Design and Analysis of Optimization Methods for Subdivision Surface Fitting

Kin-Shing D. Cheng, Wenping Wang, Hong Qin, Kwan-Yee K. Wong, Huaiping Yang and Yang Liu

***Abstract*— We present a complete framework for computing a subdivision surface to approximate unorganized point sample data, which is a *separable* nonlinear least squares problem. We study the convergence and stability of three geometrically-motivated optimization schemes and reveal their intrinsic relations with standard methods for constrained nonlinear optimization. A commonly-used method in graphics, called *point distance minimization*, is shown to use a variant of the gradient descent step and thus has only linear convergence. The second method, called *tangent distance minimization*, which is well-known in computer vision, is shown to use the Gauss-Newton step, and thus demonstrates near quadratic convergence for zero residual problems but may not converge otherwise. Finally, we show that an optimization scheme called *squared distance minimization*, recently proposed by Pottmann et al., can be derived from the Newton method. Hence, with proper regularization, tangent distance minimization and squared distance minimization are more efficient than point distance minimization. We also investigate the effects of two step size control methods – Levenberg-Marquardt regularization and the Armijo rule – on the convergence stability and efficiency of the above optimization schemes.**

***Index Terms*— subdivision surface, fitting, optimization, squared distance.**

I. Introduction

Shapes represented by 3D unorganized geometric points are now readily available as the widespread use of 3D scanning devices for shape acquisition becomes a common practice. For geometric processing, we often need to fit a surface to such point samples. Subdivision surface is a preferred representation because of its compactness and ability to accommodate general control mesh connectivity. From the optimization point of view, the surface fitting problem is a *separable* nonlinear least squares problem. In principle, we need to minimize an objective function consisting of a geometric error term and a smoothing term. The geometric error term can be approximated by different functions measuring the squared distance between a fitting surface and the target shape. These error functions lead to different local quadratic models of the objective function and result in different optimization efficiency.

K.S. D. Cheng, W. Wang, K.-Y. K. Wong, H. Yang and Y. Liu are with the Department of Computer Science, the University of Hong Kong, Pokfulam Road, Hong Kong. E-mail: {ksdcheng, wenping, kykwong, hpyang, yliu}@cs.hku.hk

H. Qin is with the Department of Computer Science, State University of New York at Stony Brook, Stony Brook, New York, U.S.A. E-mail: qin@cs.sunysb.edu

A. Our Work

This paper is an extension to our work [1] presented at Pacific Graphics 2004. We present a general and complete framework for computing a subdivision surface via geometric point fitting. Specifically, we focus on convergence analysis of optimization schemes for solving this surface fitting problem. Suppose that the shape to be fitted, called the *target shape*, is defined by unorganized data points. To start the fitting process, an initial subdivision surface is first generated from the point cloud by applying the dual marching cubes method [2]. The control points are then optimized by minimizing an objective function through iterative quadratic minimization. New control points are added progressively in order to better capture the features of the target shape; this gives rise to a multi-staged optimization problem. Although we use Loop's subdivision surface [3] to handle triangular meshes, the proposed algorithmic flow can naturally apply to other types of subdivision surfaces based on linear schemes.

We consider three geometrically-motivated methods in this paper. The first method is based on a point-to-point distance error metric, thus called *point distance minimization* [4]. This method has been used predominantly for decades in graphics

and CAD/CAM for curve and surface fitting. It has monotonic descent but converges slowly. The second method uses a point-to-tangent distance error metric, thus called *tangent distance minimization*. This has been used in the computer vision field for model registration [5]. In general, it converges much faster than point distance minimization, but is unstable for a target shape with sharp features. The third method, called *squared distance minimization*, has been recently proposed by Pottmann et al. [6] for B-spline curve and surface fitting.

Our contribution is the systematic study on the convergence behaviors of the above optimization methods in the setting of subdivision surface fitting. The derivations of these methods in the literature are only based on geometric arguments. We establish their equivalences to three well-understood optimization methods – *steepest descent, the Gauss-Newton method, and the Newton method*, respectively – for nonlinear constrained optimization.

This connection to well-known optimization techniques helps explain or understand the practical behaviors of these three methods in surface fitting. The steepest descent method is well known to have a linear convergence rate, thus explaining the slow convergence of point distance minimization. The Gauss-Newton method has quadratic convergence for zero residual problems, but it converges only linearly or may not converge at all for problems with a large residue – this also conforms with the observed behavior of tangent distance minimization. Squared distance minimization is a simple geometric incarnation of the Newton method, thus explaining its observed superior convergence in the general case. Based on this interpretation, we apply the Levenberg-Marquardt regularization to tangent distance minimization and squared distance minimization to ensure monotonic decrease of the objective function, thus improving the robustness of their convergence. Details about these basic optimization techniques can be found in standard texts on optimization (e.g., [7], [8]).

B. Related Work

The problem of computing a compact surface representation of a target shape defined by unorganized data points has many applications in computer graphics, CAD, and computer vision. Numerous approaches have been proposed over the past decades. These include fitting methods based on optimization [9], [10], [11], [12], [13], [14], active surface methods [15], [16], [17], and other approaches [18], [19], [20], [21]. Among parametric surfaces, subdivision surface has gained popularity because of its ability to deal with general object topology as well as arbitrary connectivity of the control mesh [22]. Some works are closely related to the problem addressed in this paper. In Hoppe et al.'s method [9], a mesh surface is computed to fit unorganized points via optimization of an energy function. In [11], [12], a (non-iterative) linear least squares problem is solved to produce a fitting surface composed of B-spline surfaces for quadrangle patches and Catmull-Clark surfaces for extraordinary corner patches. In [13], a Loop's subdivision surface is computed to fit a mesh surface by iterative quadratic optimization, using a simplification of the input mesh as the initial control mesh. All these methods use the *point distance (PD) error function* to approximate the geometric error between the fitting surface and the target shape in each iteration. This is essentially the parameter correction method by Hoschek [4], but we will refer this scheme as *point distance minimization*, or PDM for short. PDM belongs to the category of the alternating method [23], [7] for solving separable nonlinear least squares problems; it has only linear convergence rate and converges slowly in practice, as we will demonstrate later in this paper.

Tangent distance minimization, abbreviated as TDM, uses another error function, called *the TD error term*, based on a point-to-tangent distance, and it has been used in computer vision for 3D model registration by Chen and Medioni [5] and active curve fitting by Blake [15]. In the extension to their work in [13], Marinov and Kobbelt apply a combination of the PD error term and TD error term to improve the efficiency of their surface fitting method [24], without considering the issues of convergence analysis and step size control.

Squared distance minimization, or SDM for short, uses the so-called *squared distance* (SD) error term, which is first considered in [25] and further investigated in detail from a geometric point of view by Pottmann et al. [26] with applications to shape fitting with B-spline curves and surfaces [6], [27]. However, they did not give convergence analysis or consider step size control for ensuring convergence.

In this work we emphasize on the convergence

behaviors of PDM, TDM, and SDM in the setting of subdivision surface fitting from an optimization point of view, and investigate the step size control schemes as applied to these methods. Our results provide new and useful insights to the practical surface fitting procedures used in computer graphics. A variant of the SDM method, along with TDM and PDM, has been recently studied in [28]. The connections of our work here to that of [28], as well as their differences, are elaborated in Section V.

II. Fitting Algorithm

A. Problem Formulation

Suppose that input data points, defining a *target shape* Γ, are sampled from an underlying *target surface* Γ_T, which is a manifold surface of arbitrary genus. For convergence analysis, the second order differentiability of Γ_T is assumed. Our goal is to reconstruct the surface Γ_T by computing a subdivision surface from Γ.

Let $P(s,t)$ be a local parameterization of a fitting surface $\mathcal{S}$. The fitting error between $\mathcal{S}$ and the target surface Γ_T is measured by the sum of squared distances from a set of dense sample points on $\mathcal{S}$ to Γ_T. Denote these sample points by $S_k = P(s_k, t_k)$, which are linear combinations of the control points P_i, $k = 1, 2 \ldots, m$, of a Loop's subdivision surface $\mathcal{S}$. We assume that m is much greater than n, the number of control points, so that the fitting problem is properly constrained. Let $V_k = V(u_k, v_k) \in \Gamma_T$ be the closest point from the sample point S_k on the fitting $\mathcal{S}$ to the target surface Γ_T. The fitting error at S_k is then given by $f_k = ||S_k - V(u_k, v_k)||$. The point $V(u_k, v_k) \in \Gamma_T$ is called the *foot point* of S_k.

Denote $\mathcal{P} = \{P_i\}_{i=1}^n$ and $\mathcal{U} = \{(u_k, v_k)\}_{k=1}^m$. The control points $\mathcal{P}$ of the best fitting surface $\mathcal{S}$ are computed by solving the following optimization problem,

$$\min F(\mathcal{P},\mathcal{U}) = F_e(\mathcal{P},\mathcal{U}) + F_s(\mathcal{P}), \qquad (1)$$

where $F_e = \frac{1}{2}\sum_{k=1}^m f_k^2 = \frac{1}{2}\sum_{k=1}^m \|S_k - V(u_k, v_k)\|^2$ is the L_2 fitting error, and $F_s(\mathcal{P})$ is a regularization term that is a quadratic function of the control points $\mathcal{P}$. The variables in the function F are control points $\mathcal{P} = \{P_i\}$, and the parameter values $\mathcal{U} = \{(u_k, v_k)\}$. Clearly, F is quadratic in $\mathcal{P}$, but is, in general, a highly nonlinear function of $\mathcal{U}$.

