smooth solution curve and λ is a positive constant to modulate the weight of f_s. Here the distance $d(P(t), X_k)$ is measured orthogonal to the curve $P(t)$ from X_k. The exceptional case where the shortest distance from X_k to an open curve $P(t)$ occurs at an endpoint of $P(t)$ will be discussed separately in Section 5.

We present a novel solution to the present problem that approximates unorganized data points with a B-spline curve that starts with some properly specified initial curve and converges through iterative optimization towards the target shape of data points. Our contribution is the introduction of a novel error term defined by a quadratic approximant of squared distances from data points to the fitting curve. Therefore, this new error term is called the *squared distance error term* or *SD error term*, and the resulting iterative minimization scheme will be referred to as *squared distance minimization* or *SDM*. Because the SD error term measures faithfully the geometric distance between data points and the fitting curve, SDM converges fast and stably, in comparison with other commonly used error terms, as will be discussed shortly.

The remainder of the paper is organized as follows. We first review related previous work. Then we introduce the SD error term, outline our SDM method, and use some test examples to show the superior performance of SDM in comparison with two other commonly used methods — point distance minimization (PDM) and tangent distance minimization (TDM). We shall also discuss the B-spline curve fitting problem from the viewpoint of optimization to provide insights into the superior performance of SDM. We shall show that SDM is, in fact, a quasi-Newton method which employs a carefully chosen positive definite approximant to the true Hessian of the objective function. We shall also show that TDM is a Gauss-Newton method. Furthermore, PDM is, in fact, the alternating method for solving a separable problem and is known to have only linear convergence [1, 33]. This systematic study of the relationship between these curve fitting methods and standard optimization techniques is another contribution of our work.

2 Related Work

2.1 Spline curve fitting techniques

Fitting a curve to a set of data points is a fundamental problem in graphics (e.g. [7, 23, 25, 29, 35]) and many other application areas. Instead of attempting a comprehensive review, we shall only discuss some main results in the literature to provide a background for our work.

Let $X_k \in R^2$, $k = 1, 2, \ldots, n$, be unorganized data points representing a target shape, which is to be approximated by a closed or open planar B-spline curve $P(t) = \sum_{i=1}^{m} B_i(t)P_i$, where the $B_i(t)$ are the B-spline basis functions of a fixed order and knots, and the P_i are the control points. Since f in Eqn. (1) is a nonlinear objective function, iterative minimization comes as a natural approach. Many existing B-spline curve fitting methods invoke a data parameterization procedure to assign a parameter value t_k to each data point X_k; in some methods dealing with ordered data points, the chord length method or the centripetal method [4, 13, 17] are used for data parameterization. Then the function

$$\hat{f} = \frac{1}{2} \sum_{k} ||P(t_k) - X_k||^2 + \lambda f_s, \tag{2}$$

which is a local model of f in (1), is minimized to yield updated control points P_i, and hence the fitting curve $P(t)$. Since $\hat{f}$ is quadratic in the P_i, $\hat{f}$ can easily be minimized by solving a linear system of equations.

Hoschek [12] proposes an iterative scheme, called *intrinsic parametrization*, to assign parameter values t_k to data points X_k. Here, with some initial fitting curve $P(t)$ that roughly matches the shape of input data points, each t_k is computed by finding the closest point from the data point X_k to the current curve $P(t)$; this closest point is also called the *foot point* of X_k. (Note that only an approximate formula is used in [12] to obtain the parameter value t_k of the foot point. Recently the foot point computation is carefully studied and improved in [31].) Then with the known t_k, the same function in Eqn. (2) is minimized to get the updated fitting curve $P(t)$. These two steps of data parameterization and minimization are performed iteratively until convergence is attained.

We note that the error term $||P(t_k) - X_k||^2$ in (2) measures the squared distance between the data point X_k and a particular point $P(t_k)$ on the fitting curve; therefore, we call this error term the *point distance error term* or the *PD error term*, and denote it by

$$e_{PD,k} = ||P(t_k) - X_k||^2. \tag{3}$$

(The PD error term is illustrated in Figure 1.) Accordingly, the intrinsic parameterization method by Hoschek [12] will be called *point distance minimization*, or *PDM* for short. Note that, since the ordering of data points is not required for data parameterization via foot point projection, PDM is applicable to fitting a curve to a point cloud.

PDM or its simple variants are the most commonly applied method for curve fitting in computer graphics and CAD [7, 12, 24, 25, 31]. The same idea of PDM is also widely used for surface fitting [3, 5, 8–11, 19, 20, 34, 36, 37], with B-spline surfaces as well as other types of surfaces. The popularity

of PDM might be explained by the fact that the error term $e_{PD,k}$ is derived by simply substituting t_k in the squared distance $d^2(P(t), X_k)$ in the original objective function f in (1). However, considering that $P(t_k)$ is a variable point depending on the variable control points, we shall see that $e_{PD,k}$ is a rather poor approximation to $d^2(P(t), X_k)$, thus causing slow convergence. As a matter of fact, the present paper is just about how to use a better approximation of $d^2(P(t), X_k)$ to devise a more efficient optimization scheme. Figure 4 shows the slow convergence of PDM in comparison with SDM, the new method we are going to propose.

Another error term, used in the computer vision community (e.g. [2]), is defined by

$$e_{TD,k} = [(P(t_k) - X_k)^T N_k]^2, \tag{4}$$

where N_k is a unit normal vector of the curve $P(t)$ at the point $P(t_k)$. We shall call $e_{TD,k}$ the *tangent distance error term* or the *TD error term*, since it measures the squared distance from X_k to the tangent of the curve $P(t)$ at $P(t_k)$. The TD error term is illustrated in Figure 2. Note that N_k is fixed and thus not subject to optimization.

The TD error term $e_{TD,k} = [(P(t_k) - X_k)^T N_k]^2$ can also be used in combination with data parametrization via projection to yield a B-spline curve fitting method, as used in [2] for boundary extraction in motion tracking. We shall call this method *tangent distance minimization* or *TDM*. TDM minimizes in each iteration the function

$$f_{TD} = \frac{1}{2} \sum_k e_{TD,k} + \lambda f_s. \tag{5}$$

Hence, treating the control points P_i as variables to be optimized, the TD error term measures the squared distance from the point X_k to a moving straight line L_k that has the fixed normal vector N_k and passes through the moving point $P(t_k)$. See Figure 3.

The TD error $e_{TD,k} = [(X_k - P(t_k)) \cdot N_k]^2$ becomes zero if the point X_k is contained in the line L_k. On the other hand, the line L_k is a relatively poor approximation to the curve $P(t)$ in a neighborhood of a high-curvature point $P(t_k)$ or if X_k is far from $P(t_k)$. Hence, in these cases, the point X_k may be poorly approximated by the curve $P(t)$ even if X_k is nearly on L_k, i.e. when the TD error $[(X_k - P(t_k)) \cdot N_k]^2$ is nearly zero (see Figure 3). This disparity between approximation quality and error measurement tends to cause the instability of TDM near a high-curvature part of the target shape, as will be illustrated later with our test examples. This unstable behavior of TDM is, in fact, the consequence of using an inappropriately large step size to solve a nonlinear optimization problem; indeed, we shall show that TDM is a Gauss-Newton method for solving a nonlinear least-squares problem and its excessively large step size is due to omission of important curvature related parts in the true Hessian.

Now let us consider the geometric interpretations of the PD error term and the TD error term. If $P(t_k)$ is fixed foot point on $P(t)$ of X_k, then both $e_{PD,k}$ and $e_{TD,k}$ give the same value of $d^2(P(t_k), X_k)$. However, for optimization purpose, we need to regard $d^2(P(t_k), X_k)$ as a function of the variable control points P_i, and in this sense $e_{PD,k}$ and $e_{TD,k}$ give very different approximations to $d^2(P(t_k), X_k)$.

If treating X_k as a free point, for any constant $c > 0$, the iso-value curve $e_{PD,k} \equiv ||X - P(t_k)||^2 = c$ of the PD error term is a circle (see Figure 1), and the iso-value curve $e_{TD,k} \equiv [(X_k - P(t_k) \cdot N_k]^2 = c$ of the TD error term is a pair of parallel lines, which can be regarded as a degenerate ellipse (see Figure 2). Since PDM has relatively slow convergence, and TDM tends to have fast but unstable convergence, one may speculate whether or not a new error term with elliptic iso-value curves can be devised to yield a more balanced performance between efficiency and stability. We shall see that such an error term is naturally provided by a second order approximation to the squared distance function.

2.2 Quadratic approximation to squared distance function

Given a curve or surface C, one may associate with it the squared distance function, which assigns to each point X the square of its shortest distance to C. It has been shown recently how to compute local quadratic approximants of the squared distance function, and how to use these approximants to solve a number of geometric optimization problems [26]. We now review briefly this work. Let O be the closest point on a second-order differentiable curve C to a fixed point X_0 (see Figure 5). Consider the local Frenet frame of C with its origin at O and its two coordinate axes being the tangent vector and the normal vector of the curve C at O. Let $\rho > 0$ be the curvature radius of C at O; we use an orientation of the curve normal such that $K = (0, \rho)^T$ is the curvature center of C at O. Let d be the signed distance from X_0 to O, i.e. $|d| = ||X_0 - O||$ with $d < 0$ if X_0 and K are on opposite sides of the curve C, and $d > 0$ if X_0 and K are on the same side of C. We note that there is always $d < \rho$ when $d > 0$, for otherwise O cannot be the closest point on the curve C to X_0. Consider a point $X = (x, y)^T$ in a neighborhood of X_0 in the local Frenet frame. Then the second order approximant of the squared distance from X to the curve C is [26]

$$f_0(x, y) = \frac{d}{d - \rho} x^2 + y^2. \tag{6}$$

Geometrically, the conic section $f_0(x, y) = d^2$ has second order contact with the offset curve of the target curve C that passes through X_0.

Since $f_0(x, y)$ in (6) is indefinite when $0 < d < \rho$, the unified expression

$$\hat{f}_0(x, y) = \frac{|d|}{|d| + \rho} x^2 + y^2 \tag{7}$$

is used in [27] as a positive semi-definite quadratic error term for solving geometric optimization problems. Note that, with this modification to (6) in the case of $0 \leq d < \rho$, $\hat{f}_0(x, y)$ is only a first order approximation to the squared distance function to C in a neighborhood of X_0.

The above approximant of squared distance is used in [27] for fitting a B-spline curve to a smooth target curve as follows. Given a target curve to be approximated and some initial B-spline curve $P(t)$ with control points P_i, a set of densely distributed points S_k, called *sensor points*, are first sampled on $P(t)$. Then the approximate squared distance $f_k(S_k)$ defined by (7) from each sensor

point S_k to the fixed target curve is computed. One next needs to determine unknown incremental updates D_i to the current control points P_i so that the B-spline curve with the updated control points $P_i + D_i$ approximates the target curve more closely. Since $f_k(S_k)$ is quadratic in S_k and each S_k is a linear combination of the control points $P_i + D_i$,

$$e_k \equiv f_k(S_k(P_1 + D_1, P_2 + D_2, \ldots, P_n + D_n))$$

is quadratic in the D_i. Supposing that the regularization term f_s is also quadratic in D_i, the objective function $f = \frac{1}{2}\sum_k e_k + \lambda f_s$ can then be minimized efficiently by solving a linear system of equations to determine the updates D_k. The above optimization is iterated to make the fitting curve $P(t)$ move towards the target curve. The squared distance is an appropriate error metric for shape approximation since it measures the distance from a sensor point to the target curve, rather than the distance from a sensor point to a particular point on the target curve, as in PDM. Note that the above curve fitting scheme is applicable to a smooth target curve but is unsuitable for a point cloud because of its difficulty in computing accurate tangent and curvature of a target shape defined by noisy or sparsely distributed data points.

3 Fitting a B-spline Curve to a Point Cloud Using SDM

In this section we introduce the *SD error term* defined by a quadratic approximant of squared distance function and outline its application to fitting a B-spline curve to a point cloud. We emphasize that *the new SD error term is defined by the squared distance function of the B-spline fitting curve, rather than that of the fixed target shape*. In other words, we measure the fitting error as defined in Eqn.(1), namely orthogonal to the fitting curve. This is different from the method used in [27] (or see Section 2.2), where errors are measured orthogonal to the fixed target curve and therefore also a different objective function is minimized. Our new SD error term leads to a new approach to fitting a B-spline curve to a point cloud. Because of its use of an approximant of the squared distance function, our proposed method will be referred to as *squared distance minimization* or *SDM*.

