A Fast Solution to the Surface Reconstruction Problem

B. C. Vemuri[1] and S. H. Lai[2]

[1]Department of Computer & Information Sciences
[2]Department of Electrical Engineering
University of Florida, Gainesville, FL 32611
email:vemuri@scuba.cis.ufl.edu

Abstract

Surface reconstruction from range data acquired using a vareity of sources has been a very active area of research in computational vision over the past decade. Generalized splines have emerged as the single most popular approximation tool to this end. In this paper we present a new and fast algorithm for solving the surface reconstruction problem using the membrane spline which yields a C^0 surface. Our algorithm for dense data constraints takes $O(N)$ computational time, where N is the number of nodes in the discretization. For sparse data constraints, the algorithm requires $O(\log N/\pi^2)$ iterations with each iteration taking $O(N)$ time. The number of iterations in this case depends on a prespecified error tolerance. We demonstrate the algorithm performance on synthesized sparse range data.

1 Introduction

The problem of reconstructing and representing three-dimensional shapes has received an enormous amount of attention in vision research for the past decade. 3D shapes may be described either by the bounding surfaces or by the enclosed volumes. In this paper we will be concerned only with the surface based representations constructed from depth/range data. Depth or range data may be obtained in numerous ways eg., from stereo, from shading, from motion, or directly from a laser ranging device. In all these cases with the exception of the last technique, the data obtained are sparse due to the nature of the solution methods used. For eg., in depth from stereo techniques, the range data are specified only at a sparse set of locations eg., along object boundaries. The "filling in" may be achieved by using a surface interpolation technique. There are numerous surface interpolation methods in numerical analysis [6] and computer vision literature [1]. Among these techniques, the most popular ones in vision have been those using a variational principle formulation [9, 10, 1]. This formulation leads to what are known as energy-based surface models which may be used for interpolating sparse data, to smooth noisy depth data, and also to integrate data from multiple sensors or view points.

We will now briefly present a variational principle formulation of the surface reconstruction problem [9]. The solution to the general surface reconstruction problem is defined by a configuration of minimal energy $\mathcal{E}$ of a physical system comprised of the thin plate patches, the membrane strips, and the springs. To solve the variational principle an explicit expression must be obtained for the potential energy functional $\mathcal{E} : V \mapsto \Re$. V is the admissible space of the variational principle. The admissible space V consists of functions which represent possible plate-membrane configurations, a (Sobolev) space of bounded-energy functions. The precise expression for the variational principle

is given by

$$\mathcal{E}_s(v) = \int\int_{\Omega} \rho(x,y)\{\tau(x,y)(v_{xx}^2 + 2v_{xy}^2 + v_{yy}^2) + [1-\tau(x,y)](v_x^2 + v_y^2)\}\,\mathrm{d}x\mathrm{d}y, \tag{1}$$

where $v(x,y)$ is the admissible deflection function and v_x, v_y its partial derivatives assumed to be small. $\rho(x,y)$ and $\tau(x,y)$ are called continuity control functions, they constitute an explicit representation of depth and orientation discontinuities, respectively. The functions range over the interval $[0,1]$ and need not vary continuously. Setting the continuity control functions ρ and τ is tantamount to prior knowledge of the location of the discontinuities. Such knowledge may be obtained from registered multi-sensor data.

The above energy expression serves as a stabilizer in the overall variational principle for the surface reconstruction problem. To the stabilizer, we add data constraints via what are known as penalty terms. For the surface reconstruction from depth data, the following penalty term which measures the discrepancy between the surface and data weighted by the uncertainty in the data may be used,

$$\mathcal{E}_d(v) = \frac{1}{2}\sum c_i\,(v(x_i,y_i) - d_i)^2 \tag{2}$$

Where, d_i are the depth data points specified in the domain Ω and c_i are the uncertainty associated with the data. The total energy is

$$\mathcal{E} = \mathcal{E}_d + \lambda\mathcal{E}_s \tag{3}$$

Where λ is the regularization parameter that controls the amount of smoothing performed. The goal is to find a u that minimizes the total potential energy $\mathcal{E}(v)$ in equation 3.

To compute a numerical solution to the above minimization problem, we first discretize the functionals $\mathcal{E}_s$ and $\mathcal{E}_d$ using finite element techniques [9]. For constant $\rho(x,y)$ and $\tau(x,y)$ the energy function is a quadratic form. The energy due to the data compatibility term in discrete form becomes,

$$E_d(\mathbf{x},\mathbf{d}) = \frac{1}{2}(\mathbf{x}-\mathbf{d})^T\mathbf{K}_d(\mathbf{x}-\mathbf{d}). \tag{4}$$

