Fitting B-spline Curves to Point Clouds by Curvature-Based Squared Distance Minimization

Wenping Wang
University of Hong Kong
Helmut Pottmann
Vienna University of Technology
and
Yang Liu
University of Hong Kong

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 curvature-based 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 often 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 curvature-based 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 high curvature variations, whereas optimization based on the PD error term is the alternating method that is known to have linear convergence.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Curve, surface, solid, and object representations*; J.6 [**Computer-Aided Engineering**]: Computer-aided design (CAD); G.1.6 [**Numerical Analysis**]: Optimization—*Least squares methods*

General Terms: Shape reconstruction, scatter data approximation

Additional Key Words and Phrases: B-spline curve, curve fitting, point cloud, least squares problem, optimization, squared distance, Gauss-Newton method, quasi-Newton method

Authors' address: Wenping Wang, Department of Computer Science, The University of Hong Kong, Pokfulam Road, Hong Kong, China; email wenping@cs.hku.hk; Helmut Pottmann, Institute of Discrete Mathematics and Geometry, Vienna University of Technology, Wien, Austria; email pottmann@geometrie.tuwien.ac.at; Yang Liu, Department of Computer Science, The University of Hong Kong, Pokfulam Road, Hong Kong, China; email: yliu@cs.hku.hk. The work of the first author was partially supported by a CRCG grant from The University of Hong Kong. The second author acknowledges support by the Austrian Science Fund under grant P16002-N05.

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 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 assume throughout that the order and the knots of the B-spline curve are fixed, so they are not subject to optimization. This simplifying assumption allows us to give a clear explanation of the general idea of our new optimization scheme. The very same idea of optimization presented here can also be extended to the more general setting of fitting a NURBS curve with free weights and knots to be optimized, by variable linearization and inequality constraints.

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 method 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. One of our contributions is the introduction of a novel error term defined by a curvature-based quadratic approximant of squared distances from data points to the fitting curve. For brevity, 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 will also discuss the B-spline curve fitting problem from the viewpoint of optimization to provide insights into the superior performance of SDM. We will 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 will also show that TDM is a variant of the Gauss-Newton method. Furthermore, PDM is, in fact, the alternating method for solving a separable problem and is known to have only linear convergence [Ruhe and Wedin 1980; Bjorck 1996; Speer et al. 1998]. This systematic study of the relationship between these geometrically motivated 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., [Pavlidis 1983; Plass and Stone 1983; Pratt 1985; Walton and Xu 1991; Goshtasby 2000]) and many other application areas. Instead of attempting a comprehensive review, we will only discuss some main results in the literature to provide a background for our work.

Let $X_k \in R^2$, $k = 1, 2, \dots, n$, 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. Suppose that we have a specific B-spline curve $P_c(t) = \sum_{i=1}^{m} B_i(t)P_{c,i}$ with control points $\mathcal{P}_c = (P_{c,1}, P_{c,2}, \dots, P_{c,m})$, which can be an initial fitting curve or the current fitting generated from the last iteration. Let $\mathcal{D} = (D_1, D_2, .., D_m)$ be the variable updates to $\mathcal{P}_c$ to give the new control points $\mathcal{P}_+ = \mathcal{P}_c + \mathcal{D}$. Let $P_+(t) = \sum_{i=1}^{m} B_i(t)(P_{c,i} + D_i)$ denote the B-spline curve with updated control points $\mathcal{P}_+$.

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 [Lee 1989; Hoschek and Lasser 1993; Farin 1997] are used for data parameterization. Then, with the fixed t_k, the function

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

which is a local quadratic model of f in (1), is minimized by solving a linear system of equations to yield updated control points $\mathcal{P}_+$, and hence the updated fitting curve $P_+(t)$.

A commonly used data parameterization method is to choose t_k such that $P_c(t_k)$ is the closest point from the current fitting curve to the data point X_k; ($P_c(t_k)$ is also called the *foot point* of X_k on the curve $P_c(t)$). Then an iterative method can be developed by interleaving this step of foot point computation with the minimization of the local quadratic model $\hat{f}$ in (2) to compute the updated control points $\mathcal{P}_+$. 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 to be determined; therefore, we will 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.) This minimization scheme will be called the *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.

Hoschek [Hoschek 1988] proposes an iterative scheme, called *intrinsic parameterization*, which also uses the PD error term but performs parameter correction using a formula that is a first-order approximation to exact foot point computation. Bercovier and Jacob [Bercovier and Jacob 1994] prove that the intrinsic parameterization method is equivalent to Uzawa's method for solving a constrained minimization problem, but they do not establish the convergence rate of the intrinsic parameterization method or that of PDM. Higher order approximation or accurate computation of foot points for data parameterization are discussed in [Hoschek and Lasser 1993; Saux and Daniel 2003; Hu and Wallner 2005].

PDM, or its simple variants, are the most commonly applied method for curve fitting in computer graphics and CAD [Plass and Stone 1983; Hoschek 1988; Goshtasby 2000; Saux and Daniel 2003]. The same idea of PDM is also widely used for surface fitting [Hoppe et al. 1994; Forsey and Bartels 1995; Hoppe 1996; Ma and Kruth 1995; Haber et al. 2001; Wang and Phillips 2002; Djebali et al. 2002; Greiner et al. 2002; Weiss et al.

2002; Maekawa and Ko 2002; Taubin 2002], with B-spline surfaces as well as other types of surfaces. The popularity of PDM might be explained by its simplicity — the error term $e_{PD,k}$ in (3) 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, $e_{PD,k}$ 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.

Another error term, often used in the computer vision community (e.g., [Blake and Isard 1998]), 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 current fitting curve $P_c(t)$ at the point $P_c(t_k)$. We will call $e_{TD,k}$ the *tangent distance error term* or the *TD error term*, since $e_{TD,k}$ gives the squared distance from X_k to the tangent of $P_c(t)$ at the foot point $P_c(t_k)$ when $P_+(t)$ is $P_c(t)$. The TD error term is illustrated in Figure 2.

The TD error term $e_{TD,k} = [(P_+(t_k) - X_k)^T N_k]^2$ can also be combined with data parametrization via foot point projection to yield a B-spline curve fitting method, as used in [Blake and Isard 1998] for boundary extraction in motion tracking. We will call this method *tangent distance minimization* or *TDM*. TDM minimizes in each iteration the function

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

Treating the control points $\mathcal{P}_+$ 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 $e_{TD,k} = [(P_+(t_k) - X_k) \cdot N_k]^2$ 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 still be poorly approximated by the curve $P_+(t)$ even if X_k is nearly on L_k, i.e., when the TD error $[(P_+(t_k) - X_k) \cdot N_k]^2$ is nearly zero (see Figure 3).

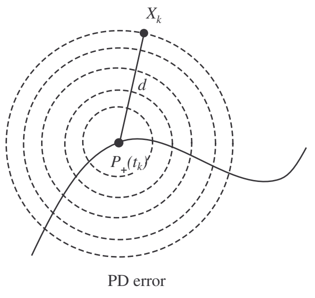


Fig. 1. Iso-values curves of the point distance (PD) error term.

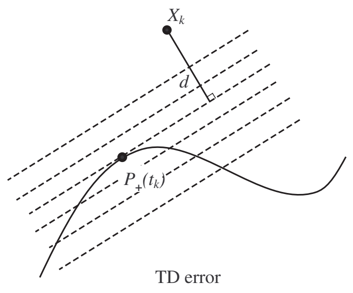


Fig. 2. Iso-values curves of the tangent distance (TD) error term.