We may treat $\mathcal{P}$ as *basic variables* and $\mathcal{U}$ as *dependent variables*, since it is required that $V(u_k, v_k)$ be the foot point of the sample point S_k, which is a linear function of $\mathcal{P}$. This leads to the following commonly-used optimization strategy in surface fitting: given an initial fitting surface with the sample points S_k, one first computes that foot points $V(u_k, v_k)$. With $\mathcal{U}$ being known, one updates the control points $\mathcal{P}$ by minimizing $F(\mathcal{P},\mathcal{U})$ over the control points $\mathcal{P}$; this is done by solving a linear system of equation, since $F(\mathcal{P},\mathcal{U})$ is quadratic in $\mathcal{P}$ and $\mathcal{U}$ has been fixed. The above two steps of foot-point computation and control point computation are iterated to further improve the fitting error until convergence.

This commonly-used iterative fitting method is called the *point distance minimization* (PDM) method. Because of the separate treatments of variables $\mathcal{P}$ and $\mathcal{U}$, the optimization problem defined in (1) is also called a separable variable problem. We will also consider two other more efficient local optimization schemes based on the same framework, i.e., TDM and SDM.

An initial fitting surface is needed to start the above iterative procedure. It is natural to begin with an initial fitting surface having the same topology as the target shape and a simple control mesh. Control points need to be inserted progressively as optimization proceeds to make the fitting surface better capture the fine features of the target shape. This means that we need to consider a multi-staged optimization problem, with proper scheduling of adding new control points.

B. Algorithmic Flow

Our proposed fitting procedure has the following main steps:

(1) *Normalization*: Normalize the target shape by uniform scaling to fit it within the cube $[0, 1]^3$.
(2) *Precomputation*: Pre-compute the distance field, as well as the tangential and curvature information of the target surface Γ for setting up error terms.
(3) *Initial mesh*: Compute an initial control mesh using the dual marching cubes method [2].
(4) *Points sampling*: Generate m dense sample points S_k^0 on the current fitting surface using the method in [29], [30].
(5) *Error function setup*: Use the sample points generated in step (4) to set up the error function

$$F_L(\mathcal{P}) = \frac{1}{m}\sum_{k=1}^m F_{L,k}(\mathcal{P}) + F_s(\mathcal{P}), \qquad (2)$$

where $F_{L,k}(\mathcal{P})$ is one of the three error terms to be introduced in Section II-D.

(6) *Minimization*: Update control points by minimizing the quadratic function $F_L(\mathcal{P})$. This is done by solving a linear system of equations using the conjugate gradient (CG) method.

(7) *Error evaluation*:
Sample new points S_k^1 on the fitting surface with updated control points, and compute their foot points V_k^1. Next compute the maximum error E_m and the root-mean-square error E_{rms}, where

$$E_m = \max_k \{||S_k^1 - V_k^1||_2\}.$$

and

$$E_{rms} = \left[\frac{1}{m}\sum_k ||S_k^1 - V_k^1||_2^2\right]^{\frac{1}{2}}.$$

Here $||S_k^1 - V_k^1||_2$ is called the local error. The algorithm is terminated, if E_m or E_{rms} falls below a pre-specified error threshold, or the number of iterations reaches some limit. If the fitting error has been reduced significantly in this step by the current iteration (but still larger than the threshold), go to Step 4 to start the next iteration. Otherwise, go to Step 8.

(8) *Refinement*: New control points are inserted in the regions of large fitting error E_m. Go to step 4 for the next iteration.

The flowchart is shown in Fig. 1. Note that remeshing could be used in step (3) to reduce the number of extraordinary vertices whose valences are not six.

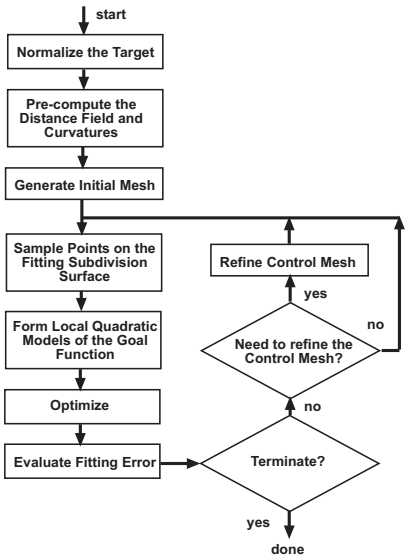


Fig. 1. Work flow.

C. Preprocessing and Initialization

To quickly obtain the foot points V_k^0 of the sample points S_k^0, we pre-compute an adaptive distance field of the target shape using the idea in [31]. The pre-computed information, such as distances and foot points, are stored at the corners of adaptive octtree cells. During the optimization process, the foot point of a sample point S_k^0 is computed by trilinear interpolation from the stored values at the corners of the smallest cell containing S_k^0.

The normal vector and the principal curvatures of the target surface Γ_T at V_k are also pre-computed from the point cloud Γ for setting up the TD error functions and the SD error functions. For a given target point V_k, we first find its neighboring points $V_{k,j}$. A local coordinate frame at V_k can then be defined by the principal curvature directions and the normal direction which are the eigenvectors of the covariance matrix $\mathcal{CV}$ given by

$$\mathcal{CV} = \sum_j (V_{k,j} - V_{k,c})(V_{k,j} - V_{k,c})^T,$$

where $V_{k,c}$ is the centroid of V_k's neighboring points. A polynomial $z = k_1x^2 + k_2y^2$ is then fitted to the neighboring points $V_{n(i,j)}$ in this local frame and the principal curvatures are simply set to $2k_1$ and $2k_2$.

We use the dual marching cubes method devised by Schaefer and Warren [2] to compute the initial control mesh from the dual grid of cells of an adaptive octree partition of the target data set. The meshes generated by this method are topologically faithful, adaptive to curvature or features of the target shape, and therefore more desirable than those generated by the ordinary marching cubes method [32], which tends to produce too many fragmented triangles, even in flat regions of the target shape, as observed in [1].

D. Error Functions

1) PD Error Term: Suppose that the fitting surface has been sampled by the points S_k^0, whose foot points are V_k^0. Let S_k be the variable points of S_k^0, depending on the variable control points $\mathcal{P}$. A simple error term is given by

$$f_{PD,k}^2 = ||S_k - V_k^0||^2, \tag{3}$$

which is called the *point distance* (PD) error term because $f_{PD,k}$ is the distance between S_k and V_k^0. The optimization scheme resulting from using the PD error term for $F_{L,k}(\mathcal{P})$ in (2) is called *point distance minimization* or *PDM*.

2) TD Error Term: When the target surface Γ_T is relatively flat around V_k^0, the tangent plane of Γ_T at V_k^0 is a good approximation to Γ_T in a neighborhood of V_k^0. This observation leads to the so-called *tangent distance (TD) error term*, defined as

$$f_{TD,k}^2 = |(S_k - V_k^0)^T N_k|^2, \tag{4}$$

where N_k is a constant unit normal vector to Γ at V_k^0. Clearly, $f_{TD,k}^2$ is the squared distance from S_k to the tangent plane $(X - V_k^0)N_k = 0$. The optimization scheme resulting from using the TD error term is called *tangent distance minimization* or *TDM*.

3) SD error term : Now we consider the second order approximation to the squared distance from S_k to Γ_T, as proposed in [6], [25]. Let d be the signed distance such that $|d| = ||S_k^0 - V_k^0||$. Let ρ_1 and ρ_2 denote the principal curvature radii of Γ_T at V_k^0, associated with the principal unit direction vectors T_1 and T_2. We make the following convention on the signs of d and ρ_i, $i = 1, 2$. Suppose that N is a normal vector to the target surface Γ_T. If the sample point S_k is on the side of Γ_T pointed to by N, i.e., $N \cdot (S_k^0 - V_k^0) > 0$, then $d > 0$; otherwise, $d \leq 0$. Similarly, if the curvature center in the normal section along the tangent vector T_i is on the side of Γ_T pointed to by N, then $\rho_i > 0$; otherwise, $\rho_i \leq 0$, $i = 1, 2$.

Since V_k^0 is locally the closest point from S_k^0 to Γ_T, then we have $|d| < |\rho_i|$ when d and ρ_i have the same sign. It can be shown [25], [26] that the second order approximation to the true squared distance is

$$\begin{aligned} \hat{f}_{SD,k}^2 &= \alpha_1[(S_k - V_k^0)^T T_1]^2 + \alpha_2[(S_k - V_k^0)^T T_2]^2 \\ &+ [(S_k - V_k^0)^T N]^2, \end{aligned} \tag{5}$$

where $\alpha_1 = d/(d - \rho_1)$ and $\alpha_2 = d/(d - \rho_2)$. The coefficient α_1 (or α_2) becomes negative if ρ_1 (or ρ_2) and d have the same sign. Denote $[\alpha]_+ = \max\{\alpha, 0\}$. To have a non-negative error function, $\hat{f}_{SD,k}^2$ is modified to

$$\begin{aligned} f_{SD,k}^2 &= [\alpha_1]_+[(S_k - V_k^0)^T T_1]^2 + [\alpha_2]_+[(S_k - V_k^0)^T T_2]^2 \\ &+ [(S_k - V_k^0)^T N]^2. \end{aligned} \tag{6}$$

This term $f_{SD,k}^2$ is called the *squared distance (SD) error term*, since it is derived from a second order approximation to the true squared distance. The ellipsoid in Fig. 2 shows an iso-distance surface of the SD error term for the case $\alpha_1 > 0$ and $\alpha_2 > 0$; the SD error term reduces to the TD error term if $\alpha_1 = 0$ and $\alpha_2 = 0$. The optimization scheme resulting from using the SD error term is called *squared distance minimization* or *SDM*.