3.1 A new quadratic approximation to squared distance

Given a B-spline curve $P(t) = \sum_{i=1}^{m} B_i(t)P_i$ with control points $\mathcal{P} = (P_1, P_2, \ldots, P_m)$, let $\mathcal{D} = (D_1, D_2, .., D_m)$ be the variable incremental updates to give the new control points $\mathcal{P} + \mathcal{D}$. Let $P_D(t) = \sum_{i=1}^{m} B_i(t)(P_i + D_i)$ denote the B-spline curve with updated control points. For each data point X_k, define its corresponding parameter value t_k by finding the closest point $P(t_k)$, called the *foot point*, on the B-spline curve $P(t)$. Let d be the signed distance from X_k to $P(t_k)$, i.e. $|d| = ||P(t_k) - X_k||$, with the assumption on its sign being the same as made in Section 2.2. Let $\tilde{T}_k$ and $\tilde{N}_k$ be the unit tangent vector and the unit normal vector of the curve $P_D(t)$ at the point $P_D(t_k)$. Let $\tilde{\rho}$ denote the curvature radius of $P_D(t)$ at $P_D(t_k)$. With the fixed t_k, when the control points $\mathcal{P}$ are changed by $\mathcal{D}$, using the approximant obtained in [26] (see Section 2.2) expressed in

the global coordinate frame, it is easy to see that the second-order approximant of the squared distance from X_k to the curve $P_D(t)$ is

$$\tilde{h}_k(\mathcal{D}) = \frac{d}{d-\tilde{\rho}}[(P_D(t_k) - X_k)^T \tilde{T}_k]^2 + [(P_D(t_k) - X_k)^T \tilde{N}_k]^2. \tag{8}$$

The function $\tilde{h}_k(\mathcal{D})$ has, in general, a complicated expression in $\mathcal{D}$, since $P_D(t_k)$, $\tilde{T}_k$, $\tilde{N}_k$ and $\tilde{\rho}$ all depend on the variable control points $\mathcal{P} + \mathcal{D}$. (See Figure 6 for an illustration of $\tilde{h}_k(\mathcal{D})$). In order to obtain a quadratic error term that is positive semi-definite, some simplification has to be made to $\tilde{h}_k(\mathcal{D})$. Among several possibilities, balancing simplicity and accuracy, we adopt the scheme of letting only the point $P_D(t_k)$ vary with $\mathcal{D}$ and fixing $\tilde{T}_k$ and $\tilde{N}_k$ and $\tilde{\rho}$, i.e. as if $\mathcal{D} = 0$. This simplifies $\tilde{h}_k(\mathcal{D})$ into the quadratic function

$$h_k(\mathcal{D}) = \frac{d}{d-\rho}[(P_D(t_k) - X_k)^T T_k]^2 + [(P_D(t_k) - X_k)^T N_k]^2, \tag{9}$$

where T_k and N_k are respectively the unit tangent vector and the unit normal vector of the curve $P(t)$ at the point $P(t_k)$, and ρ is the curvature radius of $P(t)$ at $P(t_k)$. That is, T_k, N_k, and ρ do not vary with $\mathcal{D}$.

Note that $h_k(\mathcal{D})$ may take a negative value when $0 < d < \tilde{\rho}$. In order to obtain a positive semi-definite error term, based on $h_k(\mathcal{D})$, we define the SD error term as

$$e_{SD,k}(\mathcal{D}) = \begin{cases} \frac{d}{d-\rho}[(P_D(t_k) - X_k)^T T_k]^2 + [(P_D(t_k) - X_k)^T N_k]^2, & \text{if } d < 0, \\ [(P_D(t_k) - X_k)^T N_k]^2, & \text{if } 0 \le d < \rho, \end{cases} \tag{10}$$

Clearly, $e_{SD,k}(\mathcal{D})$ is a positive semi-definite quadratic function of $\mathcal{D}$ in all cases. Note that, in the case of $0 \le d < \rho$, $e_{SD,k}$ is only a first order approximation to the squared distance function.

When $d < 0$, the level-set curve of $e_{SD,k}(\mathcal{D}) = c$ is an ellipse centered at the point $P_D(t_k)$, if the X_k is treated as a variable point. When the control points P_i change, the ellipse is translated by keeping its center at $P_D(t_k)$ but with its shape, size and orientation remaining unchanged (see Figure 7). Note that the SD error term $e_{SD,k}$ becomes the TD error term $e_{TD,k}$ if $0 \le d < \rho$, i.e. when the data point X_k is sufficiently near $P(t_k)$ (relative to the magnitude of ρ) *and* X_k is on the convex side of the curve $P(t)$ (i.e. X_k and the curvature center K are on the same side of the curve $P(t)$). The use of the TD error term here will not cause instability, since in this case the tangent line is a relatively good approximation to the curve $P(t)$ in a neighborhood of $P(t_k)$.

3.2 Main steps of SDM

The main steps of the SDM method are as follows.

(1) Specify a proper initial shape of a B-spline fitting curve.

(2) Compute SD error terms for all data points to obtain a local approximation f_{SD} of the objective function f, defined by

$$f_{SD} = \frac{1}{2}\sum_{k} e_{SD,k} + \lambda f_s.$$

(3) Solve a linear system of equations to minimize f_{SD} to obtain an updated B-spline curve.

(4) Repeat steps 2 and 3 until convergence, e.g. until a pre-specified error threshold is satisfied or the incremental change of the control points falls below a preset threshold.

4 Experiments and Comparison

In this section we use some test examples to compare SDM with PDM and TDM for fitting a closed B-spline curve to unorganized data points in the plane. The quadrtic function to be optimized in each iteration has the form

$$f = \frac{1}{2}\sum_{k=1}^{n} e_k + \alpha F_1 + \beta F_2, \tag{11}$$

where e_k is defined by a particular error term (PD, TD, or SD) for data point X_k, and F_1 and F_2 are energy terms defined by

$$F_1 = \int \|P'(t)\|^2 \mathrm{d}t, \quad F_2 = \int \|P''(t)\|^2 \mathrm{d}t, \tag{12}$$

and α, $\beta \geq 0$ are constants. In our implementation F_1 and F_2 are integrated explicitly without numerical approximation.

For a fixed B-spline fitting curve $P(t)$, the Euclidean distance from data point X_k to $P(t)$ is denoted by d_k. Then, for evaluating the approximation error, we define the average error

$$Error\_Ave = \left[\frac{1}{n}\sum_{k=1}^{n} d_k^2\right]^{1/2},$$

and the maximum error

$$Error\_Max = \max_{k=1}^{n}\{d_k\}.$$

We present below the results of applying the three methods – PDM, TDM, and SDM – to fitting a cubic B-spline curve with uniform knots to several sets of unorganized data points. The same values of energy coefficients $\alpha = 0$ and $\beta = 0.001$ are used for all the examples in this section, unless specified otherwise. In some examples we use an initial shape that is quite different from the target shape in order to compare the abilities of different methods in converging to an acceptable minimum.

For some of the test examples shown below, we place the fitting curves generated in successive iterations at successive heights to form an *evolution surface* to show the evolution of an iterative optimization process under consideration (PDM or SDM). Data points or a subset of them are displayed at the top of an evolution surface. A striped texture is used to depict the trajectories of points of fixed parameter values on the fitting curve. The trajectories of control points are also shown. Log scale (base-10) is used for the height axis in these figures to accommodate for the large number of iterations needed by PDM. For size reference, a base square of size 2.2×2.2 is shown along with these evolution surfaces.

Example 1 Non-uniform data points on a circle. (Refer to Figure 8.) TDM and SDM converge with roughly the same speed, and both of them converge much faster than PDM does, as shown in Figures 8(e) and (f). The evolution surfaces in Figure 9 show that PDM takes about 100 iterations to reach the same small error values that is produced by SDM in less than 10 iterations. Close inspection on Figure 9(b) reveals slight rotation of the fitting curves generated by SDM, which is due to the non-uniqueness of a minimizer, since an optimal fitting curve is nearly invariant under rotation for data points on a circle.

Example 2 (Refer to Figure 10.) For this set of data points, SDM again converges much faster than PDM does, while TDM is trapped in a local minimum, producing a curve with self-intersection. The evolution surfaces formed by the curves generated by PDM and SDM are shown in Figure 11, viewed from two different directions.

Example 3 (Refer to Figure 12.) After three iterations, the fitting curve generated by SDM already reaches a much smaller error than the curve generated by PDM and TDM. Furthermore, TDM becomes divergent after 40 iterations. The evolution surfaces for PDM and SDM are shown in Figure 13.

Example 4 (Refer to Figure 14.) This set of data points is extremely noisy. After 50 iterations, SDM has already produced an acceptable result, but PDM is still converging slowly and TDM becomes unstable.

Example 5 (Refer to Figure 15.) The difficulty with this test lies in the corner points of the target shape and the highly non-uniform distribution of the control points of the initial B-spline curve. After 20 steps of iteration, PDM is trapped in an unacceptable local minimum and TDM becomes divergent, while SDM converges successfully. The most remarkable property of SDM demonstrated in this example is its strong tangential flow responsible for re-distributing clustered control points over the target shape to well approximate the four corner points.

Example 6 (Refer to Figure 16.) PDM is trapped in a local minimum and stops improving after four iterations (see Figure 13(a)). TDM does not converge for this data set. For SDM, the stable convergence begins to set in after 30 iterations. The radical reduction of the fitting error of SDM at iteration 30 is caused by a large tangential displacement of the control points, as shown by the evolution surface in Figure 13(b). The coefficients $\alpha = 0$ and $\beta = 0.00001$ are used in this example.

The six examples presented above are selected from numerous examples which we have experimented with. The following observations can be made from our experiments.

1) PDM exhibits the slowest convergence among the three methods, and is often trapped at a poor local minimum. Our experiments confirm the theoretical conclusion that PDM has, in general, only linear convergence. This is further explained in Section 6.

2) TDM demonstrates fast convergence when the target shape is not so noisy (i.e. representing a small-residue problem) and the fitting curve is relatively near the target shape (i.e. $|d| \leq \rho$), but often becomes unstable or even develops self-intersection in a high curvature region of the target shape or if the fitting curve is relatively far from the target shape. Increasing the value of the energy coefficient β in (11) may improve the stability of TDM, as well as the fairness of the fitting curve, but often at the expense having to accept a larger fitting error.

3) SDM exhibits much faster convergence than PDM does. The convergence of SDM is about as fast as that of TDM; moreover, SDM is much more stable than TDM, since TDM often does not converge at all for target shapes with shape features. This is mainly due to the fact that TDM is a Gauss-Newton method without step size control. We shall discuss this in more detail in Section 6.

4) The iso-value curves of the SD error term are ellipses aligned with the tangent at a point of the fitting curve. Therefore, at a low-curvature region of a B-spline fitting curve, the control points of the fitting curve, as well as points on the fitting curve, can flow in the tangential direction to attain a better distribution without causing much penalty from the SD error term. Meanwhile, as desired, such a flow is dampened at a high-curvature region due to the role played by the curvature radius ρ in the SDM error formula. As a comparison, tangential flow of control points is inhibited by the PD error term, causing stagnant improvement. In contrast, this tangential flow is checked nowhere by the TD error term, since the TD error term ignores curvature variation on the fitting curve, thus leading to unstable convergence in the presence of corner points in the target shape.

The ease of implementation and per-iteration computation time of SDM are nearly the same as those of PDM and TDM, since the three methods share the same framework but only with different quadratic error terms. The per-iteration computation time of SDM is mainly determined by the number of data points. The dominant part of computation time is the computation of foot points of all data points in each iteration. For example, for the set of 1,630 data points used in Example 4, computation of each iteration takes about 0.15 seconds on a PC with Pentium IV 2.4GHz CPU and 256 MB RAM, with over 95% of this time spent on foot point computation.

5 Implementation Issues

In this section we discuss the following implementation issues for facilitating the convergence or improving the computational efficiency of SDM: 1) initial shape specification; 2) insertion and

deletion of control points; 3) fast computation of error terms; and 4) adapting SDM to fit an open B-spline curve to data points.

5.1 Initial shape specification

Although SDM has better convergence behavior than PDM or TDM, it is still a scheme based on iterative local optimization. Therefore, there is no guarantee of convergence to a global optimum by using SDM. In fact, like PDM and TDM, the success of using SDM to obtain an optimal approximation depends on the choice of the initial B-spline curve. While a simple initial curve may work well for relatively simple target shapes, for a complex target shape, an initial shape roughly matching the target shape is necessary for stable convergence. Given a complex target shape, we first use a quad-tree cell partition to obtain a collection of connected cells of possibly different sizes that cover the data points (see Figure 18). Then a sequence of feature points of these covering cells are used as the control points of an initial B-spline curve.

Note that the union of all the covering cells should be topologically equivalent to the target curve. This is ensured by determining the cell sizes by the sampling density of the data points and the sizes of features in the target shape. Specifically, the cell sizes should be large enough to ensure cell connectivity when data sampling density ($1/d$ in Figure 18) is small, and they should be sufficiently small so that features of the target shape are preserved; for example, the gap of width f in Figure 18 must not be filled.

We assume that the target shape is a simply connected curve or an open curve without self-intersection; fitting target shapes with self-intersection is possible only if an appropriate initial shape is specified using more a priori knowledge about the target shape.

As an alternative to our approach, one could certainly also use the approach based on minimum spanning tree as in [18] to specify an initial B-spline curve.