This disparity between approximation quality and error measurement is the cause of the instability of TDM near a high-curvature part of the target shape, as will be illustrated later with 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 will show that TDM uses Gauss-Newton iteration 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. Since $P_c(t_k)$ is fixed foot point on the current fitting curve $P_c(t)$ of X_k, if $P_+(t)$ is the same as $P_c(t)$ then both $e_{PD,k}$ and $e_{TD,k}$ give the same value of $d^2(P_c(t_k), X_k)$. However, for optimization purpose, we need to regard $d^2(P_+(t_k), X_k)$ as a function of variable control points $\mathcal{P}_+$, 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 $e_{PD,k} \equiv ||P_+(t_k) - X_k||^2 = c$ of the PD error term is a circle (see Figure 1), and the iso-value curve $e_{TD,k} \equiv [(P_+(t_k) - X_k) \cdot N_k]^2 = c$ 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 ellipse-shaped iso-value curves can be devised to yield a more balanced performance between efficiency and stability. We will see that such an error term is naturally provided by a curvature-based quadratic approximation to the squared distance function.

2.2 Second order approximation to squared distance function

Given a curve $\mathcal{C}$ in E^2, one may define the squared distance function that assigns to each point X in E^2 the squared distance from X to $\mathcal{C}$. The second order approximation to this distance function is given in [Ambrosio and Montegazza 1998]. This approximation is then studied in detail and applied to solving a number of shape fitting problems in [Pottmann et al. 2002; Pottmann and Hofer 2003]. Below we review this work briefly.

Let O be the closest point on a twice differentiable curve $\mathcal{C}$ to a fixed point X_0 (see Figure 4). Consider the local Frenet frame of $\mathcal{C}$ with its origin at O and its two coordinate axes being the tangent vector and the normal vector of the curve $\mathcal{C}$ at O. Let $\rho > 0$ be the curvature radius of $\mathcal{C}$ at O. We use an orientation of the curve normal such that $K = (0, \rho)^T$ is the curvature center of $\mathcal{C}$ 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 $\mathcal{C}$, and $d > 0$ if X_0 and K are on the same side of $\mathcal{C}$. We note that there is always $d < \rho$ when $d > 0$, for otherwise O cannot be the closest point on the curve $\mathcal{C}$ to X_0.

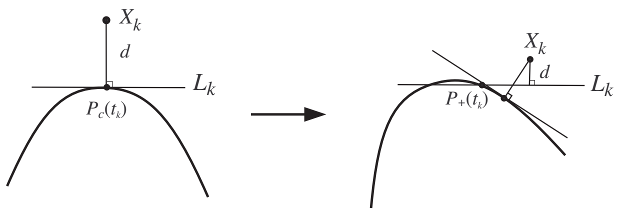


Fig. 3. In a neighborhood of a high curvature point, the true approximation error can be rather big even when the TD error d is small.

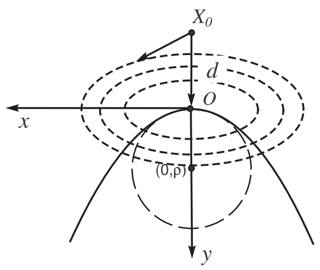


Fig. 4. A second order approximant of the squared distance function to the curve C at X_0.

Consider a point $X = (x, y)^T$ in a neighborhood of X_0. The second order approximant of the squared distance from X to the curve $\mathcal{C}$ is [Ambrosio and Montegazza 1998; Pottmann and Hofer 2003]

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

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

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

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

is used in [Pottmann et al. 2002] as a positive semi-definite quadratic error term for solving geometric optimization problems. Note that, with this modification to (6) in the case of $0 \le d < \rho$, $\hat{g}(x,y)$ becomes a first order approximation to the squared distance function to $\mathcal{C}$ in a neighborhood of X_0. An alternative to this modification is to replace the first coefficient $d/(d-\rho)$ by $\max\{0, d/(d-\rho)\}$, thus also making the distance measurement positive semi-definite in all cases.

The above approximation to the squared distance of a smooth target curve is used in [Pottmann et al. 2002] for fitting a B-spline curve to the target curve as follows. Given a target curve to be approximated and the current B-spline fitting curve $P_c(t)$ with control points $\mathcal{P}_c = \{P_{c,i}\}$, a set of densely distributed points S_k, called *sensor points*, are first sampled on $P_c(t)$. 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. Let $\mathcal{P}_+ = \mathcal{P}_c + \mathcal{D}$ denote the updated control points of the B-spline fitting curve, where $\mathcal{D}$ are the incremental updates to the current control points $\mathcal{P}_c$. The error term associated with each sensor point is defined as

$$\hat{e}_k = \hat{g}(S_k(\mathcal{P}_+)), \tag{8}$$

which is quadratic in the control points $\mathcal{P}_+$, since each S_k is a linear combination of the control points. Then the local error function $f = \frac{1}{2}\sum_k \hat{e}_k + \lambda f_s$, where f_s is a regularization term that is quadratic in $\mathcal{P}_+$, is minimized to find the updated control points $\mathcal{P}_+$ by solving a linear system of equations. This minimization step is iterated to make the fitting curve $P(t)$ move towards the target curve.

The superior efficiency of the above curve fitting scheme comes from the fact that the error function $\hat{e}_k$ in (8) is an accurate approximation to the true squared distance function from S_k to the target curve $\mathcal{C}$, in terms of the incremental updates $\mathcal{D}$. However, this method assumes that the target shape is a smooth curve whose tangent and curvature information can easily be evaluated or estimated accurately, thus it is not applicable to a target point defined by a point cloud because of the difficulty in computing accurate tangent and curvature from noisy or sparsely distributed data points.

3. FITTING A B-SPLINE CURVE TO A POINT CLOUD USING SDM

In this section we introduce a new *SD error term* for fitting a B-spline curve to a point cloud. We emphasize that *this new SD error term is defined by a quadratic approximation to 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, in contrast to the method in [Pottmann et al. 2002] (or see Section 2.2) in which errors are measured orthogonal to the fixed target curve and therefore also a different objective function is minimized.

3.1 A new quadratic approximation to the squared distance

Given the current B-spline fitting curve $P_c(t) = \sum_{i=1}^{m} B_i(t)P_{c,i}$, let $P_+(t)$ denote the fitting curve with updated control points $\mathcal{P}_+ = \mathcal{P}_c + \mathcal{D}$, where $\mathcal{P}_c = \{P_{c,i}\}$ and $\mathcal{D}$ are incremental updates to $\mathcal{P}_c$. Suppose that $P_c(t_k)$ is the foot point of the data point X_k on $P_c(t)$. Let T_k and N_k be the unit tangent vector and the unit normal vector of the current fitting curve $P_c(t)$ at the foot point $P_c(t_k)$, $\rho > 0$ is the curvature radius of $P_c(t)$ at $P_c(t_k)$ and $|d| = \|P_c(t_k) - X_k\|$, with the same convention on the sign of d as made in Section 2.2. When the control points $\mathcal{P}_+$ change, with the same parameter t_k, the foot point $P_c(t_k)$ becomes a variable point $P_+(t_k)$, and the unit tangent vector $\tilde{T}_k$, the unit normal vector $\tilde{N}_k$ and curvature radius $\tilde{\rho}$ of the curve $P_+(t)$ at the point $P_+(t_k)$ all vary with $\mathcal{P}_+$.