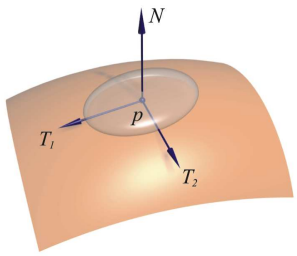


Fig. 2. An iso-surface of SD error term.

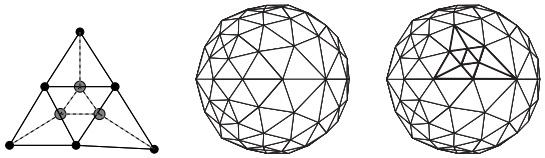


Fig. 3. Triangle split.

E. Smoothing Term

The smoothing term in (2) is defined as

$$F_s = \frac{\lambda}{n}\sum_{i=1}^{n} W(P_i)^T W(P_i),$$

where P_i, $i = 1, 2, \ldots, n$, are the control points and $W(\cdot)$ is a discrete version of Laplacian [33]. It is difficult to automatically choose an appropriate value of the coefficient λ [34]; therefore, most existing methods choose λ in a heuristic manner. Likewise, in our test examples, the initial value for λ is set to 0.001 at the beginning, and is reduced gradually as the optimization proceeds.

We use the conjugate gradient method to minimize the error function (2). This iterative solver is terminated if the relative error improvement is less than 10^{-6} or the number of iterations reaches 200. These parameters produce satisfactory results in our experiments.

F. Local Refinement

When the fitting error remains large due to the insufficient number of control points, new control

points are inserted to triangles that have large local errors and split the triangles in a 1-to-4 manner (see Fig. 3). This step is called *local refinement*. To avoid undesirable T-vertices, the neighboring triangles are also split, following the Red-Green splitting scheme [35].

III. Optimization Properties

In this section we shall establish the connection between the three optimization techniques introduced so far – PDM, TDM and SDM – and standard optimization techniques in optimization theory. Due to space limit, we shall only discuss some basic facts that we are going to use directly; the reader is referred to standard texts (e.g., [7], [8]) for detailed introduction to optimization theory.

A. Basics of Optimization

First consider the Newton method. Given an objective function $f(x) : \mathcal{R}^N \to \mathcal{R}$ and the current variable value x_c near a local minimum of $f(x)$, the next iterate x_+ is the minimizer of the local quadratic model $m_c(x)$ of $f(x)$ about x_c, where

$$\begin{aligned} m_c(x) &= f(x_c) + \nabla f(x_c)^T (x - x_c) \\ &+ \tfrac{1}{2}(x - x_c)^T \nabla^2 f(x_c)(x - x_c), \end{aligned} \quad (7)$$

which is the second order Taylor expansion of $f(x)$ at x_c. If $\nabla^2 f(x_c)$ is positive definite, x_+ is the unique solution of the equation

$$0 = \nabla m_c(x_+) = \nabla f(x_c) + \nabla^2 f(x_c)(x_+ - x_c). \quad (8)$$

The Newton method has local quadratic convergence.

Various iterative schemes can be obtained by replacing the Hessian $\nabla^2 f(x_c)$ in Eqn. (7) by different estimates of it. Among these, the Gauss-Newton method is preferred for solving nonlinear least squares problems [8]. Consider a nonlinear least squares problem,

$$f(x) = \tfrac{1}{2} \sum_k r_k(x)^2. \quad (9)$$

The Hessian of $f(x)$ is

$$\nabla^2 f(x) = \sum_k \nabla r_k(x) \nabla r_k(x)^T + \sum_k r_k(x) \nabla^2 r_k(x). \quad (10)$$

In the Gauss-Newton method, one discards the second-order term to use the approximate Hessian

$$\tilde{\nabla}^2 f(x) = \sum_k \nabla r_k(x) \nabla r_k(x)^T \quad (11)$$

to replace the Hessian $\nabla^2 f(x_c)$ in (8) to compute the next iterate x_+. The Gauss-Newton method has quadratic convergence for zero residual problems because the discarded term $\sum_k r_k(x) \nabla^2 r_k(x)$ is negligible in those cases. However, the Gauss-Newton method only converges linearly or may not converge at all for large residual problems because the Hessian is poorly approximated in these situations [8].

B. Surface Fitting as a Separable Problem

The surface fitting problem can be formulated as a separable nonlinear least squares problem with the following objective function,

$$\begin{aligned} F(\mathcal{P}, \mathcal{U}) &= \sum_{k=1}^{m} F_k(\mathcal{P}, \mathcal{U}) + F_s(\mathcal{P}) \\ &= \tfrac{1}{2} \sum_{k=1}^{m} \| S_k(\mathcal{P}) - V(u_k, v_k) \|^2 + F_s(\mathcal{P}), \end{aligned} \quad (12)$$

where S_k is a linear function of the control points $\mathcal{P}$. To simplify notation, we denote $S = S_k(\mathcal{P})$, $(u, v) = (u_k, v_k)$, and $E = S - V(u, v)$. Then $F_k = \frac{1}{2} E^T E$. In the following, we will use $\mathcal{P}$, u and v as subscripts to denote derivatives with respect to these variables.

Since $V(u, v)$ is the foot point of the sample point S to the target surface Γ_T, E is perpendicular to the tangent plane of Γ_T at $V(u, v)$. It follows that

$$\begin{aligned} (S - V(u, v))^T V_u = E^T V_u = 0, \\ (S - V(u, v))^T V_v = E^T V_v = 0. \end{aligned} \quad (13)$$

These two equations are constraints tieing the variables (u, v) to the control points $\mathcal{P}$. In fact, the two equations in (13) are also necessary conditions for the objective function F to have a local minimum, since the partial derivatives of F_k with respect to u and v are

$$\begin{aligned} \partial F_k / \partial u &= (S - V(u, v))^T E_u, \\ \partial F_k / \partial v &= (S - V(u, v))^T E_v, \end{aligned}$$

and noting that $E_u = -V_u$ and $E_v = -V_v$. Therefore the introduction of these constraints does not

preclude any minimizer of the original optimization problem.

We first discuss the convergence behavior of PDM. PDM performs the following two alternating steps iteratively: (1) solving a linear system of equations to obtain the variable control points P_i while fixing the parameter values (u_k, v_k); and (2) computing foot points to find the (u_k, v_k) with the fixed control points P_i. The first step decreases the value of the objective function but moves away from the constraints (13). The second step moves the iterate back to give a feasible point satisfying the constraints (13). This is, in fact, the *alternating method* for solving a separable and constrained nonlinear problem, and is known to have linear convergence [7].

It can be shown that TDM uses a Gauss-Newton step and SDM is equivalent to the Newton method. The proofs of these facts are given in appendices in order to have a better flow of discussion.

C. Regularization and Step Size Control

The Gauss-Newton method works poorly for large residual problems, so the Levenberg-Marquardt method (the LM method), which is a regularized version of the Gauss-Newton method, is normally used [8]. The essence of the LM method is that the local quadratic model is trusted only within a small enough neighborhood of the current point x_c, defined by the constraint $\|s\| \leq \triangle_k$, where s is the step. The selection of the value of $\triangle_k$ depends on the degree of the agreement between the local model and the objective function. The optimality condition for this constrained optimization gives

$$(\nabla^2 f(x_c) + \nu_c I)s = -\nabla f(x_c). \tag{14}$$

The Hessian is then approximated as

$$\nabla^2 f(x) \approx \sum_k \nabla r_k(x) \nabla r_k(x)^T + \nu_c I. \tag{15}$$

In other words, s is computed by replacing $\nabla^2 f(x_c)$ in (8) by $\sum_k \nabla r_k(x) \nabla r_k(x)^T + \nu_c I$.

In LM regularization the gain ratio is monitored, which is the ratio of the actual decrease in the objective function to the decrease predicted by the local model [36]. If the gain ratio is small, meaning that the current model is a poor approximation to the goal function, ν_c is increased so that the next step will be closer to the steepest descent direction and the step size is reduced. If the gain ratio is large, meaning that the current model is a good approximation to the goal function, ν_c is decreased so that the next step will be closer to a Gauss-Newton step. By monitoring the agreement between the local model and the actual objective function, this approach tries to get the advantages of both the steepest descent method and the Gauss-Newton method. For all values of ν_c, called the LM parameter, the coefficient matrix is positive definite. Note that both the direction and the step size are modified in the LM method. In the same vein, we will also consider the LM regularization of SDM in next section.

Besides the LM method, the Armijo method [8] is also commonly used for guaranteeing convergence. After the direction of a step s has been determined by a particular method (PDM, SDM or TDM), the step size is decided by performing a line search. Although the Armijo method improves convergence, extra goal function and gradient evaluations are required and these increase the computational time. Different from the LM method, only the step size is modified in the Armijo method. The effectiveness of the LM method and the Armijo method will be investigated in the experiments in Section IV.

IV. Experiments

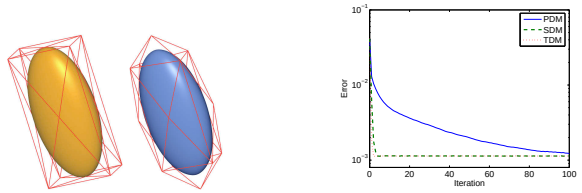
We will present test examples computed by PDM, TDM and SDM to observe and confirm the convergence behavior of the three methods, as influenced by initial control mesh specification, smoothness term and step control methods (i.e., the LM method and the Armijo rule.) We will also present examples of subdivision surface reconstruction from complex target shapes. All experiments were run on a PC with Intel Xeon 2.8 GHz CPU and 2.00 GB RAM. All data sets are first scaled uniformly to fit into a rectangular box with the longest side equals to 1.0.