5.2 Control point insertion and deletion

The number of control points of a fitting curve may be inadequate or redundant for achieving a pre-specified approximation accuracy. The inadequacy can be caused by the use of an initial shape that is too simple for the target shape, while the redundancy may be a problem when the initial polygon is generated by the procedure described in Section 5.1. When there are not enough control points, more control points need to be inserted to provide sufficient degree of freedom of a B-spline curve in order to achieve the desired approximation accuracy. When there are redundant control points, some control points need to be deleted in order to yield a compact B-spline fitting curve.

Control point insertion and deletion for a B-spline curve is studied in [38], though in a slightly different context. A method similar to the one in [38] is used in our implementation.

5.3 Fast setup of error terms

Efficient computation of foot points on the fitting curve of data points is important, especially in the case of a large number of data points, since an error term needs to be computed for each of these points in every iteration. We use the following speedup method consisting of two phases: *preprocessing* and *query*. In preprocessing we first sample a sufficient number of points on the fitting curve and compute the normal lines of the fitting curve at all the sample points. Then we record the intersections between these normal lines and all non-empty quad-tree cells generated in the preceding step of specifying an initial fitting curve (see Figure 19). In the query phase, for a data point, X_0 say, we find its covering cell and the two normal lines closest to X_0. Let the two normal lines be associated with parameter values t_1 and t_2 of the fitting curve. Let d_1 and d_2 denote the distances from X_0 to the two lines. Then a good estimate $P(\tilde{t}_0)$ of the foot-point of X_0 is given by the linear interpolation $\tilde{t}_0 = (d_2 t_1 + d_1 t_2)/(d_1 + d_2)$. The point $P(\tilde{t}_0)$ is then used as an initial point in a Newton-like iterative procedure to find the foot point $P(t_0)$ of X_0.

Figure 20 shows an example of using SDM and PDM to fit a B-spline curve to the contour of a Chinese character "Tian", meaning *sky*. In this example, the procedures described in subsections 5.1 to 5.3 are used. Again we see that, to achieve the same level of approximation error, PDM needs more control points and more iterations to produce a fitting curve than SDM does. In this example, PDM does not reach the same local minimum of SDM, explaining why different numbers of control points might be needed by PDM and SDM. Our test showed that TDM fails to converge for this example, because the font outline has a number of high curvature feature points.

5.4 Fitting an open B-spline curve

SDM can also be used to fit an open curve to a point cloud that represents an open target curve, with some necessary modifications to ensure that the endpoints of the fitting curve are properly determined. We assume that the target curve is not self-intersecting and that a proper initial shape of an open fitting curve is provided. The data points near an end of the target curve are called *target endpoints*. There are two cases to consider: *Case 1*: some data points cannot be projected to inner points of the fitting curve, and such points are called *outer data points* with respect to the fitting curve under consideration. *Case 2*: all data points can be projected to inner points of the fitting curve. In the first case, the error term associated with an outer data point is derived by interpolation of the SD error term and the PD error term. Specifically, referring to Figure 21, let T_0 be the unit tangent vector of the fitting curve $P(t)$ at its endpoint P_0. Let X_0 be an outer point such that P_0 is the closest point from the curve $P(t)$ to X_0. Let θ denote the angle between the tangent line of the curve $P(t)$ at P_0 and the vector $X_0 - P_0$, with $|\theta| < \pi/2$. Then the error term used $e_{outer,0}$ for X_0 is given by the following interpolation of the PD error term and SD error term,

$$e_{outer,0} = \cos\theta \, e_{PD,0} + (1 - \cos\theta) e_{SD,0}. \tag{13}$$

Here P_0 is regarded as a function of the control points. The rationale behind the interpolation in (13) is to use the PD error term partially for outer data points so that, through iterative optimization,

the endpoint P_0 of the fitting curve is pulled towards the target endpoints; of course, the SD error terms are still used for all other non-outer points. Note that the outer points in a target shape are identified relative to the current fitting curve; therefore we may have different data points as outer points in every iteration.

In the second case the initial fitting curve is longer than the target shape. In this we just use the standard SDM method — that is, use the SDM error term for each data point, to make the fitting curve to contract to fit the target shape. A non-zero but small value of α for the energy term F_1 in (12) may also be used to speed up the speed of contraction. We note that, when the initial shape is specified by the quad-tree partition approach in Sections 5.1, it suffices to just follow the procedure in case (1) to produce a satisfactory fitting curve.

We are going to present two examples of fitting open B-spline curves to data points, using the technique described above, in combination with SDM and PDM. The first example is shown in Figure 22. We note that, in this example, PDM takes about 600 iterations to reach the same approximation error that is achieved by SDM in 20 iterations. This is again due to the strong tangential flow of B-spline control points that is accommodated by the SDM error term. We note that TDM works well for this example as well.

The second example, shown in Figure 23, is an application to reconstructing a revolution surface from a point cloud scanned in by a laser range scanner, following a method proposed in [28]. The basic idea is as follows. First, the rotation axis of the revolution surface is estimated, and this axis is used to rotate the input data points in 3D (Figure 23(a)) into data points lying on a 2D plane (Figure 23(b)), from which the profile curve is to be reconstructed. Here we use SDM to fit an open B-spline curve to the 2D data points shown in Figure 23(b), and then use this B-spline curve to generate a revolution surface approximating the input 3D data points shown in Figures 23(c) and (d).

6 Discussion from the Viewpoint of Optimization

In this section we discuss SDM, as well as PDM and TDM, for B-spline curve approximation from the viewpoint of optimization. The B-spline fitting problem, as formulated in (1), can also be seen as the nonlinear optimization problem of minimizing

$$f = \frac{1}{2}\sum_{k=1}^{n} ||P(t_k) - X_k||^2 + \lambda f_s, \tag{14}$$

where $P(t_k)$ is a normal foot point of X_k, i.e.,

$$(P(t_k) - X_k)^T P_t'(t_k) = 0, \quad k = 1, 2, \ldots, n. \tag{15}$$

This viewpoint will be helpful in the computation of gradient and Hessian of the objective function f. For simplicity of discussion, in the following we will ignore the regularization term f_s; our conclusion is still applicable with f_s being taken into consideration, since f_s is independent of the

t_k and is quadratic in the P_i, assuming that λ is a fixed constant throughout all iterations. We note that the present problem of curve fitting is a *nonlinear least squares problem*.

We now explain why PDM is a variant of the steepest descent method, and therefore has a linear convergence rate. Given a planar B-spline fitting curve $P(t)$ with control points P_i and data points X_k, PDM minimizes the error function $f(\mathcal{P}, \mathcal{T})$ defined by (14), which is a function in the Euclidean space E^{2m+n} spanned by $\mathcal{P}$ and $\mathcal{T}$, where $\mathcal{P} = \{P_i\}_{i=1}^m$ are the control points of the fitting curve and $\mathcal{T} = \{t_k\}_{k=1}^n$ are the parameter values associated the data points X_k.

PDM has the following two steps that are carried out in each iteration (see Figure 24): (1) For fixed parameter values $\mathcal{T}_0 = \{t_{k,0}\}$ and current control points $\mathcal{P}_0 = \{P_{i,0}\}$, find new control points $\mathcal{P}_1 = \{P_{i,1}\}$ by minimization of the quadratic function $f(\mathcal{P}, \mathcal{T}_0)$. This is done by solving a linear system of equations; (2) Considering the control points $\mathcal{P}_1$ produced in step 1 as fixed, find new parameter values $\mathcal{T}_1 = \{t_{k,1}\}$ by minimization of the error function $f(\mathcal{P}_1, \mathcal{T})$. This is done by computing the foot points $P(t_k)$ of the data points X_k on the fitting curve $P(t)$ with the control points $\mathcal{P}_1$.

Due to the separate and alternate minimization of the $\mathcal{P}$ and $\mathcal{T}$ variables in each iteration, PDM is an *alternating method*, which is a typical optimization technique for solving a separable nonlinear least squares problem and is known to have only linear convergence [1, 33]. Figure 24 shows the zigzag behavior of PDM near a local minimum, which is reminiscent of the crawling behavior of the gradient descent method.

We shall investigate the standard algorithms for nonlinear least squares problems, namely Gauss-Newton iteration and the Levenberg-Marquart method [15], in connection with TDM. We shall show that Gauss-Newton based on variable projection [1] is exactly the same as TDM, which is used in [2]. From this we conclude on scenarios where TDM works well: small residual problems (data points are close to the solution curve) and a good initial position of the fitting curve; for a zero residual problem, optimization theory tells us that this method exhibits even quadratic convergence. Levenberg-Marquart is seen as a regularized version of TDM.

Our SDM scheme is finer than these standard methods (i.e. gradient descent and Gauss-Newton) for solving a nonlinear least squares problem. Although SDM is not a full Newton method because we do not compute the complete Hessian, it comes close to it; in SDM we approximate the complete Hessian by simply approximating the squared distance to a fitting curve, thus making SDM adaptable to local curvature variation.

In fact, all the methods above can be seen as gradient descent schemes in some metric. Whereas SDM chooses carefully the metric and TDM does this at least close to the target shape, PDM uses a metric which is not well adapted at all. Finally, we point to a global convergence improvement of all three methods, namely step size control with standard methods of optimization [15].

6.1 Methods based on gradient

The objective function in (14) can also be regarded as a function of the m control points $\mathcal{P} = (P_1, P_2, \ldots, P_m)$, i.e., a function $f: R^{2m} \to R$; the dependence of $\mathcal{T}$ on $\mathcal{P}$ is built in the constraints

in (15). For a fixed k, to indicate the dependence of t_k on $\mathcal{P}$, we write $t_k = t(\mathcal{P})$, omitting the subscript for simplicity. Denote $\mathcal{F} = P(t_k) - X_k$ and $f_k = \|\mathcal{F}\|$. Then the constraints (15) can be re-written, for each k, as

$$\mathcal{F}_t^T \mathcal{F} = 0. \tag{16}$$

Here and in the sequel we will denote partial derivatives with a subscript, e.g. $\mathcal{F}_t := \partial\mathcal{F}/\partial t$, $\mathcal{F}_{tt} := \partial^2\mathcal{F}/\partial t^2$.

We first compute the gradients of f_k^2 and f_k. Since $f_k^2 = \mathcal{F}^T\mathcal{F}$, by (16), we have

$$\nabla f_k^2 = \nabla(\mathcal{F}^T\mathcal{F}) = 2\left(\mathcal{F}_{\mathcal{P}}^T + \nabla t \mathcal{F}_t^T\right)\mathcal{F} = 2\mathcal{F}_{\mathcal{P}}^T\mathcal{F}. \tag{17}$$

Here, ∇t is the gradient of t_k with respect to $\mathcal{P}$, and $\mathcal{F}_{\mathcal{P}}$ is the matrix representing the derivative of $\mathcal{F}$ with respect to $\mathcal{P}$, not taking the dependency of t_k on $\mathcal{P}$ into account. Since $\nabla f_k^2 = 2 f_k \nabla f_k$, we obtain

$$\nabla f_k = \mathcal{F}_{\mathcal{P}}^T \frac{\mathcal{F}}{f_k} = \mathcal{F}_{\mathcal{P}}^T \frac{\mathcal{F}}{||\mathcal{F}||} = -\mathcal{F}_{\mathcal{P}}^T N_k, \tag{18}$$

where $N_k = -\mathcal{F}/||\mathcal{F}||$ is the unit normal vector of the curve $P(t)$ at $P(t_k)$. Then the gradient of f is found to be

$$\nabla f = \frac{1}{2}\sum_k \nabla f_k^2 = \left(\sum_{k=1}^n B_1(t_k)(P(t_k) - X_k), \ldots, \sum_{k=1}^n B_m(t_k)(P(t_k) - X_k)\right)^T.$$

Each component of the above gradient vector, i.e.

$$(\nabla f)_i = \sum_{k=1}^n B_i(t_k)(P(t_k) - X_k), \tag{19}$$

stands for a 2D vector associated with the i-th control point P_i, which is a weighted sum of the error vectors $P(t_k) - X_k$, where the weights are given by the i-th basis function B_i, evaluated at the parameter t_k of the foot point; of course, only error vectors in the support of B_i have influence.

At some places, it will be convenient to represent the B-spline curve in matrix form,

$$P(t) = B(t)\mathcal{P}.$$

Here, $B(t)$ is the $2 \times 2m$ matrix $(B_1(t)I_2, \ldots, B_m(t)I_2)$ with I_2 being 2×2 identity matrix. Now, the gradient vector $\nabla f \in R^{2m}$ can be written as

$$\nabla f = \sum_{k=1}^n B^T(t_k)(P(t_k) - X_k) = \sum_{k=1}^n B^T(t_k)B(t_k)\mathcal{P} - \sum_{k=1}^n B^T(t_k)X_k. \tag{20}$$

Moving with an appropriate step size s in the direction of the negative gradient means displacement of the control points via

$$P_{i,new} = P_i + s\sum_{k=1}^n B_i(t_k)(X_k - P(t_k)). \tag{21}$$

Updating the control points iteratively by (21) results in a gradient descent algorithm as the simplest solution to the present optimization problem. With a careful choice of the step size s, this algorithm converges linearly to a local minimum [15]. Even with another method with faster convergence being employed in the later phase, a few gradient-descent steps can be useful to improving a given initial position of a B-spline curve.