To obtain a quadratic approximation to the squared distance from X_k to the curve $P_+(t)$, we assume that $\tilde{T}_k$, $\tilde{N}_k$ and $\tilde{\rho}$ are fixed to be T_k, N_k and ρ, i.e., they do not vary with $\mathcal{P}_+$. This assumption implies that, locally at the point $P_c(t_k)$, we will only consider a differential translation of the curve $P_c(t)$ into the curve $P_+(t)$. Since this translation is relative to the data point X_k, we may view, in a relative sense, that $P_+(t)$ is a fixed curve and X_k undergoes a translation. Therefore, we may use the formula (6) to approximate the squared distance from X_k to the curve $P_+(t)$, expressed in the global coordinate system, as

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

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

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

Clearly, $e_{SD,k}(\mathcal{D})$ is a positive semi-definite quadratic function of $\mathcal{D}$ in all cases. Since $e_{SD,k}(\mathcal{D})$ is derived from a direct attempt to accurately approximate the squared distance function, we will call $e_{SD,k}(\mathcal{D})$ the *squared distance error term* or *SD error term* for short. We stress that, due to the simplifications we have made, $e_{SD,k}(\mathcal{D})$ is, in general, no longer a second order approximation to the squared distance function, but rather a first order approximation that is more accurate than the PD error term or the TD error term.

When $d < 0$, the level-set curve of $e_{SD,k}(\mathcal{D}) = c$ is an ellipse centered at the point $P_+(t_k)$, if the X_k is treated as a variable point. When the control points $\mathcal{P}_+$ change, the ellipse is translated by keeping its center at $P_+(t_k)$ but with its shape, size and orientation remaining unchanged (see Figure 5). The SD error term $e_{SD,k}$ becomes the TD error term $e_{TD,k}$ when $0 \le d < \rho$, i.e., when (i) the data point X_k is sufficiently near $P_c(t_k)$ relative to the magnitude of ρ; *and* (ii) X_k is on the convex side of the curve $P_c(t)$, i.e., X_k and the curvature center K are on the same side of the curve $P_c(t)$. 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 $P_c(t)$ in a neighborhood of $P_c(t_k)$.

Iterative minimization of the squared distance using the SD error term (10), interleaved with foot point computation for data parameterization, will be called *squared distance minimization* or *SDM*. Although the tangent and normal vectors T_k and N_k, as well as ρ, are kept fixed during an iteration, they are updated at the beginning of each iteration to reflect the continual change of the shape of the fitting curve.

The above derivation of the SD error term is based on a geometric argument. Using a second order Taylor expansion, we will reveal the relationship between this SD error term and the Newton iteration later in Section 6.3, thus providing another derivation of the SD error term.

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 quadratic approximation f_{SD} of the objective

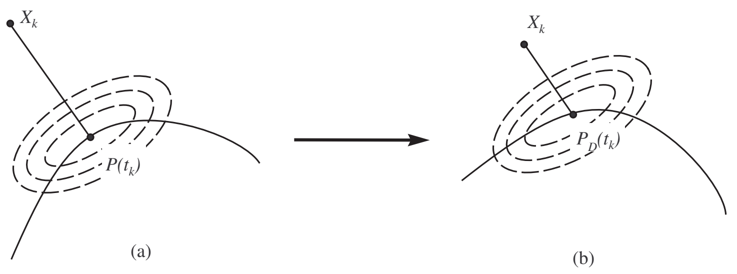


Fig. 5. The SD error term $e_{SD,k}(\mathcal{D})$ defined by (10) is shown via its iso-value curves in the case of $d < 0$. (a) Before updating the control points $\mathcal{P}$. (b) The translation of $e_{SD,k}(\mathcal{D})$ after updating $\mathcal{P}$.

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 fitting curve.

(4) Repeat steps 2 and 3 until convergence, i.e., 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 quadratic 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 a particular error term (PD, TD or SD) for the data point X_k, F_1 and F_2 are energy terms defined as

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

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

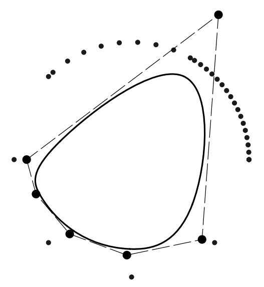
For a fixed B-spline fitting curve $P(t)$, the Euclidean distance from data point X_k to $P(t)$ is denoted by $d_k = \|P(t_k) - X_k\|$, where $P(t_k)$ is the foot point of X_k on $P(t)$. Then, for evaluating the approximation error, we define the average error, which is the *root mean square error*, as

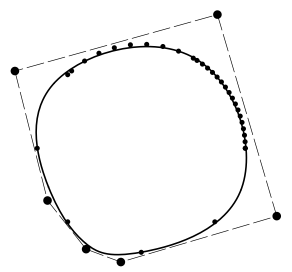
$$Error\_Ave = \left[\frac{1}{n}\sum_{k=1}^{n} d_k^2\right]^{1/2},$$

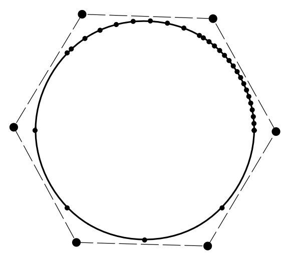
and the maximum error as

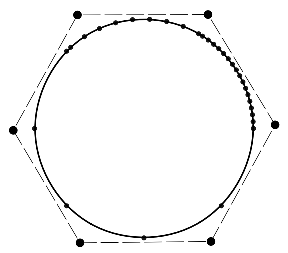
$$Error\_Max = \max_{k=1}^{n}\{d_k\}.$$

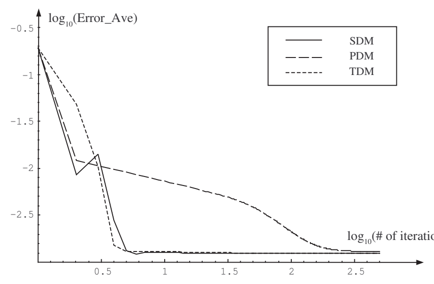
We present below the results of applying the three methods — PDM, TDM and SDM — to fitting a cubic B-spline curve with uniform knots to several sets of unorganized data points. The same values of energy coefficients $\alpha = 0$ and $\beta = 0.001$ are used for all the examples in this section, unless specified otherwise.


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

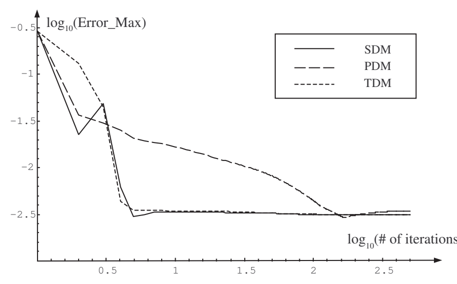

(b) The fitting curve generated by PDM in 10 iterations.


(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.

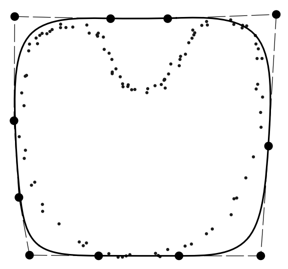
Fig. 6. (Example 1) Comparison of the three methods on a set of 32 sparse data points on a circle. In this figure log scale (base 10) is used for the iteration axis in Figures (e) and (f) in order to distinguish the error curves of TDM and SDM.

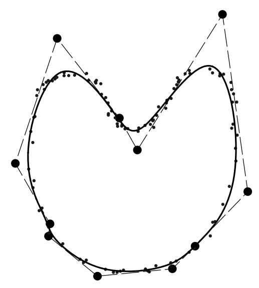
Example 1. *Non-uniform data points on a circle.* (See Figure 6.) TDM and SDM converge with roughly the same speed, and both converge much faster than PDM does, as shown in Figures 6(e) and (f). PDM takes about 100 iterations to reach the same small error values produced by SDM in fewer than 10 iterations.