A. Initial mesh and sharp feature

We first consider applying PDM, TDM and SDM to a data set with two different initial control meshes, without the smoothness term or regularization. Here no new control points are added during optimization.

Example 1: (Refer to Fig. 4.) The target shape is an ellipsoid with semi-principal axes being 0.25, 0.5

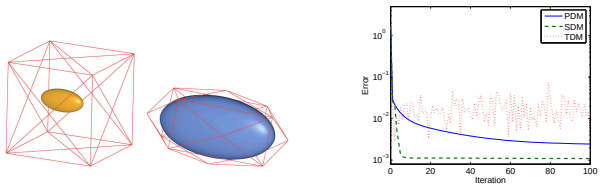
and 1.0 and the initial mesh has 14 control points, as shown in Fig. 4. The optimized surface by SDM is shown in Fig. 4(a); the optimized surface by TDM is similar to that by SDM and therefore not shown. The error curves are shown in Fig. 4(b).



(a) (b)

Fig. 4. (Example 1). (a) Left: Target shape and initial mesh; Right: Optimized mesh and surface by SDM. (b) Error curves.

Example 2: (Refer to Fig. 5.) The same target shape is used here as in Example 1, but the initial control mesh is now farther away from the target shape, as shown in Fig. 5(a). The surface reconstructed with SDM is also shown. The error curves of PDM, TDM and SDM are shown in Fig. 5(b). TDM does not converge for this data set.

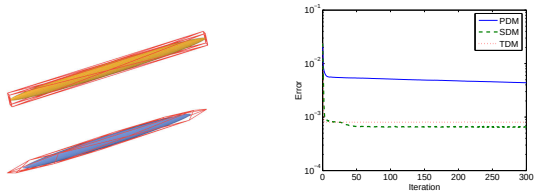


(a) (b)

Fig. 5. (Example 2). (a) Left: Target shape and initial mesh; Right: Optimized mesh and surface by SDM. (b) Error curves.

Discussion: Examples 1 and 2 show that SDM converges much faster than PDM. TDM has similar convergence rate as SDM, but may easily become unstable if the initial mesh is far away from the target shape. That is because TDM, using a Gauss-Newton step, discards from the Hessian the part $r(x)\nabla^2 r(x)$ that is related to the curvature and residue; here $r(x)$ reflects the distance between the initial fitting surface and the target shape. Therefore TDM should always be used with LM regularization for stable convergence, as will be seen shortly.

Example 3: (Refer to Fig. 6.) Here we consider an elongated ellipsoid with two sharp ends, with semi-principal axes being 0.125, 0.25 and 4.0, as shown in Fig. 6(a) with the initial mesh. We use this target shape to test PDM, TDM and SDM in the presence of sharp features. The optimized subdivision surface by SDM is also shown. The optimized surface by TDM is similar to that by SDM and is therefore not shown. The error curves of PDM, TDM and SDM are shown in Fig. 6(b). We see that SDM and TDM have similar convergence behaviors, and PDM converges to a poor local minimum with a larger residual error.



(a) (b)

Fig. 6. (Example 3). (a) Top: Target shape and initial mesh; Bottom: Optimized mesh and surface by SDM. (b) Error curves.

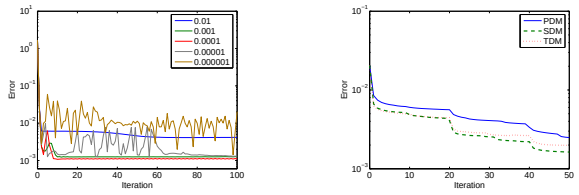
B. Smoothness Term and Multi-stage Optimization

In this section we consider the effects of the smoothness term on convergence behaviors and insertion of new control points.

Example 4: (Refer to Fig. 5(a) and Fig. 7(a).) The data set and initial control mesh used here are the same as those in Example 2 (see Fig. 5(a)), where TDM fails to converge. Now we want to observe whether the introduction of a smoothness term can make the convergence of TDM stable. The smoothness term is defined in Section II-D. We tested different values of coefficients $\lambda = 10^{-i}$, $i = 2, 3, 4, 5, 6$, resulting in the error curves of TDM shown in Fig. 7(a).

Discussion: In this example, $\lambda = 0.001$ or 0.0001 leads to stable convergence with a small fitting error. Bigger values of λ make the surface too "stiff", giving large fitting errors, while smaller values of λ fail to make TDM stable. We remark that, except for trial-and-error or methods based on user assistance, there is currently no commonly accepted general scheme that can automatically determine the coefficient of the smoothness term in the context of curve or surface fitting.

Example 5: (Refer to Fig. 7(b).) Here we will test the effects of adaptively adding new control points and reducing the smoothness coefficient progressively at different stages of the fitting process, using PDM, TDM and SDM. The target shape is the ball joint model containing $137,062$ points (Fig. 9) and the initial control mesh has 128 control



(a) (b)

Fig. 7. (a) Error of TDM for Example 4. (b) Error for Example 5.

points. The initial smoothness coefficient is 0.01. Throughout optimization, new control points are inserted at regions of large errors progressively. The final number of control points at the 50-th iteration is 202, 151 and 199 for PDM, TDM and SDM, respectively. The smoothness coefficient is decreased to 0.001 and 0.0001 at the 20^{th} and the 40^{th} iterations, respectively. Fig. 7(b) shows the error curves of PDM, TDM and SDM. Obvious reduction of the fitting error can be observed due to the control points insertion and the smoothing coefficient adjustment. During each stage between the insertions of control points, SDM and TDM are again more efficient than PDM.

C. LM Regularization and the Armijo Rule

Since TDM uses a Gauss-Newton step, it exhibits instability if used without any regularization. In this section we apply LM regularization to TDM, and also to SDM, to observe its effect on the convergence of the two methods.

Example 6: (LM regularization of TDM. Refer to Fig. 4(a) and Fig. 8(a)). Here the target shape and initial control mesh are the same as in Example 1, as shown in Fig. 4(a). No smoothing term is used. Let TDMLM denote TDM with LM regularization. Fig. 8(a) shows the error curves for TDMLM and TDM. TDMLM takes 0.551s to get E_{rms} smaller than 0.002. The error curves in Fig. 8(a) indicate that TDMLM delivers both fast convergence and monotonic descent of the fitting error, while the error of TDM is not monotonically decreasing.

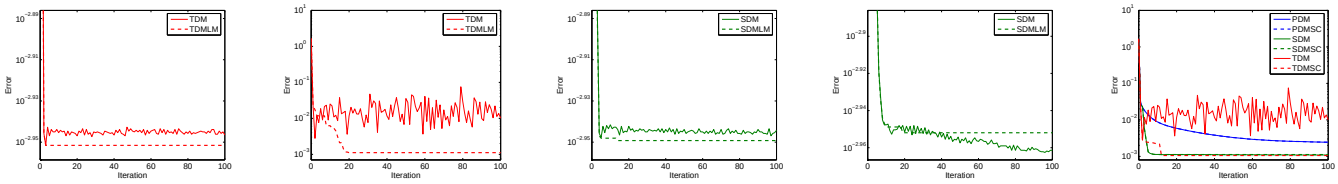
Example 7: (LM regularization of TDM. Refer to Fig. 5(a) and Fig. 8(b).) Here the target shape and the initial control mesh are the same as used in Example 2, as shown in Fig. 5(a). Recall that TDM fails to converge in that example. Now we apply TDMLM without smoothing term. Fig. 8(b) shows the error curves of TDMLM and TDM. Clearly, TDMLM converges fast and stably. TDMLM takes 1.593s to have an E_{rms} smaller than 0.002.

Example 8: (LM regularization of SDM. Refer to Fig. 4(a) and Fig. 8(c).) Here we apply LM regularization to SDM for the same target shape and initial mesh as in Example 6, as shown in Fig. 4(a). The regularized SDM will be denoted by SDMLM. Fig. 8(c) shows the error curves of SDM and SDMLM, where the SDM error curve here is the re-scaled version of the same SDM error curve in Fig. 4. It can be seen that the SDM error curve is actually not monotonically descending but that of SDMLM is. SDMLM takes 1.011s to get E_{rms} smaller than 0.002.

Example 9: (LM regularization of SDM. Refer to Fig. 5(a) and Fig. 8(d).) In this example SDM and SDMLM are applied to the same target shape and initial control mesh as in Example 7, as shown in Fig. 5(a). The error curves of SDM and SDMLM are shown in Fig. 8(d). Again, we see that SDMLM leads to more stable convergence than SDM without regularization.