6.2 Gauss-Newton iteration and its variants

A Newton method minimizes the second-order approximant of the objective function at the current position x_c to obtain the next iterate x_+. To find this quadratic approximant for a nonlinear least squares problem with $f = \frac{1}{2}\sum_k f_k^2$, one needs to compute the Hessian of f, which is

$$\nabla^2 f = \sum_{k=1}^{n} \nabla f_k \cdot (\nabla f_k)^T + \sum_{k=1}^{n} f_k \nabla^2 f_k. \tag{22}$$

Since the computation of $\nabla^2 f_k$ is usually too costly, the Gauss-Newton method uses only the first part in (22) to approximate the Hessian $\nabla^2 f$. This is equivalent to computing the minimizer x_+ of the linear least squares problem

$$\min \frac{1}{2} \sum_k [f_k(x_c) + \nabla(f_k(x_c))^T \cdot (x - x_c)]^2.$$

That is, a linear approximation of f_k is used in the Gauss-Newton method.

In the problem of B-spline curve fitting, the step $x - x_c$ is given by the displacement vectors $\mathcal{D} = (D_1, \ldots, D_m)$ of the m control points. From Eqn. (18), we have

$$\nabla f_k = -\mathcal{F}_{\mathcal{P}}^T N_k = -B^T(t_k) N_k$$

Therefore Gauss-Newton iteration for B-spline curve fitting performs iterative minimization of

$$f_{GN} = \frac{1}{2} \sum_k [f_k - \sum_i B_i(t_k) D_i^T N_k]^2,$$

interleaved the step of foot point computation in order to satisfy the constraints (15). Since $N_k = (X_k - P(t_k))/\|P(t_k) - X_k\|$, we have $f_k = (X_k - P(t_k))^T N_k$. Noting that $P(t_k) = \sum_i B_i(t_k) P_i$, we obtain

$$\begin{aligned} f_{GN} &= \frac{1}{2} \sum_k [(X_k - P(t_k))^T N_k - \sum_i B_i(t_k) D_i^T N_k]^2 \\ &= \frac{1}{2} \sum_k [(X_k - \sum_i B_i(t_k)(P_i + D_i))^T N_k]^2 \qquad (23) \\ &= \frac{1}{2} \sum_k [(X_k - P_D(t_k))^T N_k]^2 = \frac{1}{2} \sum_k e_{TD,k}, \qquad (24) \end{aligned}$$

where $e_{TD,k}$ is defined in Eqn. (4). Hence, the Gauss-Newton method is equivalent to tangent distance minimization (TDM). Note that the Gauss-Newton method does not consider the change of tangent direction of the fitting curve. Moreover, the TD error term $e_{TD,k}$, unlike the SDM error term proposed in this paper, counts for neither the distance from the X_k to the curve $P(t)$ nor the curvature of the curve $P(t)$, reflecting the fact that the Gauss-Newton method omits the term $f_k \nabla^2 f_k$ in (22).

It is well-known [Kelley 1999, pp. 24] that, if x_c is sufficiently close to the minimizer x^* of f, the distance $\|e_c\| = \|x_c - x^*\|$ of the current iterate to x^* is related to the error $\|e_+\|$ in the next iterate by

$$\|e_+\| \leq K(\|e_c\|^2 + \|R(x^*)\| \, \|e_c\|), \tag{25}$$

where $R(x^*) = (f_1, \ldots, f_n)(x^*)$ is the residual at x^*, and K is a constant which involves the Jacobian of $R(x)$. It follows from (25) that for a zero residual problem Gauss-Newton iteration converges quadratically and the data points can be fitted exactly. Furthermore, Gauss-Newton iteration has fast convergence for good initial data and a small residual problem. For a large residual problem, the Gauss-Newton iteration may not converge at all.

Some variants of the Gauss-Newton method are possible. If only a scalar multiple of the Gauss-Newton step, $s(x_+ - x_c)$, usually with $0 < s < 1$, is used for stepping to the next solution, then one obtains the *damped Gauss-Newton method* [15].

Another way to modify Gauss-Newton is a regularization with the *Levenberg-Marquart method* [15], in which a scalar multiple of the unit matrix is added to the approximate Hessian. In our setting, this method requires the minimization of

$$f_{LM} = \frac{1}{2} \sum_k [(X_k - P_D(t_k))^T N_k]^2 + \nu_c \sum_i ||D_i||^2.$$

Thus, the regularization term penalizes large changes D_i in the control points. It can be shown that using a regularization parameter ν_c of the order of the norm of the residual, i.e. $O(\|R(x_c)\|)$, one obtains still quadratic convergence for a zero residual problem. A drawback of the Levenberg-Marquart method is that the same magnitude of regularization is applied to every control point, without taking into account the curvature variation at different locations.

By writing the fitting error as a function of the parameter values t_k, the Levenberg-Marquart method is used in [30] to iteratively update the t_k, and faster convergence of this method than a variant of PDM is reported. However, although the foot point computation is avoided, it is noted in [30] that this L-M method is about 10 times slower than PDM per iteration.

A Gauss-Newton method is implemented in [33] to update the control points P_i and the T_j together, therefore avoiding the costly step of computing foot points of data points. However, a relatively large linear system of equations needs to be solved, since now the parameter values of a large number of data points also enter optimization. The comparison of TDM with this *global* Gauss-Newton method, as well as other variants of the L-M method, is an interesting problem but is beyond the scope of this paper.

6.3 SDM – a quasi-Newton method

We shall derive the expression of the Newton method and then reveal the difference between our SDM scheme and the Newton method to show that SDM is, in fact, a quasi-Newton method. The key to this analysis is deriving a suitable expression of the Hessian of the objective function.

For a fixed k, consider at first the term $f_k^2 = \mathcal{F}^T\mathcal{F}$. By (17), we have $\nabla f_k^2 = 2\mathcal{F}_{\mathcal{P}}^T\mathcal{F}$. The derivative of ∇f_k^2 yields the Hessian

$$\begin{aligned}\nabla^2 f_k^2 &= 2[\left(\mathcal{F}_{\mathcal{P}\mathcal{P}} + \mathcal{F}_{\mathcal{P}t}\nabla t^T\right)^T \mathcal{F} + \mathcal{F}_{\mathcal{P}}^T\left(\mathcal{F}_{\mathcal{P}} + \mathcal{F}_t\nabla t^T\right)] \\ &= 2[\nabla t\mathcal{F}_{\mathcal{P}t}^T\mathcal{F} + \mathcal{F}_{\mathcal{P}}^T\mathcal{F}_{\mathcal{P}} + \mathcal{F}_{\mathcal{P}}^T\mathcal{F}_t\nabla t^T]. \end{aligned} \tag{26}$$

Here we used $\mathcal{F}_{\mathcal{P}\mathcal{P}} = 0$, since $\mathcal{F}$ is linear in $\mathcal{P}$. Again ∇t is the gradient of $t_k = t(\mathcal{P})$ with respect to $\mathcal{P}$.

On the other hand, we need to find the relationship between ∇t and $\mathcal{F}_{\mathcal{P}}$. Differentiating the constraint (16), we obtain

$$\left(\mathcal{F}_{\mathcal{P}t}^T + \nabla t\mathcal{F}_{tt}^T\right)\mathcal{F} + \left(\mathcal{F}_{\mathcal{P}}^T + \nabla t\mathcal{F}_t^T\right)\mathcal{F}_t = 0. \tag{27}$$

Solving for ∇t yields

$$\nabla t = -\frac{\mathcal{F}_{\mathcal{P}t}^T\mathcal{F} + \mathcal{F}_{\mathcal{P}}^T\mathcal{F}_t}{\mathcal{F}_{tt}^T\mathcal{F} + \mathcal{F}_t^T\mathcal{F}_t}.$$

Substituting this expression of ∇t in (26), we can obtain the complete Hessian $\nabla^2 f_k^2$.

We will now make a simplification and neglect the term $\mathcal{F}_{\mathcal{P}t}^T\mathcal{F}$, i.e. set it to zero. This results in an approximate Hessian $\widetilde{\nabla}^2 f_k^2$. To interpret this approximate Hessian geometrically, we let s denote the arc length parameter of the B-spline curve $P(t)$. Then we have

$$\begin{aligned}\frac{1}{2}\widetilde{\nabla}^2 f_k^2 &= \mathcal{F}_{\mathcal{P}}^T\mathcal{F}_{\mathcal{P}} - \frac{\mathcal{F}_{\mathcal{P}}^T\mathcal{F}_t\mathcal{F}_t^T\mathcal{F}_{\mathcal{P}}}{\mathcal{F}_{tt}^T\mathcal{F} + \mathcal{F}_t^T\mathcal{F}_t} \\ &= \mathcal{F}_{\mathcal{P}}^T\mathcal{F}_{\mathcal{P}} - \frac{(s_t)^2\mathcal{F}_{\mathcal{P}}^T\mathcal{F}_s\mathcal{F}_s^T\mathcal{F}_{\mathcal{P}}}{[\mathcal{F}_{ss}^T(s_t)^2 + \mathcal{F}_s^T s_{tt}]\mathcal{F} + \mathcal{F}_s^T\mathcal{F}_s(s_t)^2} \\ &= \mathcal{F}_{\mathcal{P}}^T\mathcal{F}_{\mathcal{P}} - \frac{(s_t)^2\mathcal{F}_{\mathcal{P}}^T\mathcal{F}_s\mathcal{F}_s^T\mathcal{F}_{\mathcal{P}}}{\mathcal{F}_{ss}^T\mathcal{F}(s_t)^2 + \mathcal{F}_s^T\mathcal{F}_s(s_t)^2} = \mathcal{F}_{\mathcal{P}}^T\mathcal{F}_{\mathcal{P}} - \frac{\mathcal{F}_{\mathcal{P}}^T\mathcal{F}_s\mathcal{F}_s^T\mathcal{F}_{\mathcal{P}}}{(\mathcal{F}_{ss}^T\mathcal{F} + \mathcal{F}_s^T\mathcal{F}_s)}. \end{aligned}$$

In the above, the term $\mathcal{F}_s^T\mathcal{F}s_{tt}$ drops out due to the constraint (16) and the fact that $\mathcal{F}_s^T$ and $\mathcal{F}_t^T$ are collinear.

Clearly, $\mathcal{F}_s^T\mathcal{F}_s = 1$, since $\mathcal{F}_s = T_k$ is the unit tangent vector of $P(t)$. Also, $\mathcal{F}_{ss} = \kappa N_k$ is the curvature vector of $P(t)$ at $P(t_k)$, where N_k is the unit normal vector of $P(t)$ at $P(t_k)$. Therefore $\mathcal{F}_{ss}^T\mathcal{F} = -d\kappa$, where d is defined in Section 3, since, by the constraint (16), $\mathcal{F} = P(t_k) - X_k$ is in the direction of the unit normal vector. Hence, we obtain

$$\frac{1}{2}\widetilde{\nabla}^2 f_k^2 = \mathcal{F}_{\mathcal{P}}^T\mathcal{F}_{\mathcal{P}} - \frac{\mathcal{F}_{\mathcal{P}}^T T_k T_k^T\mathcal{F}_{\mathcal{P}}}{-d\kappa + 1}. \tag{28}$$

This equation is further rewritten as

$$\begin{aligned}\frac{1}{2}\widetilde{\nabla}^2 f_k^2 &= \mathcal{F}_{\mathcal{P}}^T\left(I - T_k T_k^T\right)\mathcal{F}_{\mathcal{P}} - \frac{d\kappa \mathcal{F}_{\mathcal{P}}^T T_k T_k^T \mathcal{F}_{\mathcal{P}}}{-d\kappa + 1} \\ &= \mathcal{F}_{\mathcal{P}}^T N_k N_k^T \mathcal{F}_{\mathcal{P}} + \frac{d}{d-\rho}\mathcal{F}_{\mathcal{P}}^T T_k T_k^T \mathcal{F}_{\mathcal{P}}. \end{aligned} \tag{29}$$

Now we consider the relationship between SDM and the quasi-Newton method obtained above by replacing the Hessian $\nabla^2 f_k^2$ by $\widetilde{\nabla}^2 f_k^2$. Note that

$$\mathcal{F}_{\mathcal{P}}^T N_k N_k^T \mathcal{F}_{\mathcal{P}} = \nabla f_k (\nabla f_k)^T,$$

which is the first term in (22) that is used by Gauss-Newton iteration to approximate the true Hessian. Thus, replacing the Hessian $\nabla^2 f_k^2$ by $\widetilde{\nabla}^2 f_k^2$ in the Newton method is equivalent to adding the second term in (29) to the Gauss-Newton method to yield a quasi-Newton method. Recall that the Gauss-Newton method is the same as TDM. Therefore, noting that $(P(t_k) - X_k)^T T_k = 0$ (by (15)), the above quasi-Newton method minimizes the quadratic function

$$\begin{aligned} f_{QN} &= f_{GN} + \frac{1}{2}\sum_{k=1}^{n}\frac{d}{d-\rho}[(\sum_i B_i(t_k)D_i)^T T_k]^2 \\ &= \frac{1}{2}\sum_{k=1}^{n}\left\{[(P_D(t_k) - X_k)^T N_k]^2 + \frac{d}{d-\rho}[(P_D(t_k) - P(t_k))^T T_k]^2\right\} \\ &= \frac{1}{2}\sum_{k=1}^{n}\left\{[(P_D(t_k) - X_k)^T N_k]^2 + \frac{d}{d-\rho}[(P_D(t_k) - X_k)^T T_k]^2\right\} \\ &= \frac{1}{2}\sum_{k=1}^{n} h_k(\mathcal{D}), \end{aligned}$$

where $h_k(\mathcal{D})$ is defined in (9). Hence, the quasi-Newton method given by f_{QN} above is exactly the SDM scheme before replacing indefinite quadratic forms by semi-definite ones, i.e. the SD error term defined in (10). Hence, we have shown that the SDM method is a quasi-Newton method obtained by discarding the term $\mathcal{F}^T\mathcal{F}_{\mathcal{P}t}$, which amounts to disregarding the change $\mathcal{F}_{\mathcal{P}t}$ of the tangent vector $P_t'(t_k)$ caused by the change of the control points.