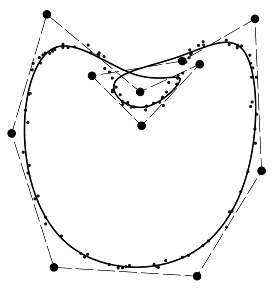
Example 2. (See Figure 7.) For this set of data points, SDM again converges much faster than PDM does, while TDM is trapped in a local minimum, producing a curve with self-intersection. To visualize the evolution of an iterative optimization process (PDM or SDM), we place the fitting curves generated in successive iterations at successive heights to form an *evolution surface* (see Figure 8). 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 evolving control points are shown by white curves in space. 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.

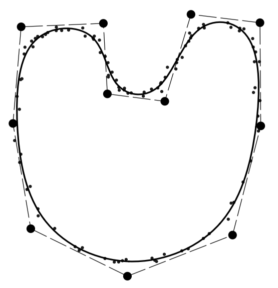
The evolution surfaces generated by PDM and SDM (Figure 8), viewed from two different directions, show that PDM experiences a slow convergence process, while SDM converges quickly with conspicuous tangent flow of the control points in the first 10 iterations.

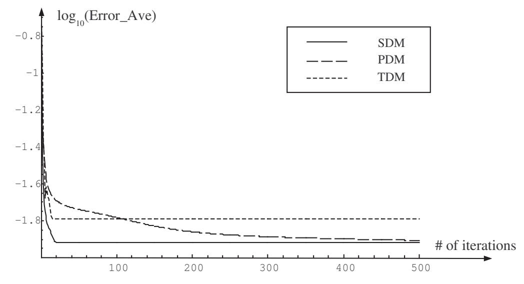
Example 3. (See Figure 9.) This set of data points is extremely noisy. After 50 iterations, SDM has already produced an acceptable result, while PDM converges slowly and TDM becomes unstable. Here strong tangential flow of control points is observed in SDM to move some control points at the bottom of the initial curve to the top in (d).


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

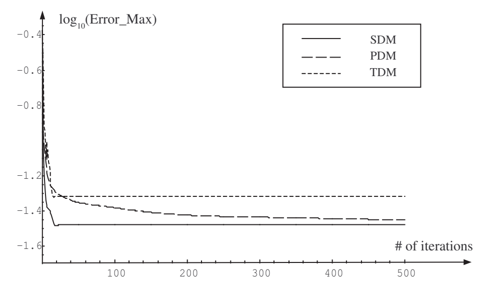

(b) The fitting curve generated by PDM in 20 iterations.


(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.



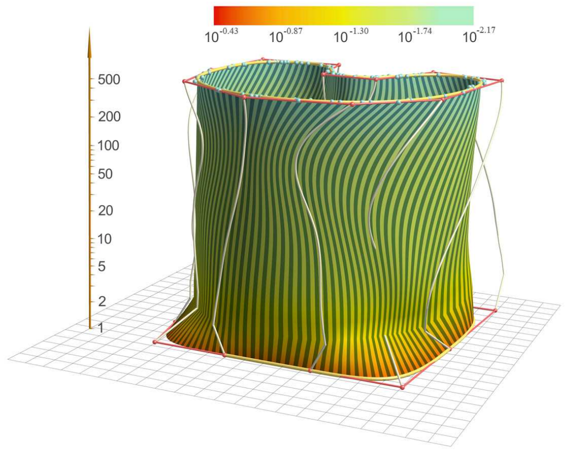
(f) The maximum error versus the number of iterations of the three methods.

Fig. 7. (Example 2) Comparison of the three methods on a set of 102 data points. SDM converges faster than PDM does. TDM is trapped in a local minimum with self-intersection in the fitting curve.

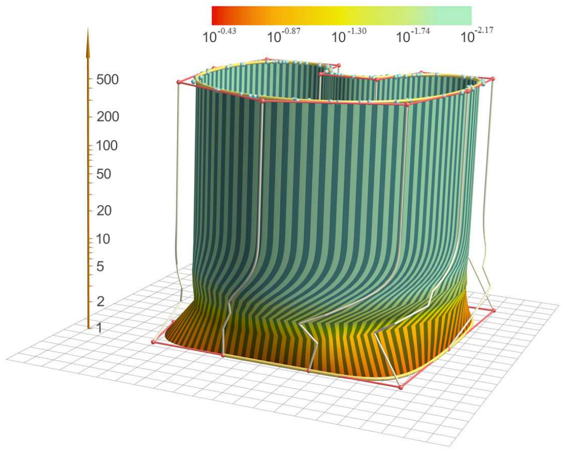
Example 4. (See Figure 10.) The difficulty with this test lies in the corner points of the target shape and the highly non-uniform distribution of the control points of the initial B-spline curve. After 20 iterations, PDM gets trapped in an unacceptable local minimum and TDM becomes divergent, while SDM converges successfully. Again SDM exhibits strong tangential flow responsible for re-distributing control points initially clustered at one side on the initial curve over the target shape to well approximate the four corner points.

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

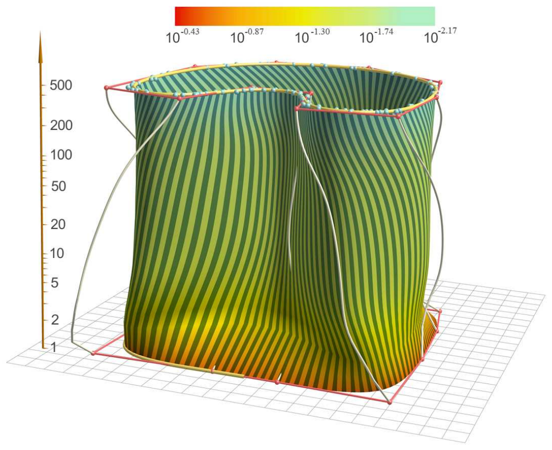
1) PDM exhibits the slowest convergence among the three methods, and is often trapped at a poor local minimum. Our experiments confirm the theoretical conclusion that PDM has, in general, only linear convergence. This is further explained in Section 6.
2) TDM demonstrates fast convergence when the target shape is not so noisy (i.e., representing a small-residue problem) and the initial 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 initial fitting curve is relatively far from the target shape. Increasing the value of the energy coefficient β in (11) usually improve the stability of TDM, as well as the fairness of the fitting curve, but often at the expense of a larger fitting error.
3) SDM exhibits much faster convergence than PDM does. The convergence of SDM is about as fast as



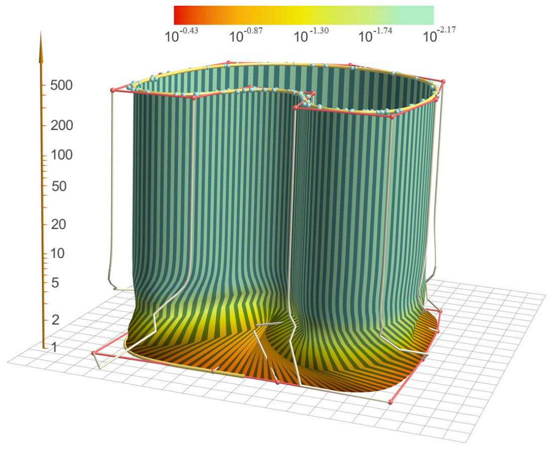
(a) PDM



(b) SDM



(c) PDM

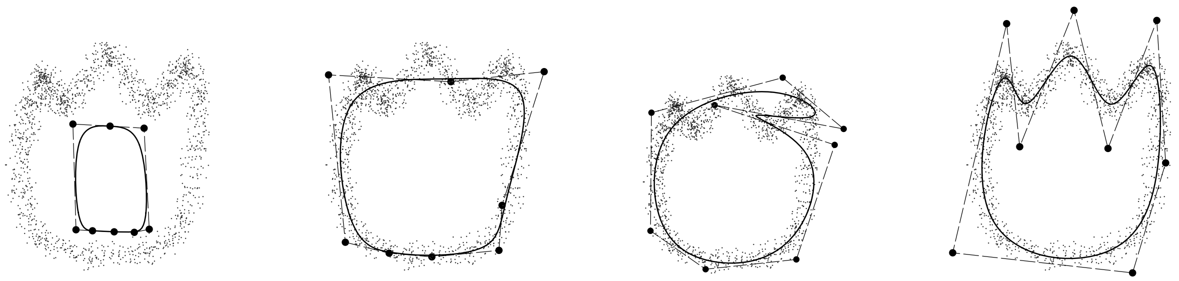


(d) SDM

Fig. 8. The evolution surfaces of PDM and SDM for the data in Example 2 (Figure 7) — viewed from two different angles. (a) The evolution surface of PDM; (b) The evolution surface of SDM with the same view angle as in (a); (c) The evolution surface of PDM from another view angle; (d) The evolution surface of SDM with the same view angle as in (c).