Where $\mathbf{x}$ is the discretized surface, $\mathbf{d}$ are the data points, and $\mathbf{K}_d$ is a diagonal matrix (for uncorrelated noise in the data) containing the uncertainty σ_i associated with the data points. The smoothness energy in discrete form is

$$E_s(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}_s\mathbf{x} \tag{5}$$

where $\mathbf{K}_s$ is a very large $(n^2 x n^2)$, sparse and banded matrix called the stiffness matrix. The resulting energy function is a quadratic in $\mathbf{x}$

$$E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{K}\mathbf{x} - \mathbf{x}^T\mathbf{b} + c \tag{6}$$

with $\mathbf{K} = \lambda\mathbf{K}_s + \mathbf{K}_d$ and $\mathbf{b} = \mathbf{K}_d\mathbf{d}$. The minimum of this energy function $\mathbf{u}$ is found by solving the large sparse linear system

$$\mathbf{K}\mathbf{x} = \mathbf{b} \tag{7}$$

Another way of approaching the solution to the minimization of the energy $\mathcal{E}$ is using calculus of variations [9]. The necessary condition for the minimum of the energy functional is a partial differential equation called the Euler-Lagrange equation. Discretization of this PDE leads to a system of algebraic equations as given in 7. The large (n^2, n^2) sparse linear system in equation 7 can be solved using iterative techniques and the best known algorithms either multi-grid method [8] or hierarchical conjugate gradient [7] take $O(N\log N)$ time where, $N = n^2$. *In this paper, we develop a fast algorithm to solve the above linear system with the plate terms disabled. Our algorithm takes $O(N)$ time for dense data constraints and $O((N\log N)/\pi^2)$ time for sparse data constraints.* Our approach is based on converting this linear system into a Lyapunov matrix equation and then solving it using an iterative scheme

called the alternating direction implicit (ADI) method. We will describe the details of our technique in a subsequent section.

The rest of the paper is organized as follows, first we formulate the problem and derive the necessary results in section 2. Numerical solution technique is then described in section 3. We present some experimental results depicting the performance of our algorithm in section 4.

2 Problem Formulation

In this section, we will pose and formulate the problem of solving the algebraic system of equations resulting from the discretization of the Poisson equation or the minimization of the discrete version of the constrained variational principle discussed in the previous section.

The surface reconstruction problem can be posed as follows: Given a dense or sparse set of depth data, find a surface which passes close to the data and is smooth.

The stabilizer discussed in the previous section imposes smoothness constraints on the surface reconstruction problem leading to a C^0 surface i.e., the membrane spline or a C^1 surface namely, the thin plate spline or a combination thereof called the thin-plate-membrane spline. As mentioned earlier, the numerical solution to the surface reconstruction problem involves solving a linear system $\mathbf{Kx} = \mathbf{b}$. The matrix $\mathbf{K}$ is very large and sparse but is *not tightly banded*. This is primarily due to the fact that a vector has been used to represent a 2D array which leads to representing neighboring interactions in 2D by interactions between very distant elements in 1D. No existing algorithms for such a block banded matrix lead to an $O(N)$ solution. The $\mathbf{K}$ matrix is a sum of two matrices $\mathbf{K}_s$ and $\mathbf{K}_d$ with $\mathbf{K}_s$ containing the membrane molecules defined on the interior and boundary of Ω as well as at discontinuities (see figure 1). The $\mathbf{K}_d$ matrix is a diagonal matrix containing non-zero weights. We will now describe

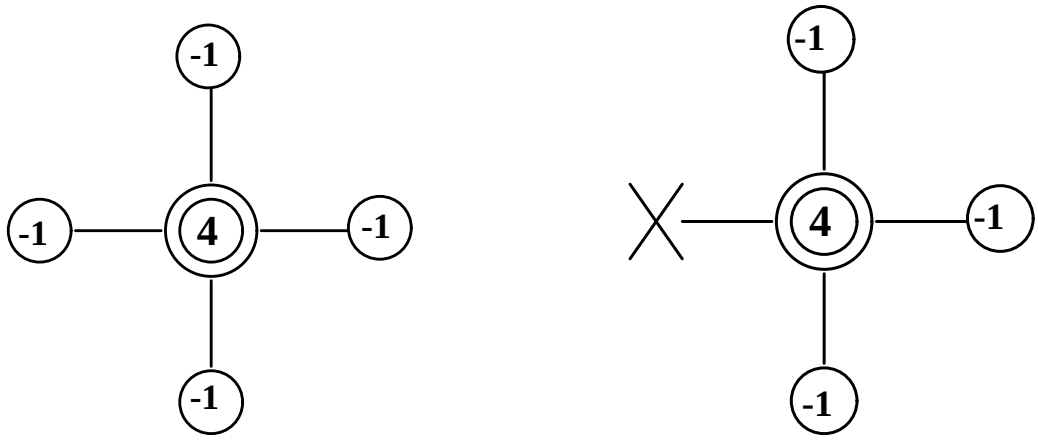

Figure 1: Computational molecules, (a) in the interior of Ω and (b) at the boundary of Ω

the derivation to transform the above linear system to a Lyapunov matrix equation.

