Proc. of The 7th China-Japan Seminar on Numerical Mathematics 0 (2006) 161–176

A Fast and Adaptive Surface Reconstruction Algorithm Based on Implicit Tensor-Product B-Spline Surfaces *

Tong WeiHua [a] Chen FaLai [a] Feng YuYu [a]

[a] *Department of Mathematics*
University of Science and Technology of China
Hefei, Anhui 230026, People's Republic of China
E-mail: tongwh@ustc.edu.cn, chenfl@ustc.edu.cn, fengyy@ustc.edu.cn

Based on the implicit tensor-product B-spline (ITPBS) representation of surfaces, we propose a fast and adaptive algorithm to solve the surface reconstruction problem. Our algorithm is driven by a surface fitting model, which amounts to solving a quadratic optimization problem. We explore the matrix form of it, and with some elaborate analysis, we put forward a fast and low memory consumption algorithm which only requires $\mathcal{O}(U)$ operations and $\mathcal{O}(M)$ storage space, where U is the number points in the point clouds and M is the number of unknown coefficients. In addition, we present heuristic ideas on how to adaptively choose the knot vectors of the ITPBS surface. For non-uniform(adaptive) sampling point sets, two algorithms are provided for generating the adaptive knot sequences. We conclude the paper with some illustrating examples and conclusion remarks.

Keywords: surface reconstruction, implicit B-spline surface, point cloud, sparse matrix

AMS Subject classification: Primary 65B15, 65B99; Secondary 65C05

1. Introduction

In recent years, the availability of fast and accurate geometric data acquisition devices, such as laser range scanners, CT scans, contact probe digitizers, etc., has made it relatively simple and cheap to acquire large collections of points on surfaces of objects. The problem of converting these sampling point sets into geometric models is referred to as *surface reconstruction*. Applications, which would benefit from this efficient and

* The authors are support by the Outstanding Youth Grant of NSF of China(No.60225002), a National Key Basic Research Project of China (No.2004CB318000), NSF of China(No.60533060 and No.60473132), the TRAPOYT in Higher Education Institute of MOE of China, and SRF for ROCS, SEM.

reliable method of building geometric models from real objects, come from Computer Aided Geometric Design, Computer Graphic, Medical image Processing, Computer Vision, etc.

There has been a large body of literatures on surface reconstruction problem in the last ten years. However, up to now, surface reconstruction is still a not well-solved problem. The underlying reason is that, mathematically, it is an ill-posed problem, i.e., there is no unique solution. Furthermore, the problem is very challenging due to the following difficulties: (1) the geometry and topology of the real objects are not known a priori and may be very complicated; (2) the huge size of sampling point set, for instance, the Stanford Lucy consists of 58,241,932 points; and (3) the presence of holes, noises, low or non-uniform sampling density, sharp features, etc.

1.1. Previous Work

Based on different representations of reconstruction surfaces, surface reconstruction algorithms can be classified into three categories: mesh representation, parametric representation and implicit representation.

The fundamental tools of mesh representation are Voronoi diagrams and Delaunay triangulations. Typically work includes: the 3D α-shapes algorithm proposed by Edelsbrunner and Mücke [11]; the Crust algorithm introduced by Amenta and Bern et al. [2], etc. Although these approaches produce mesh representation directly, the reconstructed surface is only piecewise linear and it is difficult to handle non-uniform and noisy data.

Another type of techniques is based on the parametric representation of reconstruction surfaces. It includes the Shepard's methods [29], the Radial Basis Function methods [26], the FEM methods [16], the Hierarchical B-spline Patchs method [12], etc. However, most of these methods either assume that the surface has a simple topological type, or require user intervention in setting up the patch network. Furthermore, the initial parametrization for the data points is critical to the quality of the reconstructed parametric representation. Extensive work, for instance, Ma and Kruth [22], Krishnamurthy and Levoy [21], Eck and Hoppe [10], addresses this problem. However, no solution is suitable for all the cases.

The third type of techniques is based on implicit surfaces. Implicit surfaces have many advantages, for instance, the ability of adapting to complicated topological objects, no needs for initial parametrization, etc. Early works include: Muraki [23], Hoppe et al. [15], Bajaj et.al [3], Curless and Levoy [9], Carr et al. [8]. et al. Recently, Zhao and Osher [31] applied the level set technique for computing the signed distance function and dynamically fitting point clouds. Alexa et al. [1] developed the projection-based approaches, which have the advantage that they are local

and they directly yield a point on the surface. Ohtake et al. [25] introduced the "Multi-level Partition of Unity" implicit surface, and developed an adaptive error-controlled algorithm for approximating the signed distance function.