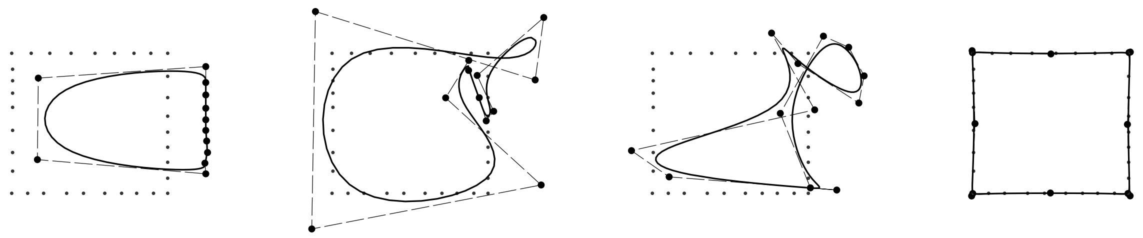
that of TDM; moreover, SDM is more stable than TDM, since TDM often does not converge for target shapes with shape features. This is mainly due to the fact that TDM is a Gauss-Newton method without step size control. We will discuss this in more detail in Section 6.

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



(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.

Fig. 9. (Example 3) Comparison of the three methods for fitting an extremely noisy data set of $1,630$ points. After 50 iterations, SDM generates a satisfactory fitting curve (d), but TDM becomes unstable (c), and PDM is still improving at a slow rate (b); PDM needs about 400 iterations to produce a fitting curve similar to the one by SDM shown in (d).



(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.

Fig. 10. (Example 4) Comparison of the three methods for approximating a square-shaped target shape consisting of 33 data points. PDM is trapped in a poor local minimum (b), TDM eventually becomes divergent (c), and SDM converges successfully in 20 iterations.

is dampened at a high-curvature region due to the role played by the curvature radius ρ and distance d in the SD error term. In contrast, tangential flow of control points is inhibited by the PD error term, causing stagnant improvement. Meanwhile, 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 use 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 the foot points of all data points in each iteration. For example, for the set of 1,630 data points used in Example 3, 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) initialization and adjustment of control points; 2) fast setup of error terms; and 3) adapting SDM to fit an open B-spline curve to data points.

5.1 Initialization and adjustment of control points

All the three methods we have discussed so far, PDM, TDM and SDM, are local minimization schemes; that is to say, their convergence depends on the initial value, i.e., the initial fitting curve. We would like to point out several possibilities of specification of the initial fitting curve, though this is not a focus of the present paper. The first obvious option is to let the user specify an initial B-spline fitting curve that is sufficiently close to the target shape and has an appropriate number of control points.

For a target shape defined by a set of dense points, an alternative is to compute a quadtree partition of the data points and then extract a sequence of points approximating the target shape from non-empty cells, i.e., those cells containing at least one data point. These extracted points can then be used as the control points of an initial B-spline fitting curve. Our experience shows that this method tends to produce too many control points at the beginning, so control point deletion is normally required during the fitting process in order to obtain a fitting curve with a minimal number of control points while still meeting a prescribed error threshold.

Another approach under our current investigation is based the active contour model. In this method, a simple initial fitting curve is specified either automatically or by the user. Then some external energy/force is used in combination with SDM to drive the fitting curve to converge to a complex target shape in a manner that ensures global convergence. The key issues here are (*i*) proper design of the external force so that features and concavities of the target shape can be captured; and (*ii*) progressive insertion of control points at appropriate locations and stages so as to provide increasing shape flexibility to cope with the complexity of the target shape. The details of this procedure are beyond the scope of the present paper and we wish to report on them in a separate paper. We refer to [Yang et al. 2004] for discussions on control point insertion and deletion in the context of a local B-spline curve fitting procedure.

5.2 Fast setup of error terms

Efficient computation of foot points on the fitting curve of data points is important, especially when they are a large number of data points, since an error term needs to be computed for each data point in every iteration. We use the following speedup method consisting of two phases: *preprocessing* and *query*. In preprocessing we first compute a uniform spatial partition of the data points with a proper cell size and record those nonempty cells. Next we sample a sufficient number of points on the fitting curve and compute the normal lines of the fitting curve at these sample points. Then we record the intersections between these normal lines and all the non-empty cells.

In the query phase, for each data point X_k, we find its containing cell and the two normal lines such that X_k is between the two lines and they are closest to X_k (see Figure 11). Let the two normal lines be associated with parameter values $\bar{t}_1$ and $\bar{t}_2$ of the fitting curve. Let d_1 and d_2 denote the distances from X_k to the two lines. Then, supposing that the current fitting curve is sufficiently close to the target shape, a good estimate $P(\tilde{t}_k)$ of the foot-point of X_k is given by linear interpolation $\tilde{t}_k = (d_2\bar{t}_1 + d_1\bar{t}_2)/(d_1 + d_2)$. The point $P(\tilde{t}_k)$ is then used as an initial point in a Newton-like iterative procedure to find the foot point $P(t_k)$ of X_k.

Figure 13 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 above are used for initial fitting curve specification and control point insertion. Again we see that, to achieve the same level of fitting quality, PDM needs about the same number of control points but much more iterations than SDM does. TDM without step size control fails to converge for this example, because the font outline has a number of high curvature feature points.

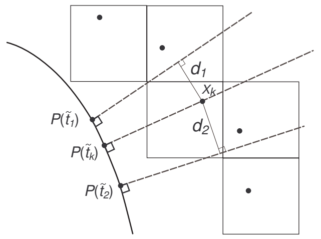


Fig. 11. Foot-point computation on a fitting curve.

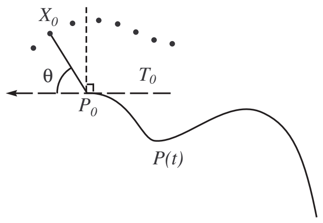


Fig. 12. Deriving an error term for an outer data point X.

5.3 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; 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 blending the SD error term and the PD error term. Specifically, referring to Figure 12, 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 $|\theta| < \pi/2$ (since X_0 is an outer point). Then the error term $e_{outer,0}$ to be used for X_0 is given by the following interpolation of the PD error term and SD error term,

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

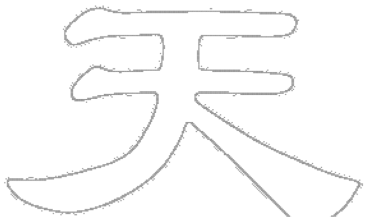
Here P_0 is regarded as a function of the control points.

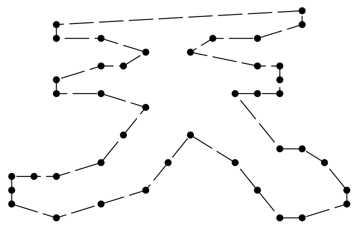
The rationale behind the interpolation in (13) is to use the PD error term partially for outer data points so that, through iterative optimization, the endpoint P_0 of the fitting curve is pulled towards the target endpoints; of course, the SD error terms are still used for all other non-outer points. 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.