2.1 The Lyapunov Matrix Equation

To transform the linear system $\mathbf{Kx} = \mathbf{b}$ into a Lyapunov matrix equation, we will first split the matrix $\mathbf{K}_s$ into two components.

$$\mathbf{K}_s = \mathbf{K}_0 + \mathbf{U}_s \mathbf{V}_s^T, \tag{8}$$

where

$$\begin{aligned}
\mathbf{K}_0 &= \begin{bmatrix} \mathbf{D} & -\mathbf{I} & & & \\ -\mathbf{I} & \ddots & \ddots & 0 & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -\mathbf{I} \\ & 0 & & -\mathbf{I} & \mathbf{D} \end{bmatrix}, \\
\mathbf{U}_s &= \begin{bmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_k \end{bmatrix}, \\
\mathbf{V}_s &= \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_k \end{bmatrix},
\end{aligned}$$

and

$$\mathbf{D} = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{bmatrix} \in \Re^{n \times n}.$$

Each of the column vectors $\mathbf{u}_i$ and $\mathbf{v}_i \in \Re^{n^2 \times 1}$ contain only one or two nonzero entry mapped to the boundary of the discretized Ω or the discontinuity locations. The matrix $\mathbf{K}_0$ is a well-structured matrix, the contents of which were discussed in the last section. The matrix $\mathbf{U}_s \mathbf{V}_s^T \left(= \sum_{i=1}^{k} u_i v_i^T \right)$, contains entries indicating the difference between the matrices $\mathbf{K}_s$ and $\mathbf{K}_0$.

The special matrix $\mathbf{K}_0$ can be decomposed as the sum of two tensor products of matrices $\mathbf{A}$ and $\mathbf{I}$, i.e.,

$$\mathbf{K}_0 = \mathbf{A} \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{A}, \tag{9}$$

with

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} \in |Re^{n \times n},$$

and $\otimes$ is the tensor (Kronecker) product.

By using the special structure of the matrix $\mathbf{K}_0$, we can rewrite any linear system of the form $\mathbf{K}_0 \mathbf{z} = \mathbf{f}$ as the following Lyapunov matrix equation

$$\mathbf{AZ} + \mathbf{ZA} = \mathbf{F}, \tag{10}$$

where $\mathbf{Z}$ and $\mathbf{F}$ are the original $(n \times n)$ matrices corresponding to their concatenated $n^2 \times 1$ vectors $\mathbf{z}$ and $\mathbf{f}$ respectively.

Therefore, instead of solving the original large linear system $\mathbf{Kx} = \mathbf{b}$ in equation 7, our approach is to solve the corresponding Lyapunov matrix equation with a modified right-hand side matrix $\mathbf{B}$. Notice that the matrix $\mathbf{A}$ in

equation 10 is very tightly banded (tridiagonal). In addition to being tridiagonal, the matrix $\mathbf{A}$ is also SPD and Toeplitz. We can use the ADI (Alternating Direction Implicit) method to solve this Lyapunov matrix equation in $O(N)$ operations for dense data and in $O(N \log N/\pi^2)$ for the sparse data constraints. We will describe he ADI method in the next section.

2.2 Modification of the Right-hand Side

Our method is based on the ADI technique tailored for the Lyapunov matrix equation with a tridiagonal, SPD and Toeplitz matrix $\mathbf{A}$. In the following, we are going to show how the original $\mathbf{Kx} = \mathbf{b}$ problem can be converted to an equivalent linear system form $\mathbf{K}_0\mathbf{x} = \mathbf{b}'$ by modifying its right-hand side. The solution of $\mathbf{K}_0\mathbf{x} = \mathbf{b}'$ may be obtained by solving the corresponding Lyapunov matrix equation containing the modified right-hand side.

The derivation is mainly based on the Sherman-Morrison-Woodbury formula [3]

$$(\mathbf{A} + \mathbf{U}\mathbf{V}^T)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{I} + \mathbf{V}^T\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}^T\mathbf{A}^{-1}, \tag{11}$$

where $\mathbf{A} \in \Re^{n\times n}$, $\mathbf{U} \in \Re^{n\times m}$, $\mathbf{V} \in \Re^{m\times n}$, and $\mathbf{I} \in \Re^{m\times m}$ is the identity matrix.