1.2. Outline of Our Paper

The paper is organized as follows. In section §2, we present some preliminary knowledge about ITPBS surfaces. In section §3, we review the surface fitting model based on IIPBS surfaces. In section §4, we present the matrix form of the surface fitting model, and develop a fast and low memory consumption algorithm to compute the matrices. In section §5, two algorithms for generating the adaptive knot sequences are provided. In section §6, some examples together with some discussions are illustrated. We end this paper with a summary and future work in section §7.

2. Implicit Tensor-Product B-Splines Surface

An *implicit surface* is defined by a function $f : \Omega \subset \mathbb{R}^3 \to \mathbb{R}$ that assigns a scalar value to each point in the space Ω. The surface is defined by $S = Z(f) = \{\mathbf{p} \in \Omega : f(\mathbf{p}) = 0\}$. Here, we choose *tensor-product B-spline functions* as the geometric representation of reconstruction surfaces, and we refer it to an *ITPBS surface*. The implicit function is defined as follows

$$f(\mathbf{p}) = \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{l} c_{ijk}N_i(x)N_j(y)N_k(z), \mathbf{p} = (x, y, z) \in \Omega \tag{2.1}$$

where the basis functions $N_i(x)$, $N_j(y)$, $N_k(z)$ are the B-spline bases of order d with respect to the knot sequences $(\xi)_{i=1}^{m+d}$, $(\eta)_{j=1}^{n+d}$, $(\zeta)_{k=1}^{l+d}$, see [30] for details.

For our reconstruction purpose, we choose the domain Ω to be a box which encloses the point clouds. The choice of the knot sequences relate nicely to the final result of reconstruction algorithm. We will revisit this issue in section §5.

3. Surface Fitting Model

In this section, we revisit the surface fitting model developed by B.Jüttler [19].

3.1. Surface reconstruction problem

The surface reconstruction problem may be stated as follows

Given an unorganized collection of points $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_U\}$ *and associated unit normal vectors* $\mathcal{N} = \{\mathbf{n}_1, \mathbf{n}_2, \ldots, \mathbf{n}_U\}$. *The goal is to determine a surface* S' *that approximates the unknown surface* S *as well as possible. Namely, we are required to find a function* $f : \Omega \subset \mathbb{R}^3 \to \mathbb{R}$, *such that* $f(\mathbf{p}_u) \approx 0$, *and* $\nabla f(\mathbf{p}_u) \approx \mathbf{n}_u$, *where* $u = 1, \ldots, U$.

Mathematically, it is an *ill-posed problem*, i.e, there is no unique solution. In order to get reasonable result, we need to propose an appropriate mathematical model.

3.2. The Mathematical Model

In the past decade, several models have been developed, e.g, Muraki's blobby model [23], Zhao's weighted minimal surface model [31] and interpolating implicit surface model(RBF) [8]. In this paper, we adopt the surface fitting model developed by Bert Jüttler in [19]. The model is to solve the following optimization problem. The *Objective function* is

$$F(\mathbf{c}) = L(\mathbf{c}) + \omega_1 N(\mathbf{c}) + \omega_2 G(\mathbf{c}) \to Min \tag{3.1}$$

where

- $\mathbf{c}$ — The vector consisting of all the ITPBS function coefficients $(c_{ijk})_{i=1,j=1,k=1}^{m,n,l}$ which are sorted in a suitable order.
- ω_1 — The weight which controls the influence of estimated normals.
- ω_2 — The weight which control the influence of the regularizing term G.
- $L(\mathbf{c})$ — describes the algebraic distance between the original surface S and the reconstruction surface S':

$$L(\mathbf{c}) = \sum_{u=1}^{U} [f(\mathbf{p}_u)]^2, \mathbf{p}_u = (p_{ux}, p_{uy}, p_{uz}) \in \mathbb{R}^3 \tag{3.2}$$

- $N(\mathbf{c})$ — reflects the normal difference between the original surface S and the reconstruction surface S':

$$N(\mathbf{c}) = \sum_{u=1}^{U} \|\nabla f(\mathbf{p}_u) - \mathbf{n}_u\|^2, \mathbf{n}_u = (n_{ux}, n_{uy}, n_{uz}) \tag{3.3}$$

- $G(\mathbf{c})$ — smooth term, or the regularizing term, which try to push the reconstruction surface to a simpler shape:

$$G(\mathbf{c}) = \iiint_{\Omega} \left(f_{xx}^2 + f_{yy}^2 + f_{zz}^2 + 2f_{xy}^2 + 2f_{xz}^2 + 2f_{yz}^2 \right) dxdydz. \tag{3.4}$$

The objective function $F(\mathbf{c})$ is a quadratic function in the unknown coefficients $\mathbf{c}$. Thus the optimization solution can be obtained by solving a linear system of equations.

4. Fast Surface Reconstruction Algorithm

In this section, we explore the matrix form of the surface fitting model developed in the last section, and put forward a fast and low memory consumption algorithm to compute the matrices and solve the optimization problem (3.1).