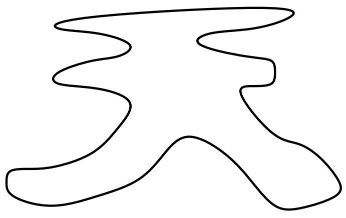
In the second case where none of the data points is an outer point, we just use the standard SDM method — that is, use the SDM error term for each data point, to make the fitting curve to contract to fit the target shape. A non-zero but small value of α for the energy term F_1 in (12) may be used to speed up the speed of contraction.

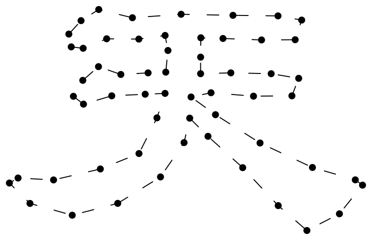
We are going to present two examples of fitting open B-spline curves to data points, using the technique described above, in combination with SDM and PDM. The first example is shown in Figure 14. We note that, in this example, PDM takes about 600 iterations to reach the same approximation error that 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 effectively for this example as well.

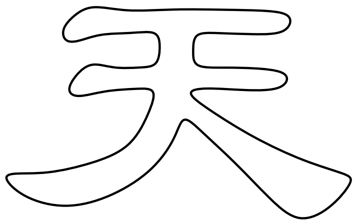
(a) A Chinese character, *tian*.

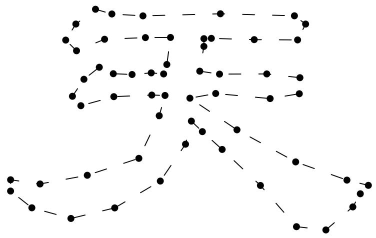

(b) The contour of the character (2,656 points).


(c) 42 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.

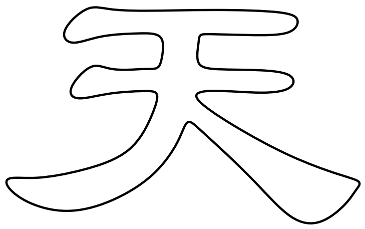
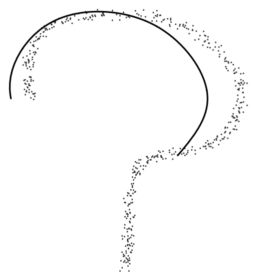
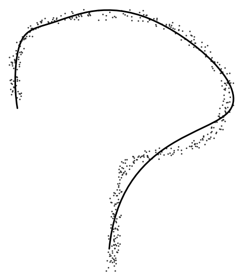
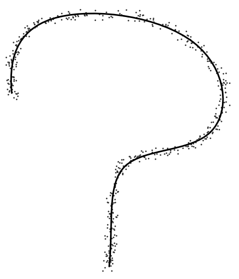

(h) The B-spline fitting curve from (g).

Fig. 13. SDM and PDM are applied to fitting a B-spline curve to the contour of a Chinese character in (a). The initial fitting curve in (d) has the control points in (c) that are extracted from a quad-tree partition of the contour data points in (b). The coefficient of the smoothing term is $\lambda = 0.005$. SDM produces the fitting curve in (f) with the 59 control points shown in (e) after 54 iterations and PDM produces the fitting curve in (h) with the 60 control points in (g) after 352 iterations.


(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.

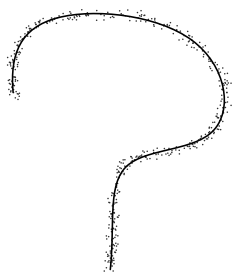
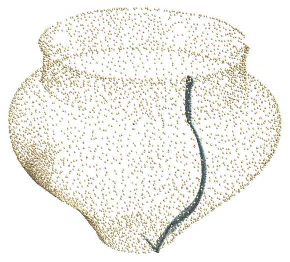
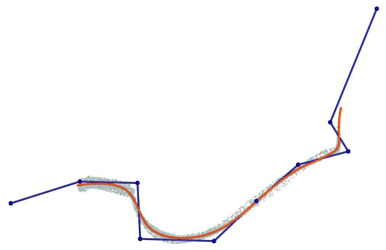

(d) The fitting curve generated by SDM in 20 iterations.

Fig. 14. Comparison of the three methods for fitting an open curve to a set of 472 data points. PDM needs about 600 iterations to reach the small approximation error of SDM as shown in (c).


(a)


(b)

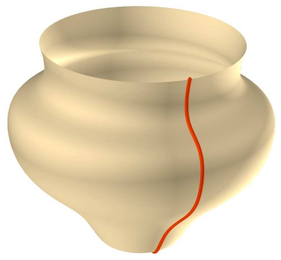

(c)

Fig. 15. Reconstruction of a revolution surface from a point cloud using a B-spline profile curve computed by SDM. (a) A revolution surface represented by a set of 5,077 points sampled from an original scan data of 423,697 points; (b) The thick planar point cloud is fitted with a uniform cubic B-spline curve computed by SDM; (c) The reconstructed revolution surface.

The second example, shown in Figure 15, is an application to reconstructing a revolution surface from a point cloud scanned in by a laser range scanner, following a method proposed in [Pottmann and Wallner 2001]. The basic idea is as follows. First, the rotation axis of the revolution surface is estimated from the data points. Then this axis is used to rotate the input data points in 3D (Figure 15(a)) into data points lying on a 2D plane (Figure 15(b)), from which the B-spline profile curve is reconstructed using SDM. Then this profile curve is used to generate a revolution surface (Figure 15(c)) approximating the input 3D data points.

6. DISCUSSION FROM 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 curve 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.,

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

We note that, treating $\mathcal{P} = \{P_i\}$ and $\mathcal{T} = \{t_k\}$ as two separable groups of variables and Eqn. (15) as constraints, this curve fitting problem is an instance of a *separable and constrained nonlinear least squares problem* for which the variable projection method using Gauss-Newton iteration often provides an efficient algorithm [Bjorck 1996]. This viewpoint will be helpful below 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 quadratic in the P_i, assuming that λ is a fixed constant throughout all iterations.

In the rest of this section we will first explain why PDM has linear convergence, by viewing it as an alternating method. We will then investigate the standard algorithms for nonlinear least squares problems, namely Gauss-Newton iteration and the Levenberg-Marquart method [Kelley 1999], in connection with TDM. We will show that the Gauss-Newton method based on variable projection [Bjorck 1996] is exactly the same as TDM, which is used in [Blake and Isard 1998]. From this we conclude on scenarios where TDM works well: a good initial position of the fitting curve and a small residual problem (i.e., data points are close to the final fitting curve); for a zero residual problem, optimization theory tells us that this method exhibits even quadratic convergence. The Levenberg-Marquart method 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 curve fitting as a nonlinear least squares problem. Although SDM is not a full Newton method because we do not use the complete Hessian, it comes close to it; in SDM we approximate the complete Hessian by a quadratic approximation to the squared distance to a fitting curve in a manner that makes SDM adaptable to local curvature variation.

In fact, all the methods above can be seen as gradient descent schemes in some appropriate 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 mention step size control [Kelley 1999] as a means of global convergence improvement of the three methods.

6.1 PDM as an alternating method

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 16): (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 $f(\mathcal{P}, \mathcal{T}_0)$; (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 $\mathcal{P}_1$. Thus, PDM minimizes the objective function in two subspaces parallel to $\mathcal{P}$ and $\mathcal{T}$, respectively, leading to a zigzag path near a local minimum as shown in Figure 16, which is reminiscent of the crawling behavior of the gradient descent method.