By decomposing every nonzero entry of the diagonal matrix $\mathbf{K}_d$ into the outer product of $\mathbf{u}'_i$ and $\mathbf{v}'_i$ and putting them into $\mathbf{U}$ and $\mathbf{V}$ as follows:

$$\begin{aligned} \mathbf{U} &= \left[\begin{array}{cccccc} \mathbf{u}_1 & \ldots & \mathbf{u}_k & \mathbf{u}'_1 & \ldots & \mathbf{u}'_{k'} \end{array}\right], \\ \mathbf{V} &= \left[\begin{array}{cccccc} \lambda\mathbf{v}_1 & \ldots & \lambda\mathbf{v}_k & \mathbf{v}'_1 & \ldots & \mathbf{v}'_{k'} \end{array}\right], \end{aligned}$$

and using equations (3) and (4), we have

$$\begin{aligned} \mathbf{K} &= \lambda\mathbf{K}_0 + \lambda\mathbf{U}_s\mathbf{V}_s^T + \mathbf{K}_d, & (12) \\ &= \lambda\mathbf{K}_0 + \mathbf{U}\mathbf{V}^T. & (13) \end{aligned}$$

By substituting the Sherman-Morrison-Woodbury formula into $\mathbf{x} = \mathbf{K}^{-1}\mathbf{b}$, we get the following result

$$\mathbf{x} = \mathbf{K}_0^{-1}(\mathbf{b} - \mathbf{U}\mathbf{y})/\lambda, \tag{14}$$

with

$$\mathbf{y} = (\mathbf{I} + \lambda^{-1}\mathbf{V}^T\mathbf{K}_0^{-1}\mathbf{U})^{-1}(\lambda^{-1}\mathbf{V}^T\mathbf{K}_0^{-1}\mathbf{b}). \tag{15}$$

Therefore, the right-hand side of $\mathbf{K}_0\mathbf{x} = \mathbf{b}'$ is given by $\mathbf{b}' = (\mathbf{b} - \mathbf{U}\mathbf{y})/\lambda$. The vector $\mathbf{y} \in \Re^{(k+k')\times 1}$ participates in modifying the values of the $\mathbf{b}$ vector at locations corresponding to the data constraints, discontinuities and boundary of Ω. Computation of vector $\mathbf{y}$ is nontrivial and will be the focus of discussion in a subsequent section.

3 Numerical Solution

Our numerical algorithm can be described as follows:

1. Compute $\mathbf{c} := \lambda^{-1}\mathbf{V}^T\mathbf{K}_0^{-1}\mathbf{b}$.
2. Form the matrix $\mathbf{E} := \mathbf{I} + \lambda^{-1}\mathbf{V}^T\mathbf{K}_0^{-1}\mathbf{U}$.
3. Solve $\mathbf{E}\mathbf{y} = \mathbf{c}$ and modify the right-hand side to $\mathbf{b}' = (\mathbf{b} - \mathbf{U}\mathbf{y})/\lambda$.

4. Solve $\mathbf{AX} + \mathbf{XA} = \mathbf{B}'$ where $\mathbf{B}'$ is the matrix form of $\mathbf{b}'$.

In steps (1) and (4), the ADI method is used to solve the Lyapunov matrix equations with different right-hand side $\mathbf{B}$ (the matrix form of vector $\mathbf{b}$) and $\mathbf{B}'$. After computing $\mathbf{x}_0 = \mathbf{K}_0^{-1}\mathbf{b}$ using the ADI method, the vector $\mathbf{c}$ is obtained by multiplying $\mathbf{x}_0$ with $\mathbf{V}^T$ whose rows serve as sampling vectors as each of them contain only single nonzero entries.

The linear system $\mathbf{Ey} = \mathbf{c}$ is nonsymmetric and needs to be solved efficiently. In this paper, we use the biconjugate gradient (BCG) technique [] in a hierarchical basis [7], to solve this problem. In the following, the details of the ADI method for the Lyapunov equation, the formation of matrix $\mathbf{E}$, and the solution to the nonsymmetric linear system $\mathbf{Ey} = \mathbf{c}$ will be given.

3.1 ADI Method for the Lyapunov Equation (Stages (1) and (4) of the algorithm)

The ADI method for solving a Lyapunov matrix equation is described in [5]. For any Lyapunov matrix equation of the form $\mathbf{AX} + \mathbf{XA} = \mathbf{B}$, the ADI method involves the following steps in each iteration $j = 1, 2, \ldots, J$,

$$(\mathbf{A} + p_j\mathbf{I})\mathbf{X}_{j-\frac{1}{2}} = \mathbf{B} - \mathbf{X}_{j-1}(\mathbf{A} - p_j\mathbf{I}) \quad (16)$$

$$\mathbf{X}_j(\mathbf{A} + p_j\mathbf{I}) = \mathbf{B} - (\mathbf{A} - p_j\mathbf{I})\mathbf{X}_{j-\frac{1}{2}} \quad (17)$$

The nice properties of matrix A help to advance the ADI iterations very efficiently in each step of the technique. Since A is SPD and tridiagonal, we can use Cholesky decomposition for the tridiagonal SPD matrix $(\mathbf{A} + p_j\mathbf{I})$ to compute the updated solution. *This solution update requires only $O(N)$ time per iteration.*