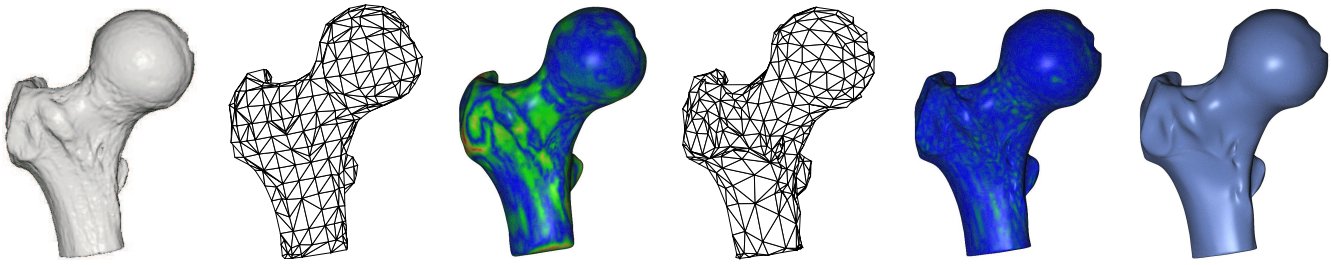
Example 10: (Refer to Fig. 5 and Fig. 8(e).) In this example step size control using the Armijo rule is applied to every step of PDM, TDM and SDM. The target shape and the initial mesh are the same as in Example 2 (shown in Fig. 5). The corresponding variants of PDM, TDM and SDM with step size control are called PDMSC, TDMSC and SDMSC, respectively. Fig. 8(e) shows the error curves of PDM, TDM and SDM and their variants. We see that the stability of TDM is improved greatly by step size control, while PDMSC and SDMSC have similar convergence behavior to PDM and SDM. PDMSC has an E_{rms} larger than 0.002 after 100 iterations (4.138s) while SDMSC and TDMSC just take 5 iterations (0.240s) and 12 iterations (0.812s) to obtain E_{rms} smaller than 0.002, respectively.

Discussion: From the three preceding examples we conclude that LM regularization is helpful to the stabilization of TDM and SDM, and it ensures monotonic decrease of fitting errors. Step size control by the Armijo rule is also very effective for TDM, but much less effective for PDM and SDM, which are often already quite stable even without step size control, though step size control does ensure monotonic decrease of SDM.



(a) (b) (c) (d) (e)

Fig. 8. Error curves (a) for Example 6; this is the zoom-in view of the TDM error curve in Figure 4. (b) for Example 7. (c) for Example 8. (d) for Example 9. (e) for Example 10.



(a) (b) (c) (d) (e) (f)

Fig. 9. Ball Joint: (a) Point cloud. ($137,062$ points; dimension: $0.87 \times 0.50 \times 1$) (b) Initial mesh. ($416$ control points) (c) Initial surface. (d) Final mesh. (551 control points) (e) Final surface. (f) Shaded subdivision surface. Max. Err.: 0.0064; RMS. Err.: 0.0009.

TABLE I

TIME STATISTICS (IN SECONDS) FOR PRE-COMPUTATION.

	# of data points	Curvatures	Distance fields
Ball joint	137,062	95.83s	52.51s
Igea	134,345	36.07s	215.69s
Rocker arm	40,177	14.91s	70.48s
Bunny	35,201	212.74s	148.26s
Buddha	543,652	4837.67s	200.66s

TABLE II

TIME BREAKDOWN (IN SECONDS).

	Eqn. setup	Eqn. solving	Err. evaluation	Ttl. time
Ball joint	38	3	31	73
Igea	83	5	63	152
Rocker arm	41	6	30	78
Bunny	39	5	32	77
Buddha	1,134	32	641	1,808

TABLE III

COMPARISON WITH THE APPROACH IN [24] FOR THE IGEA MODEL. THE ERRORS E_m AND E_{rms} ARE EXPRESSED IN PERCENTAGE OF THE DIAGONAL OF THE MODEL.

	Final # of control points	E_m (%)	E_{rms} (%)	Time taken (min:sec)
Result in [24]	1,553	0.247	0.05755	8:29
Our result	2,385	0.229	0.03472	2:32

D. Surfaces from Complex Target Shapes

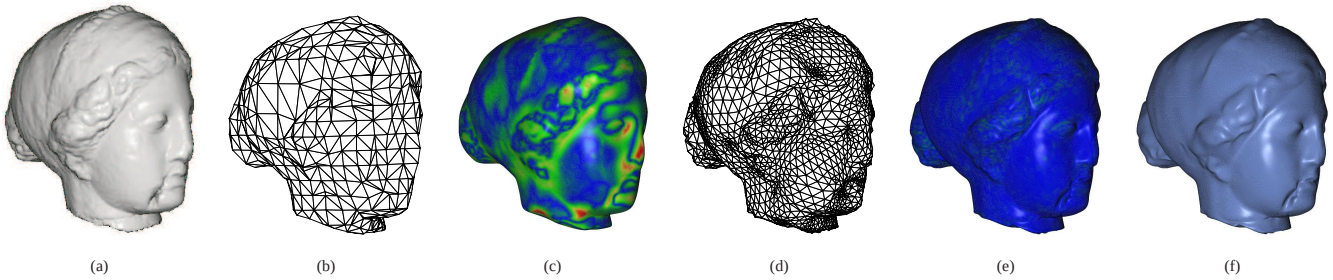
In this section we will show that our fitting method works effectively as well for complex target shapes. Figures 9, 10, 11, 12 and 13 show the data sets for a ball joint, a head (Igea), a rocker arm, a bunny and a buddha (*data sources: http://www.cyberware.com (Igea, the ball joint and the rocker arm) and http://www-graphics.stanford.edu/data/3Dscanrep/ (the bunny and the buddha)*). The figures show the initial meshes, the optimized control meshes by SDM, the initial and optimized subdivision surfaces with color error coding and the shaded optimized subdivision surfaces. The color code of a data point is interpolated in a piecewise linear manner from blue, green, yellow and red corresponding to local error values (c.f. Section II-B) 0.0, 0.0066666, 0.0133333, and 0.02, respectively.

Table I gives the timing data for the preprocessing steps. Neighboring points within a distance of 0.03 from a data point V_k are used for computing the curvatures at V_k. Table IV shows the error statistics. The numbers in *# of contr. pts* field refer to the number of control points in the initial control meshes and the final optimized control meshes. The numbers in *smooth. coeff.* field refer to the initial and the final values for the smoothing term coefficient. Table II shows the breakdown of the time used in different tasks in the optimization. The total time does not include the time on pre-computation. All the five examples were computed with SDM.

TABLE IV

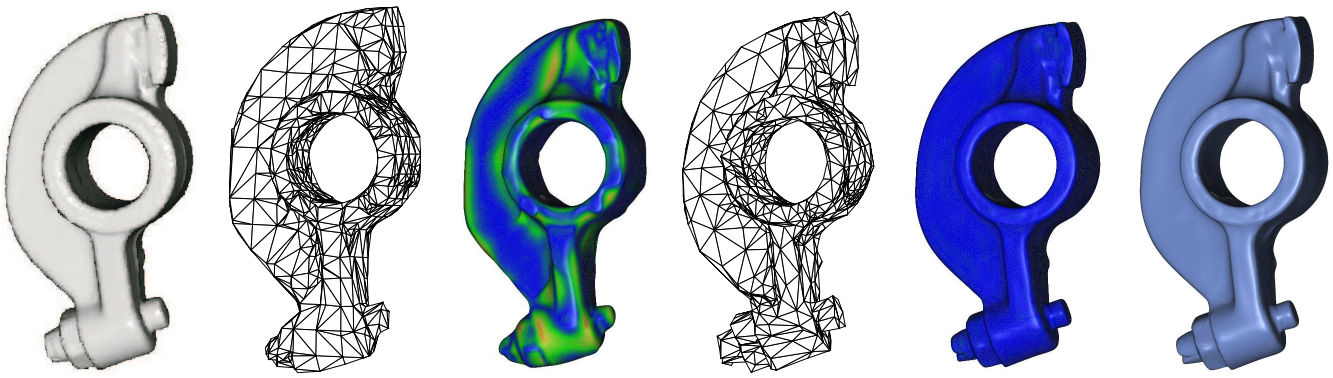
OPTIMIZATION SET UP.

	# of iter.	# of contr. pts.	smooth. coeff.	E_m	E_{rms}
Ball joint	29	416, 551	10^{-2}, 10^{-4}	0.0064	0.0009
Igea	14	526, 2,385	10^{-2}, 10^{-4}	0.0036	0.0005
Rocker arm	15	870, 950	10^{-2}, 10^{-5}	0.0029	0.0003
Bunny	14	919, 996	10^{-2}, 10^{-5}	0.0082	0.0009
Buddha	9	4,662, 18,715	10^{-3}, 10^{-5}	0.0043	0.0003



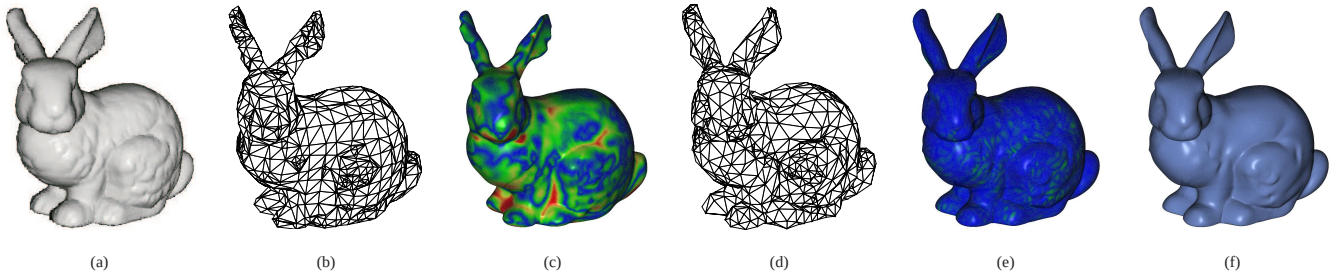
(a) (b) (c) (d) (e) (f)

Fig. 10. Igea: (a) Point cloud. $(134,345$ points; dimension: $0.70 \times 1 \times 1)$ (b) Initial mesh. $(526$ control points) (c) Initial surface. (d) Final mesh. $(2,464$ control points) (e) Final surface. (f) Shaded subdivision surface. Max. Err: 0.0036; RMS. Err.: 0.0005.