SDM does not fall into the category of the quasi-Newton methods that fulfill the so-called secant equation [15]. Instead, SDM uses another positive definite approximant of the Hessian, based on geometric considerations. Although SDM is not a standard optimization procedure, it is a computationally attractive and effective compromise between a full Newton scheme and Gauss-Newton – it picks up more contributions of the true Hessian than Gauss-Newton does, but it ignores the remaining part for reasons of computational efficiency and simplicity. Indeed, SDM is an optimization scheme that is particularly suited for solving shape fitting problems, because SDM uses an error metric that is adaptable to local curvature variation of a target shape.

6.4 Another view about PDM, TDM and SDM

It is a simplified view to regard the gradient of $f: R^{2m} \to R$ as the vector formed by the partial derivatives of f with respect to the $2m$ variables in $\mathcal{P}$. We consider a point $\mathcal{P} \in R^{2m}$, and a unit vector $\mathcal{D}$ attached to it. Then, the directional derivative of f with respect to $\mathcal{D}$ is $\nabla f(\mathcal{P})^T \mathcal{D}$. Hence, the direction $\mathcal{D} = \nabla f / \|\nabla f\|$ has the largest directional derivative, and the magnitude of that derivative is $\|\nabla f\|$. These two properties can be used as a characterization of the gradient.

We see that this interpretation of the gradient depends on the metric. So far we have used the canonical Euclidean metric, $\|\mathcal{X}\|^2 = \mathcal{X}^T \mathcal{X}$, $\mathcal{X} \in R^{2m}$. With any positive definite matrix M, a more general metric can be defined as

$$\|\mathcal{X}\|^2 = \mathcal{X}^T M \mathcal{X}. \tag{30}$$

The gradient $\nabla_M f$ with respect to this metric is related to the canonical gradient ∇f by

$$\nabla_M f = M^{-1} \nabla f. \tag{31}$$

In fact, M may even depend on $\mathcal{X}$; then we have a Riemannian metric.

In the notation used above for our B-spline fitting problem, a gradient descent algorithm with respect to the gradient $\nabla_M f$ and step size s is formulated as

$$\mathcal{P}_D = \mathcal{P} - s M^{-1} \nabla f(\mathcal{P}). \tag{32}$$

All algorithms discussed above fall into this scheme. An ordinary gradient descent results from $M = I$. The Newton method defines the metric with the Hessian $M = \nabla^2 f(\mathcal{P})$, supposing that M is positive definite. Our SDM scheme uses the approximate Hessian $M = \sum \widetilde{\nabla}^2 f_k^2$, caused by omitting the differential change of the tangent vector $P_t'(t_k)$ and making the approximate Hessian positive semi-definite. Of course, TDM also belongs to this class, but with even more terms from the true Hessian being discarded, since it does not include any second order derivative.

Finally, we can interpret PDM as follows. For any B-spline curve $P(t) = B(t)\mathcal{P}$ associated with the vector $\mathcal{P} \in R^{2m}$, one computes the parameter values t_k of the foot points of the data points X_k. With help of these parameters, one defines a local distance measure between the curve $P(t)$ and another B-spline curve $Q(t)$ by

$$d^2(P(t), Q(t)) := \sum_{k=1}^{n} (P(t_k) - Q(t_k))^2 = (\mathcal{P} - \mathcal{Q})^T \sum_{k=1}^{n} B(t_k)^T B(t_k) (\mathcal{P} - \mathcal{Q}). \tag{33}$$

Here $\mathcal{D} := \mathcal{P} - \mathcal{Q}$ can be considered as a first order displacement of $P(t)$. Thus, this introduces a local metric in R^{2m} represented the matrix

$$M := \sum_{k=1}^{n} B(t_k)^T B(t_k).$$

We now show that PDM is precisely a gradient descent method with respect to this Riemannian metric in R^{2m}. Given the current curve $P(t)$, the next curve $P_D(t) = B(t)\mathcal{P}_D$ is given by minimization of

$$\sum_{k=1}^{n}(P_D(t_k) - X_k)^T(P_D(t_k) - X_k) = \mathcal{P}_D^T M \mathcal{P}_D - 2\sum_{k=1}^{n} X_k^T B(t_k)\mathcal{P}_D + \sum_{k=1}^{n} X_k^T X_k.$$

The minimizer is found to be

$$\mathcal{P}_D = M^{-1}\sum_{k=1}^{n} B(t_k)^T X_k. \tag{34}$$

With the gradient ∇f from equation (20), we see immediately that

$$\mathcal{P}_D = \mathcal{P} - M^{-1}\nabla f(\mathcal{P}),$$

showing that PDM is a gradient descent method with respect to M and a full step $s = 1$.

6.5 Step size control

We have tested step size control on PDM, TDM, and SDM, using the Armijo rule [15]. It is found that step size control does not help much with PDM and SDM — PDM still converges slowly, and the per-iteration computation of SDM becomes much longer with moderate degree of improvement in stability. It is found that the stability of TDM improves greatly with the help of step size control, however, at the cost of much longer per–iteration time, especially when approaching a local minimum, since, due to the "flat" gradient near a local minimum, it normally gets more time-consuming to select an appropriate step size via repetitive evaluations of the fitting error.

Figure 25 shows the result of applying step size control (the Armijo rule) to TDM on the same data points and initial B-spline curves as shown in Figures 10(a) and 15(a); now both data sets are satisfactorily approximated by TDM. For comparison, refer to Figures 10(c) and 15(c) to see the unacceptable fitting curves generated by TDM without step size control.

7 Concluding Remarks

PDM is so far a dominant method used in the practice of parametric curve and surface fitting [4]. As we have shown, PDM has linear convergence in theory, converges slowly in practice, and is often trapped in a poor local minimum. TDM is another existing method used for curve fitting in computer vision (e.g. [2]). TDM converges faster than PDM, but its convergence is highly unstable. Against this backdrop, we have proposed a novel and efficient method, called SDM, for fitting B-spline curves to point cloud data. We have shown that SDM converges much faster than PDM and that SDM is much more stable than TDM. In addition, SDM is easy to implement and has similar per-iteration computation time as PDM and TDM, since they share the same framework. All this suggests that SDM is a favorable alternative to PDM or TDM for B-spline curve fitting.

In order to gain a better understanding of the above optimization methods, we have also studied the B-spline curve fitting problem from the optimization viewpoint. We note that PDM is a gradient descent method in a metric that is not well chosen. We have also shown that TDM is exactly a Gauss-Newton method for solving a nonlinear least squares problem, and its instability at high curvature regions is thus due to its omission of important parts in the true Hessian of the objective function and the lack of step size control. Finally, we have shown that our proposed SDM scheme is a quasi-Newton method using a carefully chosen approximate Hessian, and thus its superior performance in both convergence and stability does not come as a surprise. Interestingly, unlike most other quasi-Newton methods, the approximate Hessian used by SDM is not explicitly computed; it arises naturally as the consequence of using the simple SDM error term devised out of entirely geometric considerations, i.e. making use of curvature information to give a close approximation of the squared distance function. This contributes to the simplicity and efficiency of SDM.

We expect to see more studies on SDM and related problems, both theoretically and from the viewpoint of applications. An immediate extension is to apply SDM to optimize the weights and knots of a NURBS fitting curve, an issued addressed in [6, 16]. Other significant problems include the analysis of the convergence rate of SDM (i.e. to show if SDM has superlinear convergence) and the improvement of the global convergence to the target shape from a very simple "seed" shape; the latter topic has a close connection to the work on active contours [14, 21, 22, 32].

Our ongoing research shows that SDM can be applied to a large class of shape reconstruction and geometric optimization problems, such as fitting B-spline surfaces or subdivision surfaces, optimization over an analytical surface or a mesh surface, and surface registration. This extension of SDM to the surface case would be of great practical importance, in view of the wide application of the inefficient PDM to surface fitting in graphics and CAD.

References

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

[2] Andrew Blake and Michael Isard. *Active Contours*. Springer, New York, 1998.

[3] M. Djebali, M. Melkemi, and N. Sapidis. Range-image segmentation and model reconstruction based on a fit-and-merge strategy. In *Proceedings of the seventh ACM symposium on Solid Modeling and Applications*, pages 127–138, 2002.

[4] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press, New York, 4th edition, 1997.

[5] David R. Forsey and Richard H. Bartels. Surface fitting with hierarchical splines. *ACM Transactions on Graphics*, 14:134–161, 1995.

[6] Rony Goldenthal and Michel Bercovier. Spline curve approximation and design by optimal control over the knots. *Computing*, 72:53–64, 2004.

[7] A. Ardeshir Goshtasby. Grouping and parameterizing irregularly spaced points for curve fitting. *ACM Transactions on Graphics*, 19:185–203, 2000.

[8] Andreas Kolb Günther Greiner and Angela Riepl. Scattered data interpolation using data dependant optimization techniques. *Graphical Models*, 64:1–18, 2002.

[9] Jörg Haber, Frank Zeilfelder, Oleg Davydov, and Hans Peter Seidel. Smooth approximation and rendering of large scattered data sets. In *Proceedings of the conference on Visualization '01*, pages 341–348, 2001.

[10] Hugues Hoppe. Progressive meshes. In *Proceedings of SIGGRAPH'96*, pages 99–108, 1996.

[11] Hugues Hoppe, Tony DeRose, Tom Duchamp, Mark Halstead, Hubert Jin, John McDonald, Jean Schweitzer, and Werner Stuetzle. Piecewise smooth surface reconstruction. In *Proceedings of SIGGRAPH'94*, pages 295–302, 1994.

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

[13] Josef Hoschek and Dieter Lasser. *Fundamentals of Computer Aided Geometric Design*. AK Peters, 1993.

[14] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: active contour models. *International Journal of Computer Vision*, 1:321–331, 1988.

[15] C. T. Kelley. *Iterative Methods for Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, 1999.

[16] Pascal Laurent-Gengoux and Mounib Mekhilef. Optimization of a NURBS representation. *Computer-Aided Design*, 25:699–710, 1993.

[17] E. T. Y. Lee. Choosing nodes in parametric curve interpolation. *Computer-Aided Design*, 21:363–370, 1989.

[18] In-Kwon Lee. Curve reconstruction from unorganized points. *Computer Aided Geometric Design*, 16:161–177, 1999.

[19] W. Y. Ma and J. P. Kruth. Parameterization of randomly measured points for least squares fitting of B-spline curves and surfaces. *Computer-Aided Design*, 27:663–675, 1995.

[20] I. Maekawa and K.H. Ko. Surface construction by fitting unorganized curves. *Graphical Models*, 64:316–332, 2002.

[21] R. Malladi, J. Sethian, and B. C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:158–175, 1995.

[22] S. Osher and Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, New York, 2003.

[23] Theodosios Pavlidis. Curve fitting with conic splines. *ACM Transactions on Graphics*, 2:1–31, 1983.

[24] Les Piegl and Wayne Tiller. *The NURBS book*. Springer, New York, 2nd edition, 1997.

[25] Michael Plass and Maureen Stone. Curve-fitting with piecewise parametric cubics. *Computer Graphics*, 17(3):229–239, 1983.

[26] H. Pottmann and M. Hofer. Geometry of the squared distance function to curves and surfaces. In H.C. Hege and K. Polthier, editors, *Visualization and Mathematics III*, pages 223–244. 2003.

[27] H. Pottmann, S. Leopoldseder, and M. Hofer. Approximation with active B-spline curves and surfaces. In *Proceedings of Pacific Graphics 2002*, pages 8–25. IEEE Computer Society Press, 2002.