We now examine the convergence of the ADI method for the problems in stages (1) and (4) of our algorithm. Let the initial $\mathbf{X}_0 = \mathbf{0}$ (zero matrix), and $\triangle\mathbf{X}_t = \mathbf{X}_t - \mathbf{X}^*$, where $\mathbf{X}^*$ is the true solution of the Lyapunov matrix equation, $\lambda_1, \lambda_2, \ldots, \lambda_n$, and $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n$ be the eigenvalues and the eigenvectors of $\mathbf{A}$ respectively. For our particular matrix $\mathbf{A}$, the left and right eigenvalues and eigenvectors are the same, and the eigenvectors are orthogonal, i.e.,

$$\mathbf{A} = \mathbf{QDQ}, \quad (18)$$

where

$$\begin{aligned} \mathbf{Q} &= [\mathbf{q}_1 \ \mathbf{q}_2 \ \cdots \ \mathbf{q}_n], \\ \mathbf{D} &= diag[\lambda_1, \lambda_2, \cdots, \lambda_n], \\ \mathbf{q}_k &= \sqrt{\frac{2}{n+1}} \begin{bmatrix} sin\frac{k\pi}{n+1} \\ sin\frac{2k\pi}{n+1} \\ \vdots \\ sin\frac{nk\pi}{n+1} \end{bmatrix}, \end{aligned}$$

and $\lambda_k = 2 - 2\cos\frac{k\pi}{n+1}$ for $k = 1, \ldots, n$. Notice that $\mathbf{Q}$ is the matrix corresponding to the discrete sine transform and $\mathbf{Q} = \mathbf{Q}^T = \mathbf{Q}^{-1}$.

By taking the tensor product of the eigen vectors, $\mathbf{q}_k \otimes \mathbf{q}_l, k, l = 1, \ldots, n$, we can generate an orthogonal basis for $\Re^{n\times n}$ and express the true solution $\mathbf{X}^*$ in this basis as follows:

$$\mathbf{X}^* = \sum_{k=1}^{n}\sum_{l=1}^{n} a_{kl}(\mathbf{q}_k \otimes \mathbf{q}_l). \quad (19)$$

Subtracting each of the equations 16 and 17 from the Lyapunov equation $\mathbf{AX} + \mathbf{XA} = \mathbf{B}$, and expressing $\mathbf{X}$ in the

above tensor product basis, we have the following result for the error after t iterations,

$$\triangle \mathbf{X}_t = -\sum_{k=1}^{n}\sum_{l=1}^{n}\left[\prod_{j=1}^{t}(\frac{\lambda_k - p_j}{\lambda_k + p_j})(\frac{\lambda_l - p_j}{\lambda_l + p_j})\right] a_{kl}(\mathbf{q}_k \otimes \mathbf{q}_l). \tag{20}$$

The error $\triangle \mathbf{X}_t$ is bounded by

$$|\triangle \mathbf{X}_t| \leq \mu_t |\mathbf{X}^*|, \tag{21}$$

where

$$\mu_t = \max_{k,l} |\prod_{j=1}^{t}(\frac{\lambda_k - p_j}{\lambda_k + p_j})(\frac{\lambda_l - p_j}{\lambda_l + p_j})|.$$

Obviously, the error bound depends on the choice of the ADI parameters p_j. Choosing the optimal set of ADI parameters p_j's leads to the ADI minimax parameter problem. Given the bounds for the eigenvalues of the SPD matrix $\mathbf{A}$ as, $\forall k, 0 < a \leq \lambda_k \leq b$, we have,

$$\mu_t \leq \min_{p_1,\ldots,p_t} \max_{0<a\leq\lambda\leq b} \prod_{j=1}^{t}(\frac{\lambda - p_j}{\lambda + p_j})^2. \tag{22}$$

The right-hand side is the minimax ADI parameter problem. The solution to this minimax problem gives the optimum set of ADI parameters. Computing the optimal choice of the ADI parameters is described in [5].

The minimax analysis also provides the solution to the number of iterations needed in ADI to achieve a specified error tolerance ϵ in $\mathbf{X}$ such that $\|\triangle \mathbf{X}_J\| \leq \epsilon \|\mathbf{X}_{true}\|$. This number of iterations is given by

$$J = \lceil \frac{ln\frac{4}{\epsilon} ln\frac{4}{k'}}{\pi^2} \rceil, \tag{23}$$

where

$$\begin{aligned} k' &= \frac{1}{m + \sqrt{m^2 - 1}}, \\ m &= \frac{1}{2}(\kappa_2(\mathbf{A}) + \frac{1}{\kappa_2(\mathbf{A})}), \end{aligned}$$

$\lceil z \rceil$ denotes the smallest integer larger than z, and κ_2 is the ratio of the largest and smallest eigenvalues, i.e.,the 2-condition number of $\mathbf{A}$.

In the dense and sparse data cases, we have different $\mathbf{A}$ matrices in the Lyapunov matrix equations. In the following, we give the analysis of numbers of iterations needed for each of these different cases.