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 [Bjorck 1996].

6.2 TDM – Gauss-Newton iteration and its variants

First we will derive an expression of the gradient of the objective function in (14), which 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 through 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 partial derivative of $\mathcal{F}$ with respect to $\mathcal{P}$, not taking into account the dependency of t_k on $\mathcal{P}$. 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 a 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.,

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

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 those 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,

$$P(t) = B(t)\mathcal{P}, \tag{19}$$

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

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

Now let us consider the Gauss-Newton method. 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

$$\nabla^2 f = \sum_{k=1}^n \nabla f_k \cdot (\nabla f_k)^T + \sum_{k=1}^n f_k \nabla^2 f_k. \tag{21}$$

Since the computation of $\nabla^2 f_k$ is usually too costly, the Gauss-Newton method uses only the first part in (21) to approximate the Hessian $\nabla^2 f$. This amounts 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 B-spline curve fitting problem, the update 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) and Eqn. (19), since $\mathcal{F} = P(t_k) - X_k$, we have

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

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

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

which is interleaved with 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

$$\begin{aligned} f_{GN} &= \frac{1}{2}\sum_k [(X_k - P(t_k))^T N_k - \sum_i B_i(t_k) D_i^T N_k]^2 = \frac{1}{2}\sum_k [(X_k - \sum_i B_i(t_k)(P_i + D_i))^T N_k]^2 \\ &= \frac{1}{2}\sum_k [(X_k - P_+(t_k))^T N_k]^2 = \frac{1}{2}\sum_k e_{TD,k}, \end{aligned} \tag{22}$$

where $e_{TD,k}$ is defined in Eqn. (4). Hence, the minimization of the TD error function in (5) in TDM is equivalent to Gauss-Newton iteration.

The TD error term $e_{TD,k}$, unlike the SDM error term proposed in this paper, counts for neither the distance from the data point 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 (21).

Strictly speaking, TDM is not the standard Gauss-Newton method, since there is a step of foot point computation interleaved with Gauss-Newton iteration. In TDM the Gauss-Newton step is applied in the tangent plane to the constraint surface defined by the constraints (16). Such an algorithm is called a *variable projection method using a Gauss-Newton method* for solving a separable and constrained nonlinear least squares problem and can be shown [Ruhe and Wedin 1980; Bjorck 1996] to have the same asymptotic convergence behavior as the full Gauss-Newton method applied to all variables (i.e., $\mathcal{P}$ and $\mathcal{T}$ in our case).

TDM also shares the same framework of the so called *generalized reduced gradient (GRG) method* [Luenberger 1984] for solving a nonlinear constraint problem with two separable groups of variables; the steepest gradient direction is used in the tangent plane of the constraint surface in the GRG method, while a Gauss-Newton step in used in TDM.

It is well known [Kelley 1999, pp. 24] that, in a Gauss-Newton method, 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

$$\|e_+\| \le K(\|e_c\|^2 + \|R(x^*)\| \, \|e_c\|), \tag{23}$$

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 (23) 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* [Kelley 1999].

Another way to modify Gauss-Newton is a regularization with the *Levenberg-Marquart method* [Kelley 1999], in which a scalar multiple of the identity matrix is added to the approximate Hessian. In our setting, this method requires the minimization of

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

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 in the setting of curve fitting 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 [Sarkar and Menq 1991] 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 [Sarkar and Menq 1991] that this variant of the L-M method is about 10 times slower than PDM per iteration.

A global Gauss-Newton method is implemented in [Speer et al. 1998] to update the control points P_i and the parameter values t_k 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 beyond the scope of the present paper.

6.3 SDM – a quasi-Newton method

In this section we will 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 the term $f_k^2 = \mathcal{F}^T\mathcal{F}$. Denote $\mathcal{F} = (\mathcal{F}_x, \mathcal{F}_y)^T$. 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

$$\begin{aligned}\frac{1}{2}\nabla^2 f_k^2 &= \mathcal{F}_{\mathcal{P}}^T\mathcal{F}_{\mathcal{P}} + (\mathcal{F}_{\mathcal{P}}^T\mathcal{F}_t + \mathcal{F}_{\mathcal{P}t}^T\mathcal{F})\nabla t^T + \nabla t(\mathcal{F}_t^T\mathcal{F}_{\mathcal{P}} + \mathcal{F}^T\mathcal{F}_{\mathcal{P}t}) + (\mathcal{F}_{tt}^T\mathcal{F} + \mathcal{F}_t^T\mathcal{F}_t)\nabla t\nabla t^T \\ &\quad + \mathcal{F}_x\mathcal{F}_{x\,\mathcal{P}\mathcal{P}} + \mathcal{F}_y\mathcal{F}_{y\,\mathcal{P}\mathcal{P}} \\ &= \mathcal{F}_{\mathcal{P}}^T\mathcal{F}_{\mathcal{P}} + (\mathcal{F}_{\mathcal{P}}^T\mathcal{F}_t + \mathcal{F}_{\mathcal{P}t}^T\mathcal{F})\nabla t^T + \nabla t(\mathcal{F}_t^T\mathcal{F}_{\mathcal{P}} + \mathcal{F}^T\mathcal{F}_{\mathcal{P}t}) + (\mathcal{F}_{tt}^T\mathcal{F} + \mathcal{F}_t^T\mathcal{F}_t)\nabla t\nabla t^T \qquad (24)\end{aligned}$$

Here we used the fact that $\mathcal{F}_{x\,\mathcal{P}\mathcal{P}} = 0, \mathcal{F}_{y\,\mathcal{P}\mathcal{P}} = 0$, since $\mathcal{F}$ is linear in $\mathcal{P}$.

Again, ∇t stands for 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

$$\left(\mathcal{F}_{\mathcal{P}t}^T + \nabla t\mathcal{F}_{tt}^T\right)\mathcal{F} + \left(\mathcal{F}_{\mathcal{P}}^T + \nabla t\mathcal{F}_t^T\right)\mathcal{F}_t = 0. \qquad (25)$$

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 (24), we obtain the complete Hessian

$$\frac{1}{2}\nabla^2 f_k^2 = \mathcal{F}_{\mathcal{P}}^T\mathcal{F}_{\mathcal{P}} - \frac{(\mathcal{F}_{\mathcal{P}t}^T\mathcal{F} + \mathcal{F}_{\mathcal{P}}^T\mathcal{F}_t)(\mathcal{F}_t^T\mathcal{F}_{\mathcal{P}} + \mathcal{F}^T\mathcal{F}_{\mathcal{P}t})}{\mathcal{F}_{tt}^T\mathcal{F} + \mathcal{F}_t^T\mathcal{F}_t}.$$

We will now make a simplification by neglecting the term $\mathcal{F}_{\mathcal{P}t}^T\mathcal{F}$, i.e., setting it to zero. This results in an approximate Hessian $\widetilde{\nabla}^2 f_k^2$. To interpret this approximate Hessian geometrically, we introduce the arc length parameter s 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 $P(t)$. Since $\mathcal{F}_{ss}$ is the curvature vector of $P(t)$ at $P(t_k)$, we have $\mathcal{F}_{ss}^T\mathcal{F} = -d\kappa$, where $\kappa \geq 0$ is the curvature and d is the signed distance defined in Section 3. Hence, we obtain

$$\frac{1}{2}\widetilde{\nabla}^2 f_k^2 = \mathcal{F}_{\mathcal{P}}^T\mathcal{F}_{\mathcal{P}} - \frac{\mathcal{F}_{\mathcal{P}}^T T_k T_k^T\mathcal{F}_{\mathcal{P}}}{-d\kappa + 1}. \tag{26}$$