(a) (b) (c) (d) (e) (f)

Fig. 11. RockerArm: (a) Point cloud. $(40,177$ points; dimension: $0.51 \times 1 \times 0.30)$ (b) Initial mesh. $(870$ control points) (c) Initial surface. (d) Final mesh. $(950$ control points) (e) Final surface. (f) Optimized subdivision surface. Max. Err.: 0.0029; RMS. Err.: 0.0003.



(a) (b) (c) (d) (e) (f)

Fig. 12. Bunny: (a) Point cloud. $(35,201$ points; dimension: $1 \times 0.78 \times 0.99)$ (b) Initial mesh. $(919$ control points) (c) Initial surface. (d) Final mesh. $(996$ control points) (e) Final surface. (f) Optimized subdivision surface. Max. Err.: 0.0082; RMS. Err.: 0.0009.

(a) (b) (c) (d) (e) (f)

Fig. 13. Buddha: (a) Point cloud. $(543,652$ points; dimension: $0.41 \times 0.41 \times 1)$ (b) Initial mesh. $(4,662$ control points) (c) Initial surface. (d) Final mesh. $(18715$ control points) (e) Final surface. (f) Optimized subdivision surface. Max. Err.: 0.0043; RMS. Err.: 0.0003.

From Table II we observe that the time for generating entries of the matrix of the linear equations is substantial when compared with other parts. Note that the number of data points affects mainly the time used for the preprocessing steps, but does not affect much the time used in the optimization step, which is mainly determined by the number of control points.

Finally, we would like to compare the fitting result of the Igea model in Fig. 10 with that produced in [24]. Table III shows that SDM obtains comparably small E_m and E_{rms} as by the method in [24], using significantly shorter period of time. (The PC on which we ran this experiment has the same specifications as the one used in [24].) However, we note that the projection direction for fitting error evaluation is different from that in our approach. In [24], target data points are projected onto the fitting subdivision surface. In our approach, sample points on the subdivision surface are projected onto the target shape, following the framework in [6].

V. Discussion and Conclusion

We have presented a comprehensive study on a class of three methods for fitting subdivision surfaces to 3D data points, both theoretically and experimentally, from the optimization point of view. There are a variety of other optimization techniques that can be applied to shape fitting and further efforts should be made on understanding these methods as well.

There are currently two different variants of SDM, along with TDM and PDM. The study presented in this paper is based on the original framework proposed by Pottmann et al. [6], and the other variant has been recently proposed by Wang et al. [28] in the setting of B-spline curve fitting, where the connections of PDM, TDM and SDM to the standard optimization techniques are also studied. While these two different frameworks have apparently been motivated by the same geometric intuition, they need to be distinguished carefully, not merely because of the difference in the dimensions of their working spaces. In the framework of [28], the data points are projected onto the fitting curve for error measurement, while in the present paper, sampled points on the fitting surface are projected onto a fixed target shape defined by data points for error measurement. This difference means that the analysis and results of [28] do not apply to the setting in this paper. For example, here we have shown that the SDM method follows directly from the Newton method, whereas the Hessian has to be greatly simplified in order to derive the SDM method in [28]. In addition, the present paper is featured by extensive experimental study of effects of many practical aspects of optimization, including control point insertion, smoothness terms, LM regularization and the Armijo rule for step size control.

In the three optimization methods considered here, since sample points on the fitting surface are projected onto the target shape for setting up error functions, they essentially assume that the target shape is not very noisy or sparse so that the estimated normal and curvatures are reasonably accurate. The advantage of this treatment is that foot points, normal and curvature information can be computed efficiently with the aid of preprocessing of the fixed target shape. However, the drawback is the limit on the application of these methods to fitting a surface to noisy and sparse data points. One way of addressing this limitation is to use instead the methods (PDM, TDM and SDM) presented in [24] and [28]. Note that, in that case, computation of foot points, normal and curvature information has to be performed on an iteratively-updating fitting subdivision surface, and will therefore be relatively time-consuming.

References

[1] K.-S.D. Cheng, W. Wang, H. Qin, K.-Y. K. Wong, H. Yang, and Y. Liu, "Fitting Subdivision Surfaces to Unorganized Point Data Using SDM," in *Pacific Graphics 2004*, 2004, pp. 16–24.

[2] S. Schaefer and J. Warren, "Dual Marching Cubes: Primal Contouring of Dual Grids," in *Proc. Pacific Graphics'04*, 2004, pp. 70–76.

[3] C. Loop, "Smooth Subdivision Surfaces based on Triangles," M.S. thesis, Department of Mathematics, University of Utah, 1987.

[4] Josef Hoschek, "Intrinsic parameterization for approximation," *Computer Aided Geometric Design*, vol. 5, pp. 27–31, 1988.

[5] G. Medioni Y. Chen, "Object modeling by registration of multiple range images," *Image and Vision Computing*, pp. 145–155, 1992.

[6] H. Pottmann and S. Leopoldseder, "A Concept for Parametric Surface Fitting which avoids the Parametrization Problem," *Computer Aided Geometric Design*, vol. 20, pp. 343–362, 2003.

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

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

[9] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface Reconstruction from Unorganized Points," in *Proc. SIGGRAPH '92*, 1992, pp. 71–78.

[10] M. Halstead, M. Kass, and T. DeRose, "Efficient, Fair Interpolation using Catmull-Clark Surfaces," in *Proc. SIGGRAPH '93*, 1993, pp. 35–44.

[11] W. Ma and J. P. Kruth, "Parameterization of Randomly Measured Points for Least Squares Fitting of B-spline Curves and Surfaces," *Computer-Aided Design*, vol. 27, no. 9, pp. 663–675, 1995.

[12] W. Ma and N. Zhao, "Smooth Multiple B-spline Surface Fitting with Catmull-Clark Subdivision Surfaces for Extraordinary Corner Patches," *The Visual Computer*, vol. 18, pp. 415–436, 2002.

[13] M. Marinov and L. Kobbelt, "Optimization Techniques for Approximation with Subdivision Surfaces," in *ACM Symposium on Solid Modeling and Applications*, 2004, pp. 1–10.

[14] H. Suzuki, S. Takeuchi, and T. Kanai, "Subdivision Surface Fitting to a Range of Points," in *The Seventh Pacific Conference on Computer Graphics and Applications*, 1999, pp. 158–167.

[15] A. Blake and M. Isard, *Active Contours*, Springer, 1998.

[16] H. K. Zhao, S. Osher, B. Merriman, and M. Kang, "Implicit and Non-parametric Shape Reconstruction from Unorganized Data using a Variational Level Set Method," *Computer Vision and Image Understanding*, vol. 80, pp. 295–319, 2000.

[17] Y. Duan and H. Qin, "Intelligent Balloon: A Subdivision-based Deformable Model for Surface Reconstruction of Arbitrary Unknown Topology," in *Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications*, 2001, pp. 47–58.

[18] N. Litke, A. Levin, and P. Schröder, "Fitting Subdivision Surfaces," in *Proc. IEEE Visualization 2001*, 2001, pp. 319–324.

[19] B. Jüttler and A. Felis, "Least-Squares Fitting of Algebraic Spline Surfaces," *Advances in Computational Mathematics*, vol. 17, pp. 135–152, 2002.

[20] T. Kanai, "MeshToSS: Converting Subdivision Surfaces from Dense Meshes," in *Proceedings of the Vision Modeling and Visualization Conference 2001*, 2001, pp. 325–332.

[21] W.-K. Jeong, I. Ivrissimtzis, and H.-P. Seidel, "Neural Meshes: Statistical Learning Based on Normals," in *Pacific Graphics 2003*, 2003, pp. 404–408.

[22] D. Zorin, P. Schröder, A. DeRose, J. Stam, L. Kobbelt, and J. Warren, "Subdivision for Modeling and Simulation," in *SIGGRAPH '99 Course Notes*, 1999.

[23] A. Ruhe and P.-A. Wedin, "Algorithms for Separable Nonlinear Least Squares Problems," *SIAM Review*, vol. 22, pp. 318–337, 1980.

[24] M. Marinov and L. Kobbelt, "Optimization Methods for Scattered Data Approximation with Subdivision Surfaces," *Graphical Models*, vol. 67, no. 5, pp. 452–473, 2005.

[25] L. Ambrosio and C. Montegazza, "Curvature and distance function from a manifold," *Journal of Geometric Analysis*, vol. 8, pp. 723–748, 1998.

[26] H. Pottmann and M. Hofer, "Geometry of the Squared Distance Function to Curves and Surfaces," *Visualization and Mathematics III, Springer*, pp. 223–244, 2003.

[27] H. Pottmann, S. Leopoldseder, and M. Hofer, "Approximation with Active B-spline Curves and Surfaces," in *Proc. Pacific Graphics 02*, 2002, pp. 8–25.

[28] W. Wang, H. Pottmann, and Y. Liu, "Fitting B-Spline Curves to Point Clouds by Squared Distance Minimization," *ACM Transactions on Graphics*, vol. 25, no. 2, pp. 214–238, 2006.

[29] J. Stam, "Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values," in *Proc. SIGGRAPH'98*, 1998, pp. 395–404.

[30] J. Stam, "Evaluation of Loop Subdivision Surfaces," in *SIGGRAPH'98 CDROM Proceedings*, 1998.

[31] S. Leopoldseder, H. Pottmann, and H. K. Zhao, "The d^2 Tree: A Hierarchical Representation of the Squared Distance Field," Tech. Rep. 101, Institute of Geometry, Vienna University of Technology, 2003.