3.1.1 Dense Data Case

The data compatibility matrix $\mathbf{K}_d$ for dense data case is the identity matrix. We can add this identity matrix to the matrix $\mathbf{K}_0$ to form the Lyapunov matrix equation with different matrix $\mathbf{A}$ i.e., it is the original $\mathbf{A}$ plus the scaled identity matrix $\frac{1}{2\lambda}\mathbf{I}$. This doesn't change the eigenvectors of A, but its eigenvalues are shifted by $\frac{1}{2\lambda}$. Therefore, all the eigenvalues of $\mathbf{A}$ are bounded between $\frac{1}{2\lambda}$ and $4 + \frac{1}{2\lambda}$ independent of the size of the matrix $\mathbf{A}$. From equation 23, we can see the number of iterations needed will be bounded by a constant number for a given error ratio. Hence, the number of iterations needed in ADI is independent of the size of the problem for the dense data case. Thus, the computational time for the dense data case is $O(N)$.

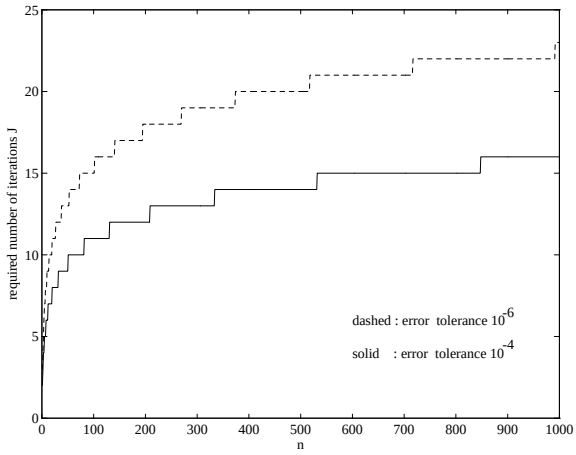

Figure 2: The number of iterations needed in the ADI method with respect to the size n of the matrix $\mathbf{A} \in \Re^{n \times n}$

3.1.2 Sparse Data Case

If we put the largest and smallest eigenvalues, $2 - 2\cos\frac{n\pi}{n+1}$ and $2 - 2\cos\frac{\pi}{n+1}$, into equation 23, we can get the following approximation for the number of iterations,

$$J \approx \ln\frac{4}{\epsilon}(\log\frac{16}{\pi^2} + \frac{\log n}{\pi^2/2}). \tag{24}$$

In figure 2, we plot the number of iterations as a function of the input matrix size $(n \times n)$. We can see that the number of iterations required is very small for practical ranges of matrix sizes typically encountered in vision problems.

The as is application of *minmax* analysis to the ADI method for our problem leads to a very loose upper bound on the number of iterations. This is due to the fact that no information about the spectrum of the membrane interpolant is incorporated into the analysis. However, we do have the a priori knowledge about the spectrum of the exact solution, i.e. the coefficients a_{kl}'s in equation 19. Taking this information into account leads to a different minimax problem which will yield a different choice of ADI parameters. Thus, we can get a tighter bound for the required number of iteration than that given in equation 23. We leave this improvement as the focus of our future research.

3.2 Formation of the Matrix E (Stage (2))

Each entry (i, j) in matrix $\mathbf{E}$ can be written as

$$\mathbf{E}_{(i,j)} = \lambda^{-1}\mathbf{v}_i^T\mathbf{K}_0^{-1}\mathbf{u}_j + \delta(i - j). \tag{25}$$

Recall that every vector $\mathbf{v}_i$ and $\mathbf{u}_j$ contain only one or two nonzero elements. This means that $\mathbf{v}_i^T\mathbf{K}_0^{-1}\mathbf{u}_j$ is the sampling at the locations v_i of the impulse response of the linear system $\mathbf{K}_0^{-1}$ or the Lyapunov system with the impulse at the locations $\mathbf{u}_j$. Unfortunately, this system is not shift-invariant and this makes the problem of efficiently forming the matrix $\mathbf{E}$ more difficult.

Assume the nonzero elements of $\mathbf{v}_i$ and $\mathbf{u}_j$ are located at (p_i, q_i) and (p_j, q_j), respectively. We can find an analytic form of the solution at (p_i, q_i) to the Lyapunov matrix equation $\mathbf{AX} + \mathbf{XA} = \mathbf{B}$, with $\mathbf{B}(k, l) = \delta(k - p_j, l - q_j)$ by

substituting equation 18 for $\mathbf{A}$ and solving for $\mathbf{X}$. This leads to,

$$X(p,q) = f_n(p_i - p_j, q_i - q_j) + f_n(p_i + p_j, q_i + q_j) - f_n(p_i - p_j, q_i + q_j) - f_n(p_i + p_j, q_i - q_j), \tag{26}$$

where

$$f_n(p,q) = \frac{1}{8}\sum_{k=1}^{n}\sum_{l=1}^{n}\frac{\cos\frac{kp\pi}{n+1}\cos\frac{lq\pi}{n+1}}{2-\cos\frac{k\pi}{n+1}-\cos\frac{l\pi}{n+1}}. \tag{27}$$

The function $f_n(p,q)$, for $0 \le p,q \le n+1$, should be precomputed and stored as an $((n+2)\times(n+2))$ lookup table. Then each entry in the matrix E can be efficiently calculated from the above equation and the look-up table.