4.1. The reduction of unknown coefficients

From the definition of implicit surfaces, we observe that the value of the implicit function f at the position which is far from the implicit surface $f = 0$ does not affect the implicit surface itself. Thus, based on the local property of B-spline functions, the coefficients of f do not have any contribution to the implicit surface whenever the support of the corresponding B-spline bases do not have any overlap with the implicit surface. So we may reduce the number of coefficients in the computation.

If let $\mathcal{I} \triangleq \Big\{(i,j,k)|\exists u \in \{1,\ldots,U\}, r \in \{1,\ldots,m\}, s \in \{1,\ldots,n\}, t \in \{1,\ldots,l\} \ : \ N_r(p_{ux})N_s(p_{uy})N_t(p_{uz}) \neq 0\Big\}$. Then, we may rewrite the ITPBS function f as

$$f(\mathbf{p}) = \sum_{(i,j,k)\in\mathcal{I}} c_{ijk}N_i(x)N_j(y)N_k(z), \mathbf{p} = (x,y,z) \in \Omega. \tag{4.1}$$

The above function f defines the same surface as the function f of (2.1), while it has much fewer unknown coefficients. It dramatically reduces the demanding computation and memory cost of the unknown coefficients.

In order to describe the matrix form of the surface fitting model, we introduce the following notation.

Let a map $P : \mathcal{I} \longleftrightarrow \{1,2,\ldots,M\} \subset \mathbb{N}$ *be* $P(i,j,k) = v$, *where* $(i,j,k) \in \mathcal{I}$ *and* $v = 1,2,\ldots,M$. *Here* $M = |\mathcal{I}|$ *is the number elements of* $\mathcal{I}$.

Then we may translate 3-dimension subscripts to 1-dimension subscripts as follows

$$c_{ijk} \longleftrightarrow c_v^{'} = Pc_{ijk}, (i,j,k) \in \mathcal{I}, v = 1,2,\ldots,M. \tag{4.2}$$

Consequently, we can rewrite all unknown coefficients $(c_{ijk})_{(i,j,k)\in\mathcal{I}}$ into a vector $\mathbf{c}$ of dimension M, which are sorted by the map P.

4.2. The matrix form of $L(\mathbf{c})$ term

The $L(\mathbf{c})$ term can be written in the matrix form as

$$L(\mathbf{c}) = \sum_{u=1}^{U} [f(\mathbf{p}_u)]^2 = \begin{bmatrix} c_1' \cdots \cdots c_M' \end{bmatrix} \begin{bmatrix} L_{11} & L_{12} & \cdots & L_{1M} \\ L_{21} & L_{22} & \cdots & L_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ L_{M1} & L_{M2} & \cdots & L_{MM} \end{bmatrix} \begin{bmatrix} c_1' \\ \vdots \\ \vdots \\ c_M' \end{bmatrix} \quad (4.3)$$

where for $s, t = 1, 2, \ldots, M$ and $P(s_i, s_j, s_k) = s$, $P(t_i, t_j, t_k) = t$

$$L_{st} = \sum_{u=1}^{U} N_{s_i}(p_{ux}) N_{s_j}(p_{uy}) N_{s_k}(p_{uz}) \cdot N_{t_i}(p_{ux}) N_{t_j}(p_{uy}) N_{t_k}(p_{uz}).$$

At first glance, the matrix $(L_{st})_{s,t=1}^{M}$ need to be constructed in $\mathcal{O}(UM^2)$ operations and to be stored in a space of $\mathcal{O}(M^2)$. The computational complexity and memory storage are not practical even for a relatively small points set, e.g., $U = 8,000$ and $M = 2,000$. However, we will show that the matrix $(L_{st})_{s,t=1}^{M}$ can be built in $\mathcal{O}(U)$ operations and be stored in a space of $\mathcal{O}(M)$. Our algorithm is based on the following observation:

The matrix $(L_{st})_{s,t=1}^{M}$ is a sparse matrix. Namely, for most of $s, t = 1, \ldots, M$, $L_{st} = 0$.

In fact, for every point $\mathbf{p}_u$, it contributes to d^6 different L_{st} at most, in terms of the local support property of B-spline functions. Consequently, in order to build the matrix, we need no more than $d^6 U$ operations. Namely, we can construct the matrix $(L_{st})_{s,t=1}^{M}$ in $\mathcal{O}(U)$ time. On the other hand, for a fixed s, there are $(2d-1)^3$ non-zero L_{st} at most. Thus, we conclude that the matrix $(L_{st})_{s,t=1}^{M}$ have $\mathcal{O}(M)$ non-zero entries.