[28] H. Pottmann and J. Wallner. *Computational Line Geometry*. Springer-Verlag, Berlin, 2001.

[29] Vaughan Pratt. Techniques for conic splines. In *Proceedings of SIGGRAPH'85*, pages 151–160, 1985.

[30] Biplab Sarkar and Chia-Hsiang Menq. Parameter optimization in approximating curves and surfaces to measurement data. *Computer Aided Geometric Design*, 8:267–290, 1991.

[31] Eric Saux and Marc Daniel. An improved Hoschek intrinsic parametrization. *Computer Aided Geometric Design*, 20:513–521, 2003.

[32] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[33] T. Speer, M. Kuppe, and J. Hoschek. Global reparameterization for curve approximation. *Computer Aided Geometric Design*, 15:869–877, 1998.

[34] Gabriel Taubin. Dual mesh resampling. *Graphical Models*, 64:94–113, 2002.

[35] D. J. Walton and R. Xu. Turning point preserving planar interpolation. *ACM Transactions on Graphics*, 10:297 – 311, 1991.

[36] Xiaohuan Corina Wang and Cary Phillips. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 129 – 138, 2002.

[37] V. Weiss, L. Andor, G. Renner, and T. Varady. Advanced surface fitting techniques. *Computer Aided Geometric Design*, 19:19–42, 2002.

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

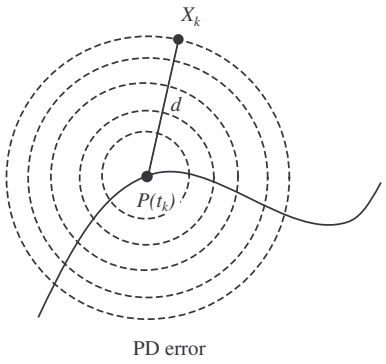


Figure 1: Iso-values curves of the point distance (PD) error term.

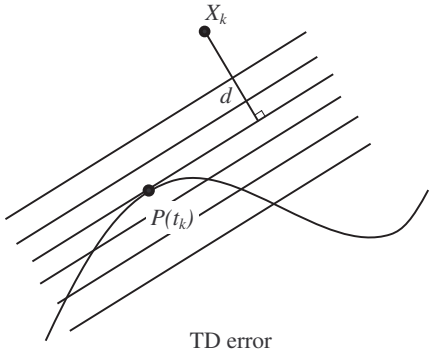


Figure 2: Iso-values curves of the tangent distance (TD) error term.

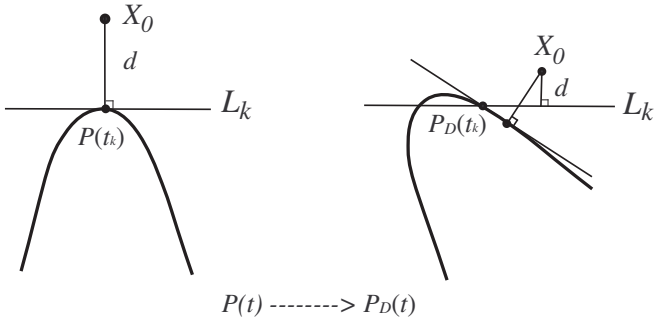


Figure 3: In a neighborhood of a high curvature point, the true approximation error can be big even though the TD error is small.

(a) Data points on a circle with an initial B-spline spline curve.

(b) The fitting curve generated by PDM in 50 iterations.

(c) The fitting curve generated by SDM in 3 iterations.

(d) The fitting curve generated by SDM in 10 iterations.

Figure 4: Comparison of PDM and SDM for sparse data points on a circle (Number of data points: 32).

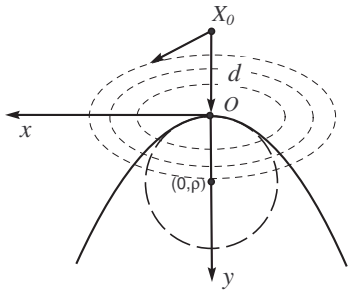


Figure 5: Quadratic approximant of the squared distance function to the curve C at X_0.

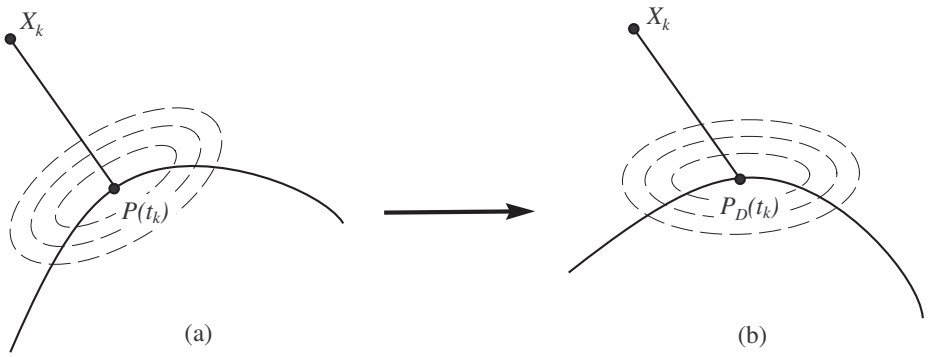


Figure 6: The approximate squared distance function $h^+(\mathbf{D})$ defined by (8) is shown via its iso-value curves in the case of $d < 0$. (a) Before updating the control points $\mathbf{P}$. (b) After updating the control points $\mathbf{P}$.

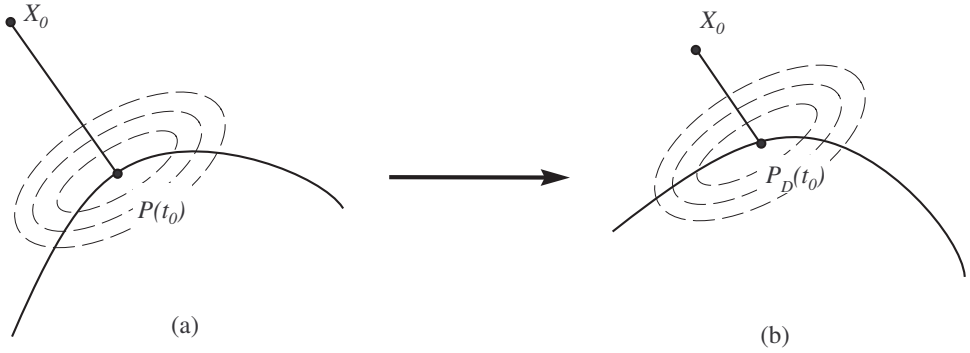
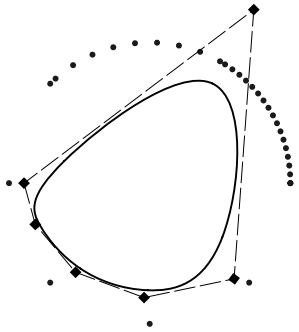
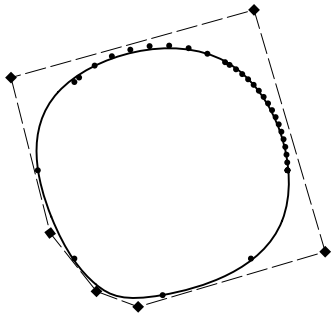


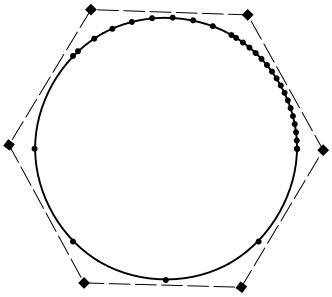
Figure 7: The SD error term $e_{SD,k}(\mathbf{D})$ defined by (10) is shown via its iso-value curves in the case of $d < 0$. (a) Before updating the control points $\mathbf{P}$. (b) The translation of $e_{SD,k}(\mathbf{D})$ after updating $\mathbf{P}$.



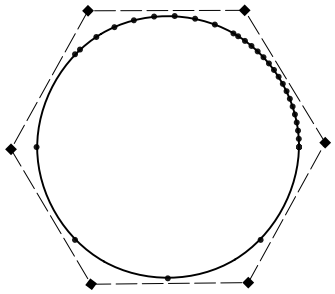
(a) Data points on a circle and an initial B-spline curve.



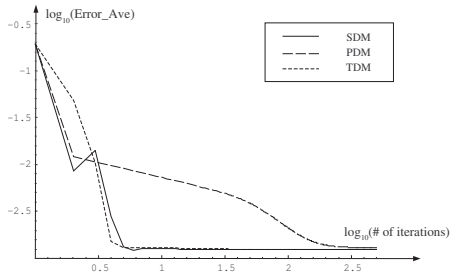
(b) The fitting curve generated by PDM in 10 iterations.



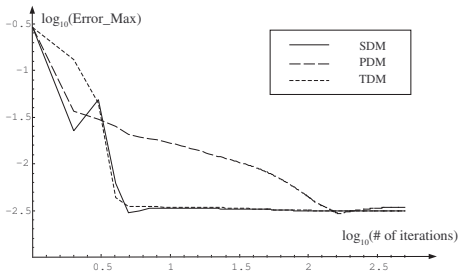
(c) The fitting curve generated by TDM in 10 iterations.



(d) The fitting curve generated by SDM in 10 iterations.

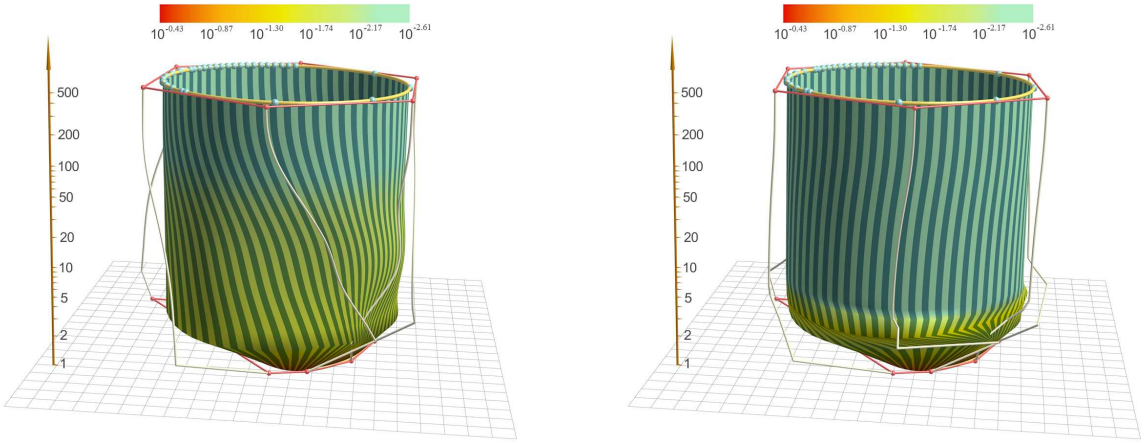


(e) The average error versus the number of iterations of the three methods.



(f) The maximum error versus the number of iterations of the three methods.

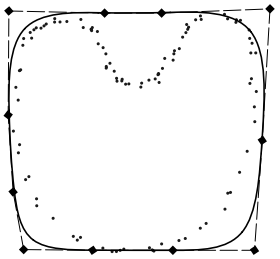
Figure 8: (Example 1) Comparison of the three methods on sparse data points a circle (Number of data points: 32). Log scale (base 10) is used for the error axis. In this figure the iteration axes in Figures (e) and (f) use log scale (base 10) in order to distinguish the error curves of TDM and SDM.



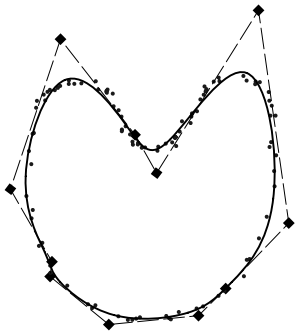
(a) The evolution surface of PDM.

(b) The evolution surface of SDM.

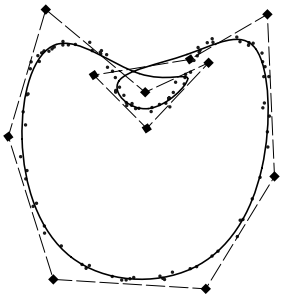
Figure 9: The evolution surfaces of PDM and SDM for the data in Example 1 (Figure 8). Log scale is used for the height axis to accommodate for the large number of iteration needed by PDM. The base square has the size 2.2×2.2.



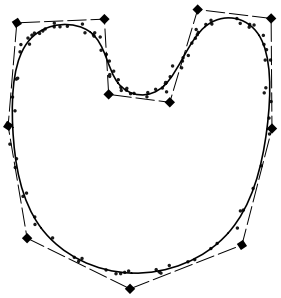
(a) Unorganized data points on a "C" shape and an initial B-spline curve.