3.3 Solution to Ey = c (Stage 3)

The linear system $\mathbf{Ey} = \mathbf{c}$ is nonsymmetric and indefinite. For the dense data case, the size of the matrix $\mathbf{E}$ is $k \times k$, where k is the number of discontinuities. For the sparse data case, The size of the matrix $\mathbf{E}$ is $(k+k')\times(k+k')$, where k is the number of boundary points and discontinuity locations and k' is the number of data points. A reasonable assumption is $(k+k') \approx O(n)$. Consequently, it is important to have a fast algorithm to compute the solution $\mathbf{y}$ to this nonsymmetric linear system.

The conjugate-gradient(CG) method is a very powerful iterative scheme to solve the symmetric positive definite linear system. A natural generalization of the CG-type method to solve the general nonsymmetric linear system is called the biconjugate gradient(BCG) method. The BCG algorithm [4] is as follows.

0. Choose $\mathbf{y}_0$, and set $\mathbf{q}_0 = \mathbf{r}_0 = \mathbf{c} - \mathbf{E}\mathbf{y}_0$; Choose $\tilde{\mathbf{r}}_0 \neq \mathbf{0}$, and set $\tilde{\mathbf{q}}_0 = \tilde{\mathbf{r}}_0$;

1. Compute $\alpha_k^N = \tilde{\mathbf{r}}_{k-1}^T\mathbf{r}_{k-1}, \alpha_k^D = \tilde{\mathbf{q}}_{k-1}^T\mathbf{E}\mathbf{q}_{k-1}$, and $\alpha_k = \alpha_k^N/\alpha_k^D$;

2. Update $\mathbf{y}_k = \mathbf{y}_{k-1} + \alpha_k\mathbf{q}_{k-1}$;

3. Set $\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_k\mathbf{E}\mathbf{q}_{k-1}$, and $\tilde{\mathbf{r}}_k = \tilde{\mathbf{r}}_{k-1} - \alpha_k\mathbf{E}^T\tilde{\mathbf{q}}_{k-1}$;

4. Compute $\beta_k^N = \tilde{\mathbf{r}}_k^T\mathbf{r}_k$, and $\beta_k = \beta_k^N/\alpha_k^N$;

5. Set $\mathbf{q}_k = \mathbf{r}_k + \beta_k\mathbf{q}_{k-1}$, and $\tilde{\mathbf{q}}_k = \tilde{\mathbf{r}}_k + \beta_k\tilde{\mathbf{q}}_{k-1}$;

6. If $\mathbf{r}_k \approx \mathbf{0}$ or $\tilde{\mathbf{r}}_k \approx \mathbf{0}$, stop; else go to 1.

Like CG , The operations and storage requirement per step in BCG is constant. The convergence behavior is similar to the CG method except that the BCG algorithm is susceptible to breakdowns and numerical instabilities. To be more specific, division by 0 (or a number very close to 0) may occur in computing α_k and β_k. These two different breakdowns may be avoided by using the look-ahead Lanczos algorithm or the quasi-minimal residual(QMR) algorithm [2].

As in the CG-algorithm, preconditioning techniques can speed up the convergence of the BCG algorithm. In this paper, we use the technique similar to the hierarchical conjugate gradient(HCG) method, which involves smoothing the residual vector in each step. However, in BCG, we have to smooth two residual vectors r_k and $\tilde{r_k}$ in each step. The smoothing operation is the same as that in HCG. We call this preconditioned BCG method the hierarchical biconjugate gradient(HBCG) method. In the BCG algorithm, each iteration requires that $\mathbf{E}\mathbf{q}_{k-1}$ and $\mathbf{E}^T\tilde{\mathbf{q}}k-1$ be computed. Since the matrix $\mathbf{E}$ is a dense matrix, the computational cost for the direct matrix-vector products in each iteration is may be too much. After examining the matrix $\mathbf{E}$, we find that most of its entries away from the diagonal are very close to 0 and the overall structure of $\mathbf{E}$ is very smooth. Therefore, we can approximate the matrix and vector with very low-order polynomials and use the polynomial integration to approximate the inner product in the off-diagonal insignificant region. This leads to a significantly fast scheme in forming the required matrix vector products.