To implement the algorithm efficiently, we need an appropriate data structure to represent the sparse matrix. Depending on the analysis of the matrix $(L_{st})_{s,t=1}^{M}$, we choose to combine a linear array with an AVL tree [20], [28] for storage. Every row of the matrix $(L_{st})_{s,t=1}^{M}$ is stored by an AVL tree, and all roots of AVL trees are stored by an linear array.

The pseudo code of the algorithm is illustrated as follows:

```
For u = 1 to U, every point p_u = (x, y, z) {
   // find the index
   Find l_x which satisfies ξ_{l_x} ≤ x < ξ_{l_x+1} and l_y, l_z as well;

   // prepare the auxiliary matrices
```

```
Compute matrix AX = (ax_ij)_{i,j=1}^d, where ax_ij = N_{i+l_x-d+1}(x) N_{j+l_x-d+1}(x),
   and analogous matrix:  AY = (ay_ij)_{i,j=1}^d, AZ = (az_ij)_{i,j=1}^d.

// add the non-zero entries
For s_i := l_x - d + 1 → l_x and t_i := l_x - d + 1 → l_x {
   For s_j := l_y - d + 1 → l_y and t_j := l_y - d + 1 → l_y {
      For s_k := l_z - d + 1 → l_z and t_k := l_z - d + 1 → l_z {
         value = ax_{s_i - l_x + d - 1, t_i - l_x + d - 1} · ay_{s_j - l_y + d - 1, t_j - l_y + d - 1} · az_{s_k - l_z + d - 1, t_k - l_z + d - 1};
         Get the index s = P(s_i, s_j, s_k), t = P(t_i, t_j, t_k);
         Add the value to L_st;
      }
   }
}
```

The algorithm given above is still not optimal. One of the obvious improvements is to exploit the symmetry of matrix $(L_{st})_{s,t=1}^{M}$, which may cut further the computational cost and memory storage in half. Now, using the above data structure and algorithm, we can deal with huge points set, for example, $U \geq 1,000,000$ and $M \geq 100,000$, in several minutes on a PC.

4.3. The matrix form of $N(\mathbf{c})$ term

The term $N(\mathbf{c})$ can be written in the following matrix form:

$$N(\mathbf{c}) = \sum_{u=1}^{U} \|\nabla f(\mathbf{p}_u) - \mathbf{n}_u\|^2 = \begin{bmatrix} c_1' \cdots\cdots c_M' \end{bmatrix} \begin{bmatrix} N_{11} & N_{12} & \cdots & N_{1M} \\ N_{21} & N_{22} & \cdots & N_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ N_{M1} & N_{M2} & \cdots & N_{MM} \end{bmatrix} \begin{bmatrix} c_1' \\ \vdots \\ \vdots \\ c_M' \end{bmatrix}$$

$$-2 \begin{bmatrix} b_1 \cdots\cdots b_M \end{bmatrix} \begin{bmatrix} c_1' \\ \vdots \\ \vdots \\ c_M' \end{bmatrix} + \sum_{u=1}^{U} \left(n_{ux}^2 + n_{uy}^2 + n_{uz}^2 \right) \tag{4.4}$$

where for $s, t = 1, 2, \ldots, M$ and $P(s_i, s_j, s_k) = s$, $P(t_i, t_j, t_k) = t$

$$N_{st} = \sum_{u=1}^{U} \left(d_{su} \cdot d_{tu} + e_{su} \cdot e_{tu} + f_{su} \cdot f_{tu} \right),$$

$$d_{su} = \frac{\partial N_{s_i}(p_{ux})}{\partial x} N_{s_j}(p_{uy}) N_{s_k}(p_{uz}), e_{su} = N_{s_i}(p_{ux}) \frac{\partial N_{s_j}(p_{uy})}{\partial y} N_{s_k}(p_{uz}),$$

$$f_{su} = N_{s_i}(p_{ux}) N_{s_j}(p_{uy}) \frac{\partial N_{s_k}(p_{uz})}{\partial z}, b_s = \sum_{u=1}^{U} \left(d_{su} \cdot n_{ux} + e_{su} \cdot n_{uy} + f_{su} \cdot n_{uz} \right),$$

and analogous terms: d_{tu}, e_{tu}, f_{tu}.

A similar analysis of term $N(\mathbf{c})$ shows that the matrix $(N_{st})_{s,t=1}^{M}$ can be computed in $\mathcal{O}(U)$ operations and stored in a space of $\mathcal{O}(M)$. The algorithm is similar too, so we omit here.