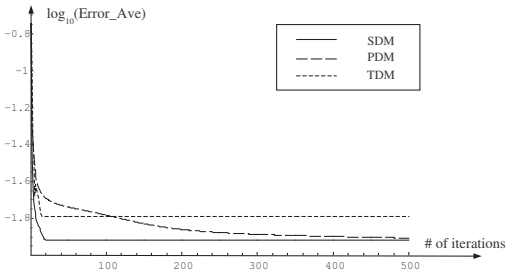
(b) The fitting curve generated by PDM in 20 iterations.



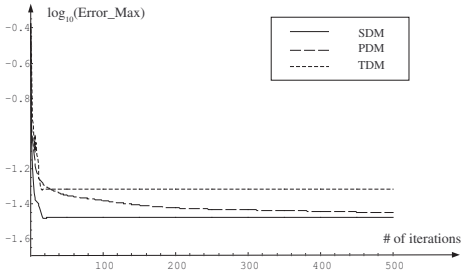
(c) The fitting curve generated by TDM in 20 iterations.



(d) The fitting curve generated by SDM in 20 iterations.

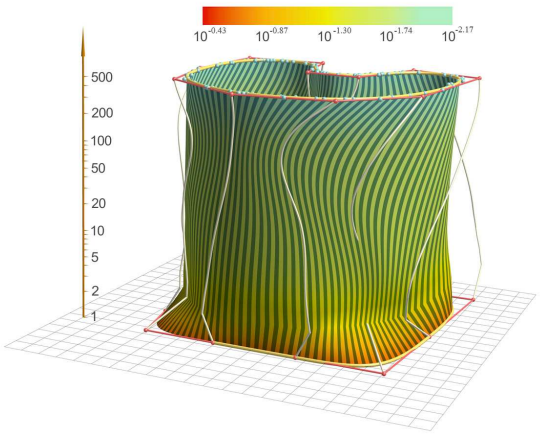


(e) The average error versus the number of iterations of the three methods.

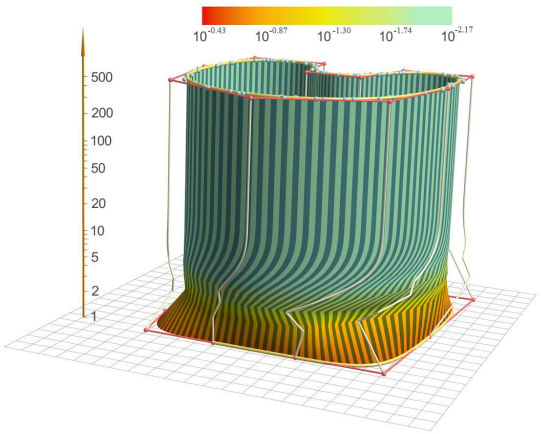


(f) The maximum error versus the number of iterations of the three methods.

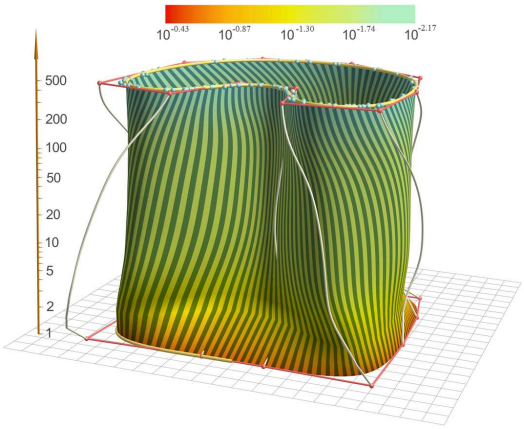
Figure 10: (Example 2) Comparison of the three methods. Log scale (base 10) is used for the error axis. SDM converges faster than PDM does. TDM is trapped in a local minimum with self-intersection in the fitting curve. Number of data points: 102.



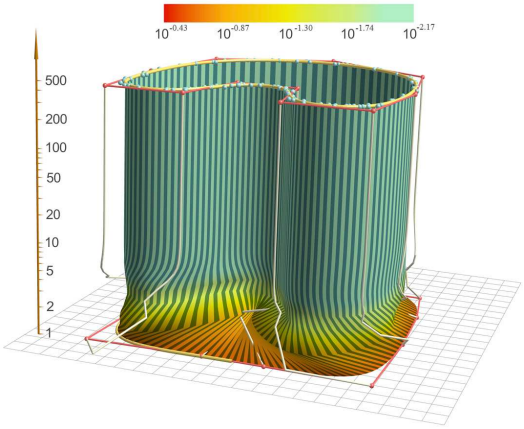
(a) The evolution surface of PDM – the same view angle as in (b).



(b) The evolution surface of SDM – the same view angle as in (a).

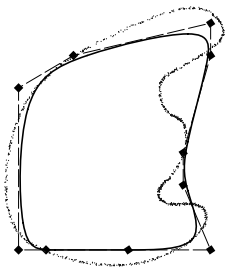


(c) The evolution surface of PDM – the same view angle as in (d).

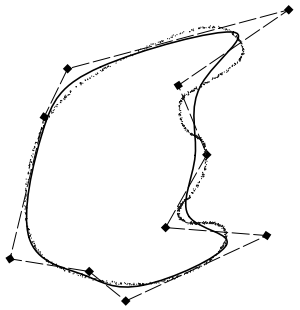


(d) The evolution surface of SDM – the same view angle as in (c).

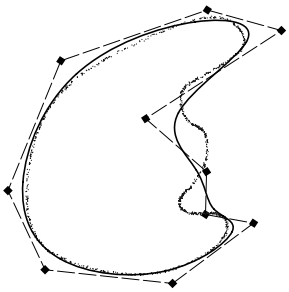
Figure 11: The evolution surfaces of PDM and SDM for the data in Example 2 (Figure 10). Two different views are shown for each surface.



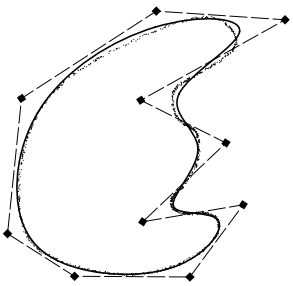
(a) A closed target curve of smooth shape and an initial B-spline curve.



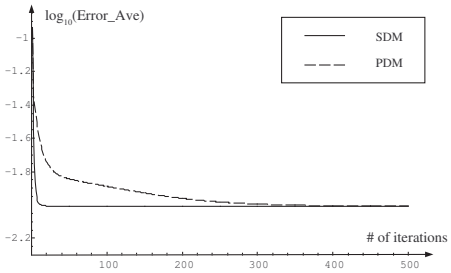
(b) The fitting curve generated by PDM in 3 iterations.



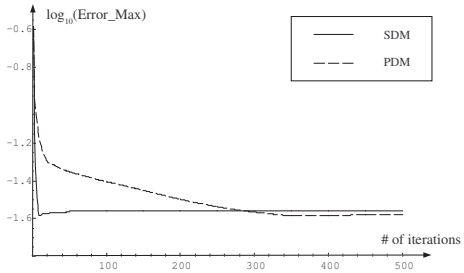
(c) The fitting curve generated by TDM in 3 iterations.



(d) The fitting curve generated by SDM in 3 iterations.

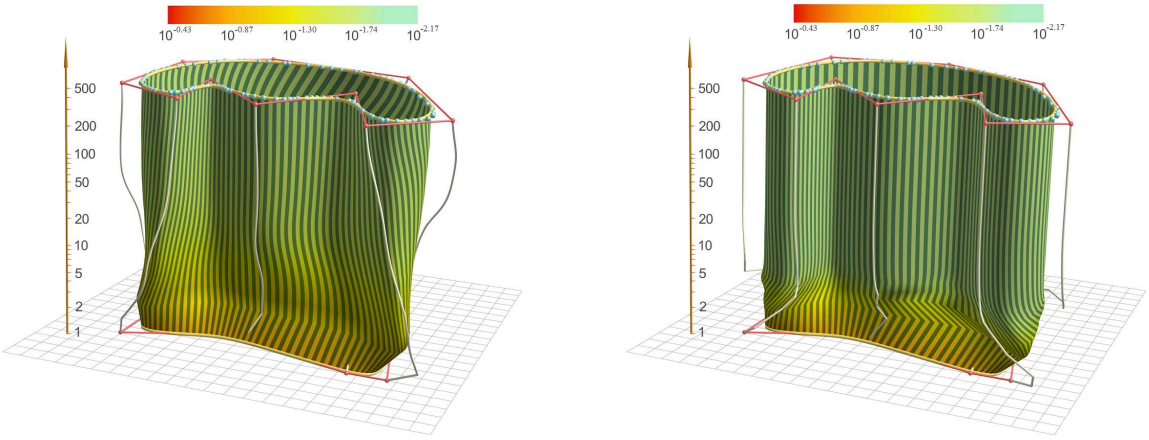


(e) The average error versus the number of iterations of PDM and SDM.



(f) The maximum error versus the number of iterations of PDM and SDM.

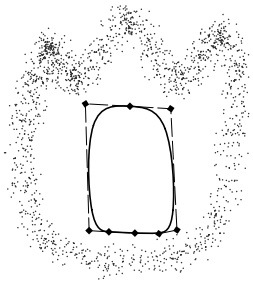
Figure 12: (Example 3) Comparison of the three methods for fitting a smooth target shape. Log scale (base 10) is used for the error axis. TDM becomes divergent after 40 iterations. (Number of data points: 1048)



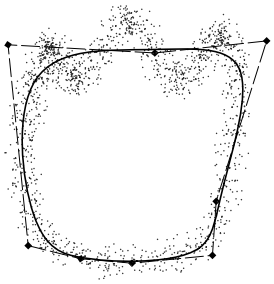
(a) The evolution surface of PDM.

(b) The evolution surface of SDM.

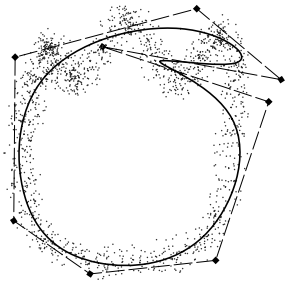
Figure 13: The evolution surfaces of PDM and SDM for the data in Example 3 (Figure 12). Only 104 of the original 1048 data points are shown for ease of visualization.



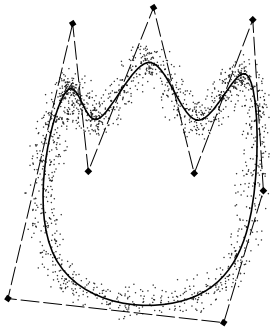
(a) A closed target shape and an initial B-spline curve.



(b) The fitting curve generated by PDM in 50 iterations.

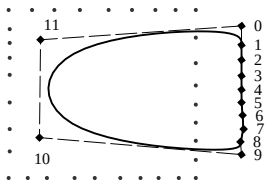


(c) The fitting curve generated by TDM in 50 iterations.

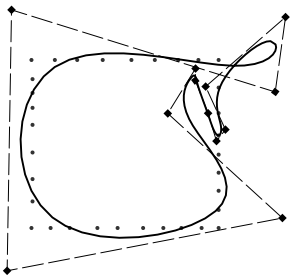


(d) The fitting curve generated by SDM in 50 iterations.

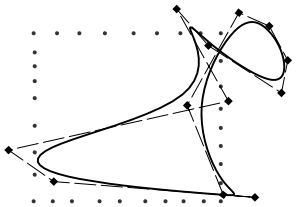
Figure 14: (Example 4) Comparison of the three methods for fitting an extremely noisy target shape. After 50 iterations, SDM generates a satisfactory fitting curve (d), but TDM becomes unstable (c), PDM is still improving at a slow rate; PDM needs about 400 iterations to generate a fitting curve similar to the one shown in (d). (Number of data points: 1630)



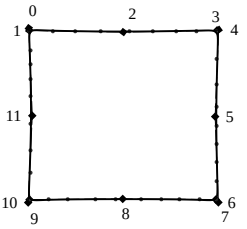
(a) A square-shaped target shape and an initial B-spline curve.



(b) The fitting curve generated by PDM in 20 iterations.

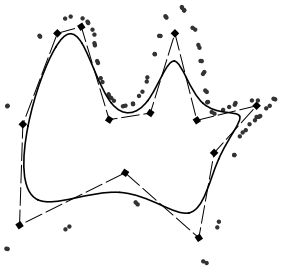


(c) The fitting curve generated by TDM in 20 iterations.

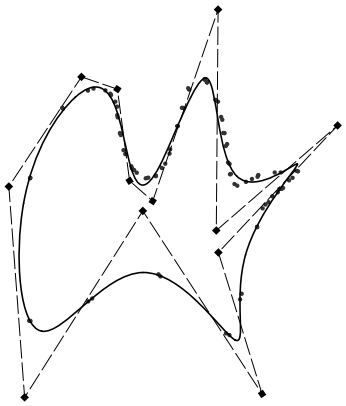


(d) The fitting curve generated by SDM in 20 iterations.