[32] W. Lorensen and H. Cline, "Marching Cubes: A High Resolution 3-D Surface Construction Algorithm," *Computer Graphics*, vol. 21, pp. 163–169, 1987.

[33] C. Lürig, L. Kobbelt, and T. Ertl, "Hierarchical Solutions for the Deformable Surface Problem in Visualization," *Graphical Models*, vol. 62, pp. 2–18, 2000.

[34] V. Weiss, L. Andor, G. Renner, and T. Varady, "Advanced Surface Fitting Techniques," *Computer Aided Geometric Design*, vol. 19, no. 1, pp. 19–42, 2002.

[35] R.E. Bank, A.H. Sherman, and A. Weiser, "Refinement Algorithm and Data Structures for Regular Local Mesh Refinement," in *Scientific Computing*, 1983, pp. 3–17.

[36] K. Madsen, H.B. Nielsen, and O. Tingleff, *Methods for Non-Linear Least Squares Problems*, Informatics and Mathematical Modelling, Technical University of Denmark, 2004.

APPENDIX A: TDM AND GAUSS-NEWTON METHOD

We shall show that TDM uses a Gauss-Newton iteration. We first compute the gradient of F by taking into account the dependence of the foot point parameters (u, v) on the control points $\mathcal{P}$ through the constraints (13). Since $F_k = \frac{1}{2}E^T E$, due to the constraints (13), we obtain the gradient vector

$$\nabla_{\mathcal{P}} F_k = (S_{\mathcal{P}}^T - \nabla_{\mathcal{P}} u V_u^T - \nabla_{\mathcal{P}} v V_v^T)E = S_{\mathcal{P}}^T E. \quad (16)$$

Here $\nabla_{\mathcal{P}} F_k$ is $3n$ vector, where n is the number of control points, and $S_{\mathcal{P}}^T$ is a $3n$ by 3 matrix. Let $F_k(x) = \frac{1}{2}f_k^2 = \frac{1}{2}E^T E$, where $f_k = \|E\|$. Since $\nabla_{\mathcal{P}} F_k = f_k \nabla_{\mathcal{P}} f_k$, we have, by (16),

$$\nabla_{\mathcal{P}} f_k = \frac{\nabla_{\mathcal{P}} F_k}{f_k} = S_{\mathcal{P}}^T \frac{E}{f_k} = S_{\mathcal{P}}^T \frac{E}{\|E\|} = S_{\mathcal{P}}^T N, \quad (17)$$

where N is the unit normal vector of the target surface Γ_T at the foot point V.

By (17), we have $\nabla_{\mathcal{P}} f_k (\nabla_{\mathcal{P}} f_k)^T = S_{\mathcal{P}}^T N N^T S_{\mathcal{P}}$. In the second order Taylor expansion of F_k at $\mathcal{P}_c$, replacing the Hessian by $\nabla_{\mathcal{P}} f_k (\nabla_{\mathcal{P}} f_k)^T$ yields,

$$\begin{aligned}
& F_k(\mathcal{P}_c) + \nabla_{\mathcal{P}} F_k(\mathcal{P}_c)^T \Delta\mathcal{P} + \tfrac{1}{2}\Delta\mathcal{P}^T \nabla_{\mathcal{P}}^2 F_k(\mathcal{P}_c)\Delta\mathcal{P} \\
\approx\ & F_k(P_c) + \nabla_{\mathcal{P}} F_k(\mathcal{P}_c)^T \Delta\mathcal{P} + \tfrac{1}{2}\Delta\mathcal{P}^T S_{\mathcal{P}}^T N N^T S_{\mathcal{P}} \Delta\mathcal{P} \\
=\ & \tfrac{1}{2}(S_c - V_c)^T (S_c - V_c) + (S_c - V_c)^T S_{\mathcal{P}} \Delta\mathcal{P} \\
+\ & \tfrac{1}{2}\Delta\mathcal{P}^T S_{\mathcal{P}}^T N N^T S_{\mathcal{P}} \Delta\mathcal{P} \\
=\ & \tfrac{1}{2}(S_c - V_c)^T N N^T (S_c - V_c) \\
+\ & (S_c - V_c)^T N N^T S_{\mathcal{P}} \Delta\mathcal{P} + \tfrac{1}{2}\Delta\mathcal{P}^T S_{\mathcal{P}}^T N N^T S_{\mathcal{P}} \Delta\mathcal{P} \\
=\ & \tfrac{1}{2}(S_c + S_{\mathcal{P}}\Delta\mathcal{P} - V_c)^T N N^T (S_c + S_{\mathcal{P}}\Delta\mathcal{P} - V_c) \\
=\ & \tfrac{1}{2}(S - V_c)^T N N^T (S - V_c) = \tfrac{1}{2}[(S - V_c)^T N]^2 \\
=\ & \tfrac{1}{2} \times \text{TD error term}, \qquad (18)
\end{aligned}$$

where the last equality follows from (4). Since $(S_c - V_c)$ and N above are parallel, we have made use of the fact that

$$(S_c - V_c)^T (S_c - V_c) = (S_c - V_c)^T N N^T (S_c - V_c)$$

and

$$(S_c - V_c)^T S_{\mathcal{P}} \Delta\mathcal{P} = f_k N^T S_{\mathcal{P}} \Delta\mathcal{P}$$
$$= f_k N^T N N^T S_{\mathcal{P}} \Delta\mathcal{P} = (S_c - V_c)^T N N^T S_{\mathcal{P}} \Delta\mathcal{P}.$$

Hence, by (18), TDM uses a Gauss-Newton step.

Appendix B: SDM and Newton Method

Now we are going to show that the SD error term is given by the Newton iteration, after appropriate modification to make the Hessian positive semi-definite. Differentiating $\nabla_{\mathcal{P}} F_k$ in (16) yields the Hessian as

$$\begin{aligned}\nabla^2_{\mathcal{P}} F_k &= \left(S^T_{\mathcal{P}} - \nabla_{\mathcal{P}} u V_u^T - \nabla_{\mathcal{P}} v V_v^T\right) S_{\mathcal{P}} + E^T S_{\mathcal{P}\mathcal{P}} \\ &= \left(S^T_{\mathcal{P}} - \nabla_{\mathcal{P}} u V_u^T - \nabla_{\mathcal{P}} v V_v^T\right) S_{\mathcal{P}}. \end{aligned} \quad (19)$$

Here $E^T S_{\mathcal{P}\mathcal{P}} = 0$, since the sample point S is a linear combination of control points $\mathcal{P}$.

Without loss of generality, suppose that $V(u, v)$ is a local regular parameterization of the target surface Γ_T such that $V_u = T_1$ and $V_v = T_2$, which are the unit principal direction vectors of Γ_T at the current foot point V_c, and then there is $V_u^T V_v = 0$. Differentiating the constraints (13) with respect to $\mathcal{P}$ yields

$$\begin{aligned} 0 &= \left(\nabla_{\mathcal{P}} u\, V_{uu}^T + \nabla_{\mathcal{P}} v\, V_{vu}^T\right) E \\ &+ \left(S^T_{\mathcal{P}} - \nabla_{\mathcal{P}} u\, V_u^T - \nabla_{\mathcal{P}} v\, V_v^T\right) V_u \\ &= \left(V_{uu}^T E - V_u^T V_u\right) \nabla_{\mathcal{P}} u + S^T_{\mathcal{P}}\, V_u + V_{vu}^T E \nabla_{\mathcal{P}} v \end{aligned} \quad (20)$$

and

$$\begin{aligned} 0 &= \left(\nabla_{\mathcal{P}} v\, V_{vv}^T + \nabla_{\mathcal{P}} u\, V_{uv}^T\right) E \\ &+ \left(S^T_{\mathcal{P}} - \nabla_{\mathcal{P}} u\, V_u^T - \nabla_{\mathcal{P}} v\, V_v^T\right) V_v \\ &= \left(V_{vv}^T E - V_v^T V_v\right) \nabla_{\mathcal{P}} v + S^T_{\mathcal{P}}\, V_v + V_{uv}^T E \nabla_{\mathcal{P}} u. \end{aligned} \quad (21)$$

From (20) and (21), we obtain

$$\nabla_{\mathcal{P}} u = -\frac{S^T_{\mathcal{P}}\, V_u + V_{vu}^T E \nabla_{\mathcal{P}} v}{V_{uu}^T E - V_u^T V_u} \quad (22)$$

and

$$\nabla_{\mathcal{P}} v = -\frac{S^T_{\mathcal{P}}\, V_v + V_{uv}^T E \nabla_{\mathcal{P}} u}{V_{vv}^T E - V_v^T V_v}. \quad (23)$$

By assumption and differential geometry of surfaces, we have $V_u = T_1$, $V_v = T_2$, $V_{uu} = \kappa_1 N$, $V_{vv} = \kappa_2 N$, $E = dN$, where κ_1, κ_2 are the signed principal curvatures at V. With the substitutions in (22) and (23), we obtain $V_{uu}^T E - V_u^T V_u = d\kappa_1 - 1$ and $V_{vv}^T E - V_v^T V_v = d\kappa_2 - 1$. Furthermore, since $V_u^T E = 0$, differentiating with respect to v yields

$$V_{uv}^T E + V_u^T E_v = V_{uv}^T E - V_u^T V_v = 0.$$

Therefore, $V_{uv}^T E = V_u^T V_v = 0$. Putting all these together, it follows from (22) and (23) that

$$\nabla_{\mathcal{P}} u = -\frac{S^T_{\mathcal{P}}\, T_1}{d\kappa_1 - 1}, \quad \nabla_{\mathcal{P}} v = -\frac{S^T_{\mathcal{P}}\, T_2}{d\kappa_2 - 1}. \quad (24)$$