Noting that $T_kT_k^T + N_kN_k^T = I$, this equation is further rewritten as

$$\frac{1}{2}\widetilde{\nabla}^2 f_k^2 = \mathcal{F}_{\mathcal{P}}^T\left(I - T_kT_k^T\right)\mathcal{F}_{\mathcal{P}} - \frac{d\kappa\mathcal{F}_{\mathcal{P}}^T T_k T_k^T\mathcal{F}_{\mathcal{P}}}{-d\kappa + 1} = \mathcal{F}_{\mathcal{P}}^T N_k N_k^T\mathcal{F}_{\mathcal{P}} + \frac{d}{d-\rho}\mathcal{F}_{\mathcal{P}}^T T_k T_k^T\mathcal{F}_{\mathcal{P}}. \tag{27}$$

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 (21) 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 (27) 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

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

where $h_k(\mathcal{D})$ is defined in (9) and d_k, ρ_k are the corresponding distance and curvature radius for the k-th term. Hence, the quasi-Newton method uses a local quadratic model f_{QN} that is the same as the quadratic approximant $h_k(\mathcal{D})$ used by the SD error term before it is turned into the semi-definite form 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 [Kelley 1999]. 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 the Gauss-Newton iteration (or TDM) does, but 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 it uses an intuitively simple error metric that is adaptable to local curvature variation of a target shape.

Each term f_k^2 in the objective function in (14) is the squared distance function. Because the exact Hessian in the second order Taylor expansion of f_k^2 is modified in order to derive the SD error term, we conclude that, in general, the SD error term is *not* a second order approximation to the squared distance, even without the modification in (10) to make the error term positive semi-definite. This is in contrast to the previous squared distance minimization method by Pottmann et al [Pottmann et al. 2002] where the quadratic approximant in (6) is always a second order approximation to the squared distance function, before the modification to turn it into the positive semi-definite error term in (7).

6.4 Step size control

We have tested step size control on PDM, TDM, and SDM, using the Armijo rule [Kelley 1999]. 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 17 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 7(a) and 10(a); now both data sets are satisfactorily approximated by B-spline curves computed with TDM. For comparison, refer to Figures 7(c) and 10(c) to see the unacceptable fitting curves generated by TDM without step size control.

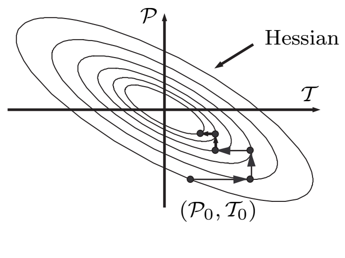


Fig. 16. The alternating minimization steps of PDM near a local minimum. $\mathcal{P}$ and $\mathcal{T}$ stand for the linear subspaces spanned by the control points and the data parameters, respectively.

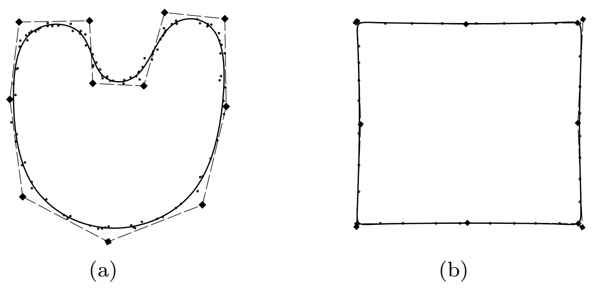


Fig. 17. TDM using the Armijo rule for step size control. (a) The fitting curve by TDM in 20 iterations for the data set in Figure 7(a); (b) The fitting curve by TDM in 20 iterations for the data set in Figure 10(a).

7. CONCLUDING REMARKS

PDM is widely used in practice for parametric curve and surface fitting [Farin 1997]. 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 a method often used for curve fitting in computer vision (e.g., [Blake and Isard 1998]). 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 empirically that SDM converges faster than PDM and that SDM is more stable than TDM. In addition, SDM is easy to implement and has similar per-iteration computation time to 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 geometrically motivated schemes, we have 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 uses a Gauss-Newton step 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 discussed in [Laurent-Gengoux and Mekhilef 1993; Goldenthal and Bercovier 2004]. Other significant problems include the analysis of the convergence rate of SDM 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 [Kass et al. 1988; Malladi et al. 1995; Osher and Fedkiw 2003; Sethian 1999].

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 current use of PDM as a predominant optimization method for surface fitting in graphics and CAD.

REFERENCES

Ambrosio, L. and Montegazza, C. 1998. Curvature and distance function from a manifold. *Journal of Geometric Analysis 8*, 723–748.

Bercovier, M. and Jacob, M. 1994. Minimization, constraints and composite bézier curves. *Computer Aided Geometric Design 11*, 533–563.

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

Blake, A. and Isard, M. 1998. *Active Contours.* Springer, New York.

Djebali, M., Melkemi, M., and Sapidis, N. 2002. 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.* 127–138.

Farin, G. 1997. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*, 4th ed. Academic Press, New York.

Forsey, D. R. and Bartels, R. H. 1995. Surface fitting with hierarchical splines. *ACM Transactions on Graphics 14*, 134–161.

Goldenthal, R. and Bercovier, M. 2004. Spline curve approximation and design by optimal control over the knots. *Computing 72*, 53–64.

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

Greiner, G., Kolb, A., and Riepl, A. 2002. Scattered data interpolation using data dependant optimization techniques. *Graphical Models 64*, 1–18.

Haber, J., Zeilfelder, F., Davydov, O., and Seidel, H. P. 2001. Smooth approximation and rendering of large scattered data sets. In *Proceedings of Visualization'01*. 341–348.

Hoppe, H. 1996. Progressive meshes. In *Proceedings of SIGGRAPH'96*. 99–108.

Hoppe, H., DeRose, T., Duchamp, T., Halstead, M., Jin, H., McDonald, J., Schweitzer, J., and Stuetzle, W. 1994. Piecewise smooth surface reconstruction. In *Proceedings of SIGGRAPH'94*. 295–302.

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

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

Hu, S. M. and Wallner, J. 2005. second order algorithm for orthogonal projection onto curves and surfaces. *Computer Aided Geometric Design 22*, 251–260.

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

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

Laurent-Gengoux, P. and Mekhilef, M. 1993. Optimization of a nurbs representation. *Computer-aided Design 25*, 699–710.

Lee, E. T. Y. 1989. Choosing nodes in parametric curve interpolation. *Computer-aided Design 21*, 363–370.

Luenberger, D. 1984. *Linear and Nonlinear Programming*. Addision-Wesley.

Ma, W. Y. and Kruth, J. P. 1995. Parameterization of randomly measured points for least squares fitting of b-spline curves and surfaces. *Computer-aided Design 27*, 663–675.

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

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

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

Pavlidis, T. 1983. Curve fitting with conic splines. *ACM Transactions on Graphics 2*, 1–31.

Plass, M. and Stone, M. 1983. Curve-fitting with piecewise parametric cubics. *Computer Graphics 17,* 3, 229–239.

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

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

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

Pratt, V. 1985. Techniques for conic splines. In *Proceedings of SIGGRAPH'85*. 151–160.

Ruhe, A. and Wedin, P. 1980. Algorithms for separable nonlinear least squares problems. *SIAM Review 22*, 318–337.

Sarkar, B. and Menq, C.-H. 1991. Parameter optimization in approximating curves and surfaces to measurement data. *Computer Aided Geometric Design 8*, 267–290.

Saux, E. and Daniel, M. 2003. An improved hoschek intrinsic parametrization. *Computer Aided Geometric Design 20*, 513–521.

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

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

Taubin, G. 2002. Dual mesh resampling. *Graphical Models 64*, 94–113.

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

Wang, X. C. and Phillips, C. 2002. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer animation*. 129 – 138.

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

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