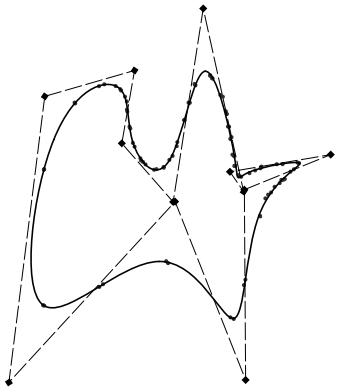
Figure 15: (Example 5) Comparison of the three methods for approximating a square-shaped target shape. Both PDM is trapped in a poor local minimum, and TDM eventually becomes divergent. (Number of data points: 33)



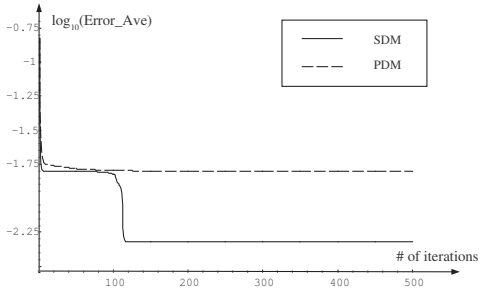
(a) A target shape with sparse data points and an initial B-spline curve.



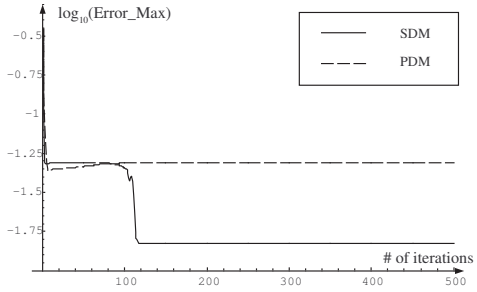
(b) The fitting curve generated by PDM in 35 iterations.



(c) The fitting curve generated by SDM in 35 iterations.

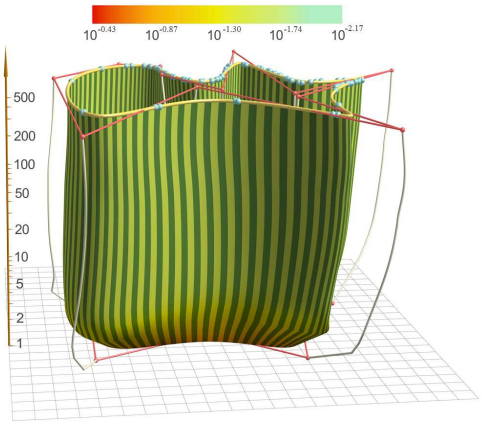


(d) The average error versus the number of iterations of PDM and SDM.

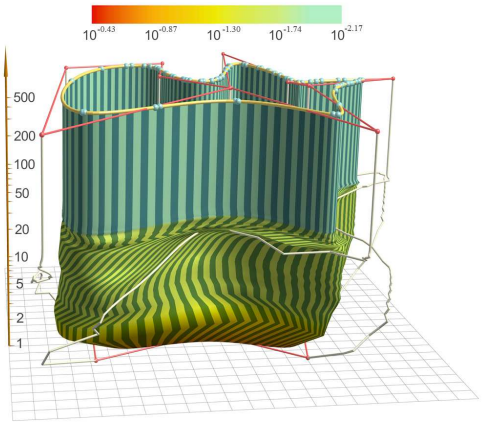


(e) The maximum error versus the number of iterations of PDM and SDM.

Figure 16: (Example 6) Comparison of the three methods for fitting sparse data points with sharp features. PDM is trapped in a local minimum that gives a larger error than the fitting curve generated in by SDM shown in (d). TDM does not converge for this example. (Number of data points: 83)



(a) The evolution surface of PDM.



(b) The evolution surface of SDM.

Figure 17: Evolution surfaces of PDM and SDM for the data in example 6 (Figure 16.

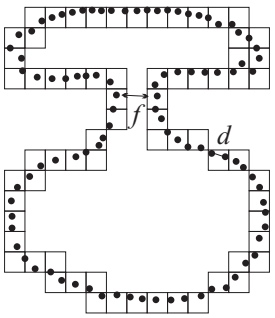


Figure 18: Quad-tree cell partition for specifying an initial curve.

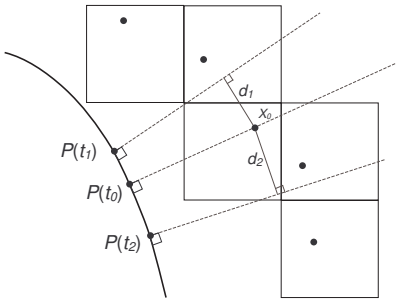
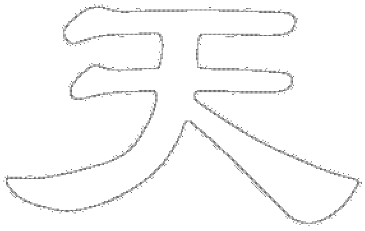
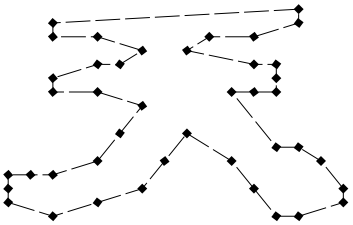


Figure 19: Computation of foot-points on a fitting curve.

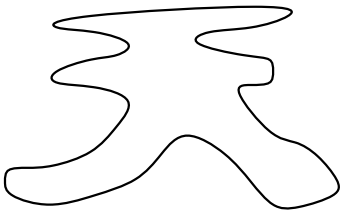
(a) A Chinese character, *tian*.



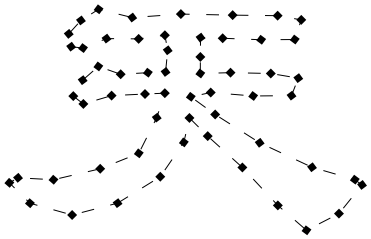
(b) The contour of the character (2656 points).



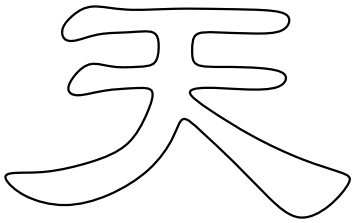
(c) 44 control points of an initial B-spline curve.



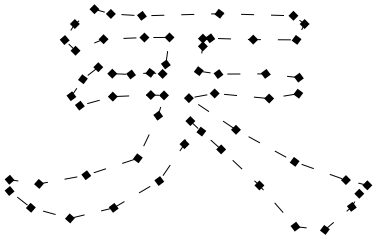
(d) The initial B-spline curve.



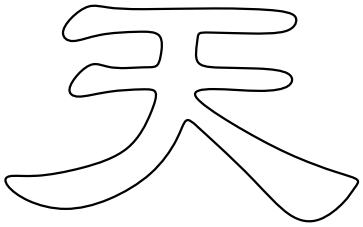
(e) Control points generated by SDM.



(f) The B-spline fitting curve from (e).



(g) Control points generated by PDM.



(h) The B-spline fitting curve from (g).

Figure 20: Control points insertion a deletion is applied together with SDM and PDM to fitting a B-spline curve to the contour of a character. The initial control points are obtained by quad-tree partition (c). The coefficient of smoothing term is $\lambda = 0.005$ and the threshold (summation of Euclidean distances) is 0.028. SDM leads to 59 control points after 54 iterations and PDM leads to 60 control points after 352 iterations.

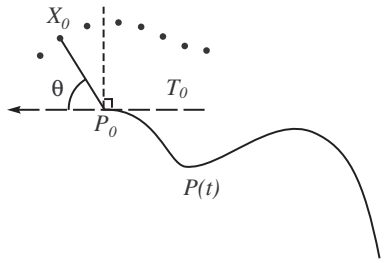
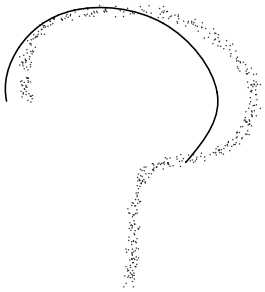
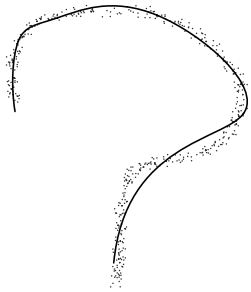


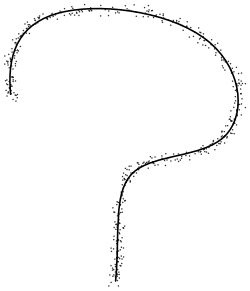
Figure 21: Deriving an error term for an outer data point X.



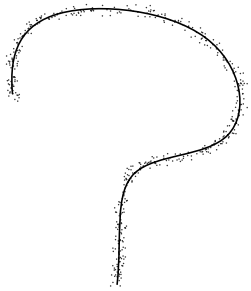
(a) An open target shape and an initial open B-spline curve with 8 control points.



(b) The fitting curve generated by PDM in 20 iterations.

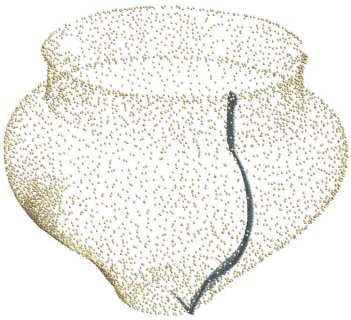


(c) The fitting curve generated by TDM in 20 iterations.

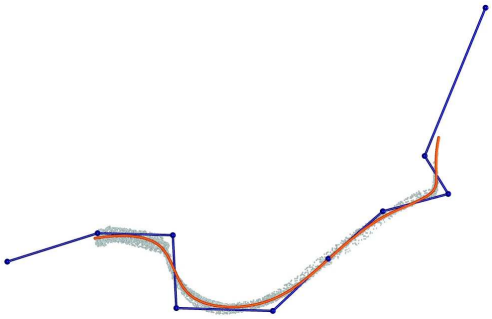


(d) The fitting curve generated by SDM in 20 iterations.

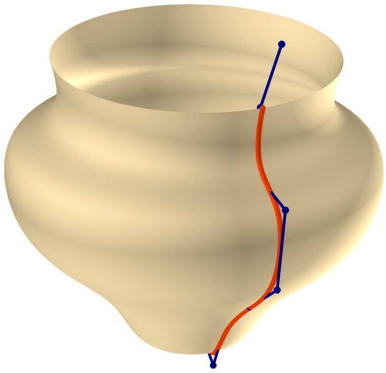
Figure 22: Comparison of the three methods for fitting an open curve to point cloud data. PDM needs about 600 iterations to reach the small approximation quality of SDM as shown in (c). TDM works well for this data set. (Number of data points: 472)



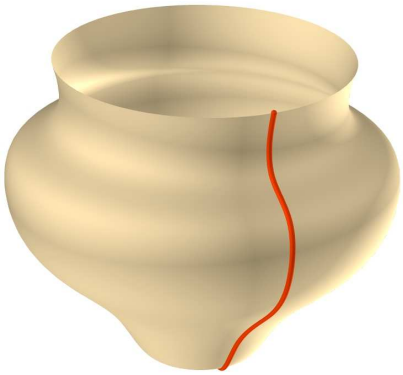
(a) A 3D point cloud representing an approximate revolution surface. The original scan data with 423697 points has been reduced to the shown 5077 points.



(b) The thick planar point cloud is fitted with a uniform cubic B-spline curve computed by SDM.



(c) The reconstructed revolution surface with the B-spline curve as a meridian.



(d) The reconstructed revolution surface.

Figure 23: Reconstruction of a revolution surface from point clouds using a B-spline fitting curve computed by SDM.

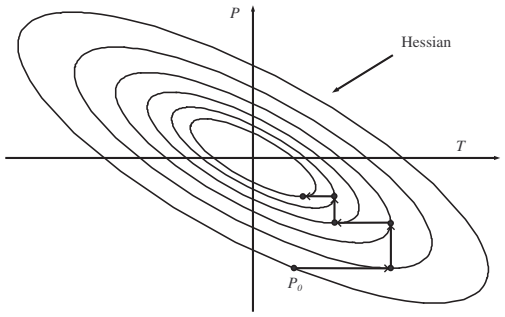
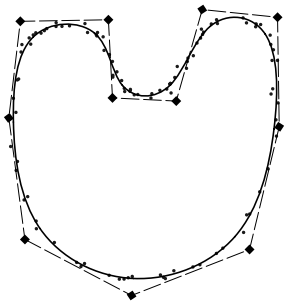
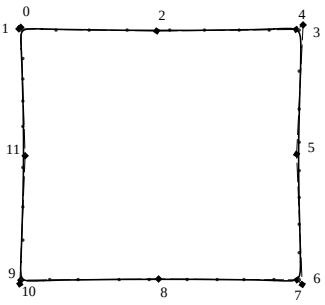


Figure 24: The alternating minimization steps of PDM. $\mathcal{P}$ and $\mathcal{T}$ stand for the linear subspaces spanned by the control points P_i and the data parameters t_k.



(a) The fitting curve generated by TDM in 20 iterations, using the Armijo rule, for the data set in Figure 10(a).



(b) The fitting curve generated by TDM in 20 iterations, using the Armijo rule, for the data set in Figure 15(a).

Figure 25: TDM with step size control.