4.4. The matrix form of $G(\mathbf{c})$ term

By performing some calculations of multivariate integrals, $G(\mathbf{c})$ can be expressed in the following matrix form:

$$G(\mathbf{c}) = \iiint_{\Omega} \left(f_{xx}^2 + f_{yy}^2 + f_{zz}^2 + 2f_{xy}^2 + 2f_{xz}^2 + 2f_{yz}^2 \right) \cdot dxdydz$$

$$= \begin{bmatrix} c_1^{'} \cdots\cdots c_M^{'} \end{bmatrix} \begin{bmatrix} G_{11} & G_{12} & \cdots & G_{1M} \\ G_{21} & G_{22} & \cdots & G_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ G_{M1} & G_{M2} & \cdots & G_{MM} \end{bmatrix} \begin{bmatrix} c_1^{'} \\ \vdots \\ \vdots \\ c_M^{'} \end{bmatrix} \tag{4.5}$$

where for $s, t = 1, 2, \ldots, M$ and $P(s_i, s_j, s_k) = s$, $P(t_i, t_j, t_k) = t$

$$G_{st} = o_{st} + p_{st} + q_{st} + 2u_{st} + 2v_{st} + 2w_{st},$$

$$o_{st} = \int_{\Omega_x} \frac{\partial^2 N_{s_i}(x)}{\partial x^2} \frac{\partial^2 N_{t_i}(x)}{\partial x^2} dx \cdot \int_{\Omega_y} N_{s_j}(y) N_{t_j}(y) dy \cdot \int_{\Omega_z} N_{s_k}(z) N_{t_k}(z) dz,$$

$$u_{st} = \int_{\Omega_x} \frac{\partial N_{s_i}(x)}{\partial x} \frac{\partial N_{t_i}(x)}{\partial x} dx \cdot \int_{\Omega_y} \frac{\partial N_{s_j}(y)}{\partial y} \frac{\partial N_{t_j}(y)}{\partial y} dy \cdot \int_{\Omega_z} N_{s_k}(z) N_{t_k}(z) dz,$$

and analogous terms: p_{st}, q_{st}, v_{st}, w_{st}.

Now a natural question is whether the matrix $(G_{st})_{s,t=1}^{M}$ can be built in $\mathcal{O}(U)$ time and stored in $\mathcal{O}(M)$ space? The answer is still "yes".

The underlying fact is that: *suppose that* $GX = (gx_{ij})_{i,j=1}^{m}$, $GPX = (gpx_{ij})_{i,j=1}^{m}$, $GPPX = (gppx_{ij})_{i,j=1}^{m}$, *where for* $i, j = 1$ *to* m*:*

$$gx_{ij} = \int_{\Omega_x} N_i(x) N_j(x) dx, \; gpx_{ij} = \int_{\Omega_x} \frac{\partial N_i(x)}{\partial x} \frac{\partial N_j(x)}{\partial x} dx, gppx_{ij} = \int_{\Omega_x} \frac{\partial^2 N_i(x)}{\partial x^2} \frac{\partial^2 N_j(x)}{\partial x^2} dx,$$

then GX, GPX, $GPPX$ are symmetric and $2d-1$ banded sparse matrices.

Thus, matrix $\left(G_{st}\right)_{s,t=1}^{M}$ is a sparse matrix. Actually, there are at most $(2d-1)^3$ non-zero entries in the each row of matrix $\left(G_{st}\right)_{s,t=1}^{M}$. Based on the above analysis, we present the pseudo code of the algorithm to compute $\left(G_{st}\right)_{s,t=1}^{M}$ as follows:

```
// build the Gram matrices
Compute GX = (gx_ij)_{i,j=1}^m, GPX = (gpx_ij)_{i,j=1}^m, GPPX = (gppx_ij)_{i,j=1}^m,
where gx_ij = ∫_{Ω_x} N_i(x)N_j(x)dx, gpx_ij = ∫_{Ω_x} (∂N_i(x)/∂x)(∂N_j(x)/∂x)dx,
   gppx_ij = ∫_{Ω_x} (∂²N_i(x)/∂x²)(∂²N_j(x)/∂x²)dx.
and analogous matrix: GY, GPY, GPPY, GZ, GPZ, GPPZ.

// add the non-zero entries
For s := 1 → M and t := 1 → M {
   Get the index (s_i, s_j, s_k) = P^{-1}(s), (t_i, t_j, t_k) = P^{-1}(t)
   If (|s_i − t_i| ≤ 2d−1) and (|s_j − t_j| ≤ 2d−1) and (|s_k − t_k| ≤ 2d−1)
{
      // compute the value
      value = gppx_{s_i t_i} · gy_{s_j t_j} · gz_{s_k t_k};
      value += gx_{s_i t_i} · gppy_{s_j t_j} · gz_{s_k t_k};
      value += gx_{s_i t_i} · gy_{s_j t_j} · gppz_{s_k t_k};
      value += 2gpx_{s_i t_i} · gpy_{s_j t_j} · gz_{s_k t_k};
      value += 2gpx_{s_i t_i} · gy_{s_j t_j} · gpz_{s_k t_k};
      value += 2gx_{s_i t_i} · gpy_{s_j t_j} · gpz_{s_k t_k};
      Add the value to G_st;
   }
}
```

The Gram matrices GX, GY, GZ can be built efficiently, see [30] or [7] for details. And with a little modification, the Gram matrices GPX, GPY, GPZ, $GPPX$, $GPPY$ and $GPPZ$ can be constructed similarly.