Recall the notation $\rho_1 = \frac{1}{\kappa_1}$ and $\rho_2 = \frac{1}{\kappa_2}$ from Section II-D. Substituting (24) in (19) yields

$$\begin{aligned}\nabla^2_{\mathcal{P}} F_k &= S^T_{\mathcal{P}} S_{\mathcal{P}} + \frac{S^T_{\mathcal{P}} T_1 T_1^T S_{\mathcal{P}}}{d\kappa_1 - 1} + \frac{S^T_{\mathcal{P}} T_2 T_2^T S_{\mathcal{P}}}{d\kappa_2 - 1} \\ &= S^T_{\mathcal{P}} (I - T_1 T_1^T - T_2 T_2^T) S_{\mathcal{P}} \\ &\quad + d\kappa_1 \frac{S^T_{\mathcal{P}} T_1 T_1^T S_{\mathcal{P}}}{d\kappa_1 - 1} + d\kappa_1 \frac{S^T_{\mathcal{P}} T_2 T_2^T S_{\mathcal{P}}}{d\kappa_2 - 1} \\ &= S^T_{\mathcal{P}} N N^T S_{\mathcal{P}} + \alpha_1 S^T_{\mathcal{P}} T_1 T_1^T S_{\mathcal{P}} + \alpha_2 S^T_{\mathcal{P}} T_2 T_2^T S_{\mathcal{P}}, \end{aligned}$$

where $\alpha_1 = d/(d - \rho_1)$ and $\alpha_2 = d/(d - \rho_2)$ as in Section II-D. Here note that $T_1 T_1^T + T_2 T_2^T + N N^T = I$ and d, ρ_1, ρ_2 are signed values (see Section II-D.3).

Thus we have obtained the Hessian $\nabla^2 F_k$. Substituting $\nabla^2 F_k$ in the following second order Taylor approximation of the objective function, noting that $(S_c - V_c)^T T_1 = (S_c - V_c)^T T_2 = 0$, we obtain

$$\begin{aligned} & F_k(\mathcal{P}_c) + \nabla_{\mathcal{P}} F_k(\mathcal{P}_c)^T \Delta\mathcal{P} + \tfrac{1}{2} \Delta\mathcal{P}^T \nabla^2_{\mathcal{P}} F_k(\mathcal{P}_c) \Delta\mathcal{P} \\ = \;& \tfrac{1}{2}(S_c - V_c)^T (S_c - V_c) + (S_c - V_c)^T S_{\mathcal{P}} \Delta\mathcal{P} \\ & + \tfrac{1}{2} \Delta\mathcal{P}^T \left(S^T_{\mathcal{P}} N N^T S_{\mathcal{P}} + \alpha_1 S^T_{\mathcal{P}} T_1 T_1^T S_{\mathcal{P}} \right. \\ & \left. + \alpha_2 S^T_{\mathcal{P}} T_2 T_2^T S_{\mathcal{P}}\right) \Delta\mathcal{P} \\ = \;& \tfrac{1}{2}(S_c - V_c)^T N N^T (S_c - V_c) \\ & + (S_c - V_c)^T N N^T S_{\mathcal{P}} \Delta\mathcal{P} + \tfrac{1}{2} \Delta\mathcal{P}^T S^T_{\mathcal{P}} N N^T S_{\mathcal{P}} \Delta\mathcal{P} \\ & + \tfrac{1}{2} \alpha_1 \left(S_c + S_{\mathcal{P}} \Delta\mathcal{P} - V_c\right)^T T_1 T_1^T \left(S_c + S_{\mathcal{P}} \Delta\mathcal{P} - V_c\right) \\ & + \tfrac{1}{2} \alpha_2 \left(S_c + S_{\mathcal{P}} \Delta\mathcal{P} - V_c\right)^T T_2 T_2^T \left(S_c + S_{\mathcal{P}} \Delta\mathcal{P} - V_c\right) \\ = \;& \tfrac{1}{2} \left(S_c + S_{\mathcal{P}} \Delta\mathcal{P} - V_c\right)^T N N^T \left(S_c + S_{\mathcal{P}} \Delta\mathcal{P} - V_c\right) \\ & + \tfrac{1}{2} \alpha_1 \left(S - V_c\right)^T T_1 T_1^T \left(S - V_c\right) \\ & + \tfrac{1}{2} \alpha_2 \left(S - V_c\right)^T T_2 T_2^T \left(S - V_c\right) \\ = \;& \tfrac{1}{2}[(S - V_c)^T N]^2 + \tfrac{1}{2} \alpha_1 [(S - V_c)^T T_1]^2 \\ & + \tfrac{1}{2} \alpha_2 [(S - V_c)^T T_2]^2 \\ = \;& \tfrac{1}{2} \hat{f}^2_{SD,k}, \end{aligned} \quad (25)$$

which, up to a constant multiple, is the second order approximation of the squared distance given in (5). Thus the SD error term defined in (6) is derived from the Newton iteration after removing negative eigenvalues from the full Hessian by setting the negative coefficients in (5), if any, to zero.

When we compute the gradient and Hessian of the global function with respect to $\mathcal{P}$, we treat $\mathcal{P}$ as basic variables and $\mathcal{U}$ as dependent variables. Furthermore, the dependence between $\mathcal{U}$ and $\mathcal{P}$ is expressed by the way we compute the gradient in (16) and the Hessian using the local linear relations (22) and (23). In fact, this local linear dependence

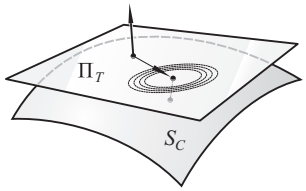


Fig. 14. Constraint surface and its tangent plane.

between $\mathcal{U}$ and $\mathcal{P}$ represents the tangent plane Π_T to the constraint hypersurface S_C defined by the constraints in (13). Therefore, the iterates used by TDM and SDM essentially move on the plane Π_T, according to their respective approximations to the Hessian (see Fig. 14).

ACKNOWLEDGEMENT

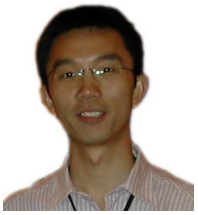
The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. HKU 7155/03E).

Kin-Shing D. Cheng received the BEng degree in computer engineering, with the first class honors, from The University of Hong Kong in 1997. He received the MPhil and PhD degrees, both in computer graphics, from the University of Hong Kong in 2001 and 2006, respectively. His research interests include computer graphics and geometric computing.

Dr. Wenping Wang is Associate Professor of Computer Science at University of Hong Kong. He received Ph.D. in Computer Science from University of Alberta in 1992. He has published widely in the fields of computer graphics, geometric computing and visualization. He is associate editor of Computer Aided Geometric Design, and has been program co-chair of several international conferences, including Geometric Modeling and Processing 2000, Pacific Graphics 2003, and ACM Symposium on Physical and Solid Modeling 2006. `http://www.cs.hku.hk/~wenping/`

At present, Dr. Hong Qin is a full professor of Computer Science in Department of Computer Science at State University of New York at Stony Brook (Stony Brook University). He received his B.S. degree and his M.S. degree in Computer Science from Peking University in Beijing, China. He received his Ph.D. (1995) degree in Computer Science from the University of Toronto. During his years at the University of Toronto (UofT), he received UofT Open Doctoral Fellowship. He was also a recipient of NSF CAREER Award from the National Science Foundation (NSF), Honda Initiation Award, and Alfred P. Sloan Research Fellow by the Sloan Foundation. In 2005, Professor Qin served as the general Co-Chair for Computer Graphics International 2005 (CGI'2005). Currently, he is an associate editor for IEEE Transactions on Visualization and Computer Graphics (IEEE TVCG), and he is also on the editorial board of The Visual Computer (International Journal of Computer Graphics). In 2007, he is the Conference Co-Chair for ACM Solid and Physical Modeling Symposium. His research interests include geometric and solid modeling, graphics, physics-based modeling and simulation, computer aided geometric design, human-computer interaction, visualization, and scientific computing. Detailed information about Dr. Hong Qin can be found from his website: `http://www.cs.sunysb.edu/~qin`

Kwan-Yee K. Wong received the BEng degree in computer engineering, with first class honors, from The Chinese University of Hong Kong in 1998. From 1998 to 2001, he studied in United Kingdom at the University of Cambridge, and he received the MPhil and PhD degrees, both in computer vision, from the University of Cambridge in 2000 and 2001, respectively. Since 2001, Dr. Wong has been with the Department of Computer Science at The University of Hong Kong, where he is now an assistant professor. His research interests are in computer vision and image processing, including camera calibration, motion tracking, model reconstruction and representation, and motion estimation from image sequences.



Huaiping Yang received the BE (1998) degree in hydraulic engineering and the PhD (2004) degree in computer science from Tsinghua University, China. In 2004, he did a one-year postdoc in the computer graphics group at the University of Hong Kong. In 2005, he starts a postdoc at JKU Linz, Austria, in the field of applied geometry, funded by a Marie Curie incoming international fellowship. His research interests include geometric modeling, computer graphics, and scientific visualization.

Yang Liu is a PhD student in the Computer Science Department at the University of Hong Kong. He received his BS (2000) and MS (2003) degrees in mathematics from the University of Science and Technology of China. His research interests include Computer-Aided Geometric Design, Computer Graphics and Computational Algebraic Geometry. `http://www.cs.hku.hk/~yliu/`