4.5. The synthetic equation

The optimization problem (3.1) can be solved by the linear equation as fellow

$$\begin{bmatrix} A_{11} & A_{12} \cdots & A_{1M} \\ A_{21} & A_{22} \cdots & A_{2M} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots \\ A_{M1} & A_{M2} \cdots & A_{MM} \end{bmatrix} \begin{bmatrix} c_1^{'} \\ \vdots \\ \vdots \\ c_M^{'} \end{bmatrix} = \begin{bmatrix} w_1 b_1 \\ \vdots \\ \vdots \\ w_1 b_M \end{bmatrix} \tag{4.6}$$

where for $s, t = 1, \ldots, M$, $A_{st} = L_{st} + \omega_1 N_{st} + \omega_2 G_{st}$.

In fact, we may combine all the previous algorithms together and store the $(A_{st})_{s,t=1}^{M}$ only. Then, we can compute it in $\mathcal{O}(U)$ time and the storage requirement is $\mathcal{O}(M)$.

4.6. Solving the synthetic equation

Based on previous analysis, we observe the following feature of the coefficient matrix $(A_{st})_{s,t=1}^{M}$: *The matrix $(A_{st})_{s,t=1}^{M}$ is symmetric, non-singular and highly sparse, if M, ω_2 are large enough.*

Thus we recommend BiConjugate Gradient Stabilized(BiCGSTAB) or Generalized Minimal Residual(GMRES) methods with the preconditioner of incomplete factorization to solve the linear system of equations (4.6). For details, see [13].

5. Adaptive Knot Sequences

In section §2, we have pointed out that the choice of the knot sequences of B-spline functions relates nicely to the final result. From experiments, we have found out that, if the sampling points set is dense($U > 10,000$) and is not too irregular, then using the equally spaced knot sequences produces quite pleasing results. However, if the sampling points set is relatively sparse($U < 10,000$) and non-uniform, then we may obtain better results by taking adaptive knot sequences. In the following, we present two algorithms to choose the knot sequences.

5.1. Density based Algorithm

The basic idea of this algorithm is that, if the sampling points are more dense at some part, then more knots are inserted in the part. In terms of the features of tensor product B-spline function, it is natural to concern the knot sequences in x, y, z directions respectively. Let (x_i, y_i, z_i),

$i = 1, 2, \ldots, U$ be all the sampling points. We sort the x, y, z coordinates in ascending order into three arrays AX, AY, AZ respectively. Now the knot vector in x direction can be constructed as

$$\begin{aligned} &\xi_1 = \cdots \xi_d = MinX, \\ &\xi_{m+1} = \cdots \xi_{m+d} = MaxX, \\ &\xi_{i+d} = (AX[i * S] + AX[i * S + 1])/2.0, \\ &i = 1, \ldots, m - d. \end{aligned} \tag{5.1}$$

where $S = U/(m - d)$ and $[MinX, MaxX]$ is the x-interval of bounding box. The knot vector in y and z directions can be constructed similarly.

Although very simple, the algorithm dramatically improves the quality of the resulting surface. Figure 2 illustrates the reconstruction surface with equal space knots while Figure 3, 4 depicts the reconstruction surface with adaptive knot sequences.

Figure 1. sampling points

Figure 2. uniformly spaced knot sequences

Figure 3. density based algorithm

Figure 4. density based algorithm

Figure 5. octree subdivision

Figure 6. octree subdivision

5.2. Octree based Algorithm

This algorithm is driven by repeated subdivision of the region which bounds the point clouds. At each level, we decide to subdivide octrees or not, in terms of the distribution characteristic of sampling points and the

predefined parameters, for instance the maximum numbers of sampling points of each octree, the maximum level of subdivision, et al. When any terminational condition has been satisfied, the subdivison process stop and return. In this process, we dynamically construct the knot sequences with the help of AVL trees. In order to obtain pleasing results, we should note that the space between adjacent knots should not be too small.

Figure 5, 6 illustrates the reconstruction surface by using above algorithm.

6. Examples and Discussions

6.1. Visualization

Conventional techniques for visualizing implicit functions mainly include polygonization, iso-surface extraction, ray tracing and volume rendering. For more details, see [6]. In our implementation, we use Bloomenthal's polygonizer [5] because of its nice continuation properties and simpleness. All implicit surfaces in current paper are generated by the algorithm. For high quality rendering, we recommend adaptive polygonization algorithms [17], [24], [27] or ray tracing algorithm [14].

6.2. Examples

In terms of sampling property, we illustrate some examples and give some brief discussions. All computations are performed on a consumer level PC with Pentium4 2.4GHz CPU with 1GB DDR RAM.

6.2.1. Uniform and dense sampling

In such case, we recommend using equally spaced knot sequences and quadric B-spline implicit surfaces ($d = 3$) for our reconstruction purpose. Figure 3 and 4 demonstrate that our algorithm works well and produces quite satisfactory results. For details, refer to table 1.

Figure 7. 172,974 points

Figure 8. time 39s, peak RAM 215M

Figure 9. 434,725 points

Figure 10. time 83s, peak RAM 364M

As we have analyzed in section §4, our algorithm is extremely applicable to these examples with large points set because it requires only $\mathcal{O}(U)$ operations and $\mathcal{O}(M)$ storage.

6.2.2. Uniform and sparse sampling

Because of sparse sampling, we need increase the order of B-spline function in order to obtain pleasing result. The underlying fact is that the higher order of B-spline functions results in bigger supports, higher order of continuity and accuracy. In practice, we recommend using equally spaced knot sequences and the cubic B-spline function ($d = 4$). Figure 11, 12 and 13 shows an example with sparse sampling.

Figure 11. 4,539 points

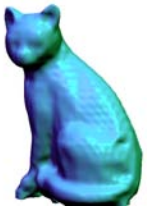

Figure 12. times 3s

Figure 13. times 20s

6.2.3. Non-uniform(adaptive) and sparse sampling

In such case, not only should we increase the order of B-spline function, but also adopt the adaptive knot sequences. Figure 3, 4, 5 and 6 illustrates such an example.

6.2.4. Incomplete or non-uniform(non-adaptive) sampling

For incomplete sampling set, our algorithms to choose adaptive knot sequences in section §5 will generally fail. This is the worst case, and we recommend taking cubic B-spline function ($d = 4$) associated with equally spaced knot sequences of relatively small m, n, l. Figure 14, 15 illustrates an example with an incomplete sampling point set. Figure 16, 17 is an example with non-uniform(non-adaptive) sampling set. For details, refer to table 1.

7. Summary and Future Work

In this paper, we have proposed a fast and low memory consumption surface reconstruction algorithm based on the ITPBS surfaces fitting model suggested by B.Jüttler. In addition, we provide two algorithms

Figure 14. 28,060 points, incomplete sampling

Figure 15. time 27s

Figure 16. 72,545 points, non-uniform sampling

Figure 17. time 77s.

Model	Number of Points	d	m, n, l	M	Computation Time	Number of Triangles	Polygonization Time	Figure
Armadillo	172,974	3	64, 64, 64	29,571	00:39	234,732	00:03	fig 8
Dragon	434,725	3	64, 64, 64	49,261	01:23	334,292	00:05	fig 10
Cat	4,539	3	34, 34, 34	8,061	00:03	155,536	00:02	fig 12
Cat	4,539	4	35, 35, 35	11,128	00:20	154,608	00:02	fig 13
Bunny	28,060	4	25, 25, 25	4,480	00:27	111,360	00:01	fig 15
Venus	72,545	4	35, 35, 35	12,229	01:17	111,640	00:01	fig 17

Table 1
Computation time, memory, polygonization of ITPS for various points set models, and timings are listed as minutes:seconds. All computations are performed on a consumbr level PC with Pentium4 2.4GHz CPU with 1GB DDR RAM.

for generating adaptive knot sequences. We also provided some examples to illustrate our algorithms. Our algorithm requires only $\mathcal{O}(U)$ operations and $\mathcal{O}(M)$ storage, thus it is extremely applicable to sampling points set of large size. Moreover, by appropriately choosing the knot sequences and order of B-spline functions, it can deal with uniform or non-uniform(adaptive) sparse sampling sets. Even for the incomplete or non-uniform(non-adaptive) sampling sets, it still works well.

However, there still exist a number of problems for future research, these include:

- Explore more elaborate mathematical model for surface reconstruction, and take into account more information such as sampling density, noise, color information, etc.
- Gain more insight into automatic choice of parameters, e.g. the knot sequences, the order of B-spline surfaces, the weight ω_1, ω_2 and m, n, l, etc.
- Improve the reconstruction quality of incomplete or non-uniform(non-adaptive) sampling point sets, and be able to faithfully reproduce sharp features.

References

[1] M.Alexa, J.Behr et al, Point set surface, In *IEEE Visualization 2001*, pp.21-28.

[2] N.Amenta, M.Bern, M.Kamvysselis, A new Voronoi-based surfaces reconstruction algorithm, In *SIGGRAPH'98 Conference Proceedings*, pp.415-422, July 1998.

[3] C.L.Bajaj, F.Bernardini, G.Xu, Automatic reconstruction of surfaces and scalar fields from 3d scans, In *SIGGRAPH'95 Conference Proceedings*, pp.109-118, July 1995.

[4] J.F.Blinn, A generalization of algebraic surface drawing, In *ACM Transactions on Graphics*, pp.235-256, vol.1, 1982.

[5] J.Bloomenthal, An implicit surface polygonizer, In *Graphics Gems IV*, pp.324-349, 1994.

[6] J.Bloomenthal, editor, *Introduction to Implicit Surfaces*, Morgan Kaufmann, San Francisco, California, 1997.

[7] C.de.Boor, *A practical guide to splines*, Springer-Verlag, Applied Mathematical Sciences, 1978.

[8] J.C.Carr, R.K.Beatson, J.B.Cherrle, et al, Reconstruction and representation of 3D objects with radial basis functions, In *SIGGRAPH'01 Conference Proceedings*, pp.67-76, July 2001.

[9] B.Curless, M.Levoy, A volumetric method for building complex models from range images, In *SIGGRAPH'96 Conference Proceedings*, pp.303-312, July 1996.

[10] M.Eck, H.Hoppe, Automatic reconstruction of B-spline surfaces of arbitrary topological type, In *SIGGRAPH'96 Conference Proceedings*, pp.325-334, July 1996.

[11] H.Edelsbrunner, E.P.Mücke, Three-dimensional alpha shapes, In *ACM Transactions on Graphics*, pp.43-72, vol.13, 1994.

[12] D.Forsey, R.Bartels, Surface fitting with hierachical splines, In *ACM Transactions on Graphics*, pp.134-161, vol.14, 1995.

[13] G.Golub, C.F.Van Loan, *Matrix Computations*, John Hopkins University Press, 3nd edition, 1996.

[14] J.C.Hart, Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces, In *The Visual Computer*, pp.527-545, Vol.12, 1996.

[15] H.Hoppe, T.DeRose, T.Duchamp, J.McDonald, and W.Stuetzle, Surface reconstruction from unorganized points, In *SIGGRAPH'92 Conference Proceedings*, pp.71-78, vol.26, July 1992.

[16] J.Hoschek, D.Lasser, *Foundamentals of computer aided geometric design*, AK Peters, Wellesley, pp.399-421, 1993.

[17] T.Ju, F.Losasso, S.Schaefer, J.Warren, Dual contouring of hermite data, In *SIGGRAPH'02 Conference Proceedings*, pp.339-346, July 2002.

[18] B.Jüttler, Least-squares fitting of algebraic spline curves via normal vector estimation, In *The Mathematics of Surfaces IX*, Springer-Verlag, R.Cipolla and R.R.Martin(Eds), pp.263-280, 2000.

[19] B.Jüttler, A.Felis, Least-squares fitting of algebraic spline surfaces, In *Advances in Computational Mathematics*, pp.135-152, vol.17, 2002.

[20] D.E.Knuth, *The Art of Computer Programming*, Addison-Wesley, 2nd edition, 1998.

[21] V.Krishnamurthy, M.Levoy, Fitting smooth surfaces to dense polygon meshes, In *SIGGRAPH'96 Conference Proceedings*, pp.313-324, July 1996.

[22] W.Ma, J.Kruth, Parametrization of randomly measured points for least squares fitting of B-spline curves and surfaces, In *Computer Aided Design*, pp.663-675, vol.27, 1995.

[23] S.Muraki, Volumetric shape description of range data using "Blobby Model", In *SIGGRAPH'91 Conference Proceedings*, pp.227-235, July 1991.

[24] Y.Ohtake, A.G.Belyaev, Dual primal mesh optimization for polygonized implicit surfaces, In *7th ACM Symposium on Solid Modeling and Applications*, pp.171-178.

[25] Y.Ohtake, A.Belyaev, M.Alexa, G.Turk, H.Seidel, Multi-level partition of unity implicits, In *Siggraph'03 Conference Proceedings*, pp.463-470, July 2003.

[26] M.J.D.Powell, A review of algorithms for thin plate spline interpolation in two dimensions, In *Advanced topics in multivariate approximation: Proceedings of the International Workshop*, pp.303-322, October 1995.

[27] J.B.Ronald, K.G.Suffer, Visualization of implicit surfaces, In *Computers & Graphics*, pp. 89-107, vol.25, 2001.

[28] S.Sahni, *Data structures, Algorithms, and Applications in C++*, McGraw-Hill, 1997.

[29] D.Shepard, A two dimensional interpolation function for irregular spaced data, In *Proceeding 23rd ACM National Conference*, pp.517-524, 1968.

[30] L.L.Schumaker, *Spline Functions: Basic Theory*, John-Wiley, 1980.

[31] H.Zhao, S.Osher, Visualization, analysis and shape reconstruction of unorganized data sets, In *Geometric Level Set Mehods in Imaging, Vision and Graphics*, Springer-Verlag, S.Osher and N.Paragios(Eds), 2002.