4 Experimental Result

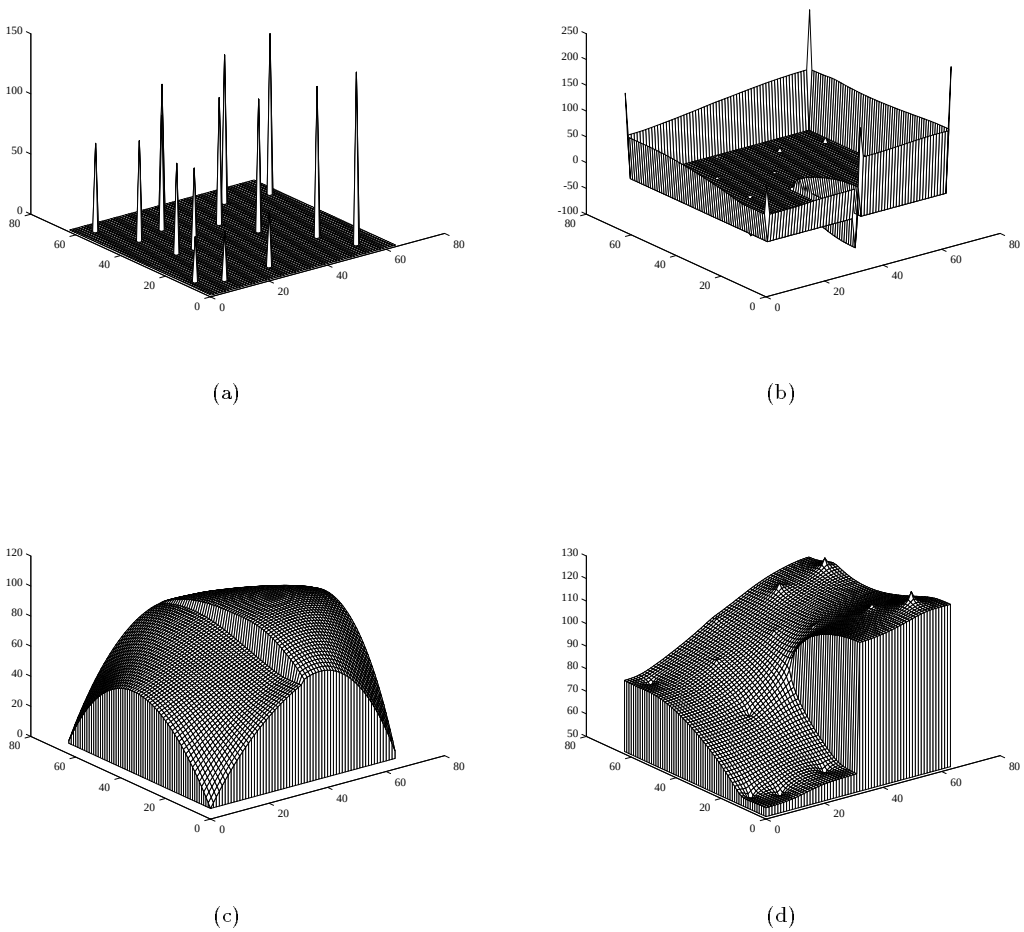

Figure 3: Surface reconstruction example; (a) original sparse data i.e., RHS of $\mathbf{Kx} = \mathbf{b}$, (b) modified right-hand side i.e., RHS of $\mathbf{K}_0\mathbf{x} = \mathbf{b}'$, (c) solution after 1 iteration of ADI method and (d) after 16 iterations.

Our algorithm was tested on several data sets (both dense and sparse) of varying input sizes. In this section, we present one example of the algorithm performance on a 64×64-node surface reconstruction problem. The input data set is very sparse and contains 15 data points randomly distributed in the plane as shown in figure 3.3. The discontinuity locations are specified along the line between coordinates (1,32) and (30,32).

The first step is to find the modified right-hand side by using the numerical method presented in section 3. The modified right-hand side $\mathbf{B}'$ is shown in figure 3.3. This depicts the changes that are made to the original data before applying the ADI method. Note the changes along the boundary, and to the data values within the boundary. The ADI iterations are then employed to obtain the interpolated surface. From equation 23, it is evident that just 16 iterations can guarantee the error tolerance ϵ to be within 10^{-6}. We depict the surfaces after 1 and 16 ADI iterations in figure 3.3.

In figure 3.3, we can see that just one ADI iteration can produce the global shape of the exact surface. This is because we advance the ADI iterations with the parameters p_j's starting with a small value and proceeding toward large values. The smaller parameters are used to recover the low frequency detail of the surface and the large values refine the shape with high frequency components. Consequently, it is not surprising that the global shape was recovered in one iteration.

This phenomena of global to local shape recovery is quite different from the other iterative numerical methods. Most of the other numerical methods do not exhibit this type of spectral characteristic during the iterations. In addition to the fast convergence rate, another advantage of the ADI method is that we can make use of the spectral properties of the prior surface model (in our case the membrane spline) to speed up the convergence rate. This can be achieved via adaptation of the ADI parameters to the spectrum of this model constrained by the given data.

5 Conclusion

In this paper, we presented a new and fast algorithm for solving the surface reconstruction problem using a membrane spline interpolant. The problem is formulated as the solution to a Lyapunov matrix equation with an appropriate right-hand side. We derive this right-hand side and present fast numerical methods to compute it. To solve the Lyapunov matrix equation, we use the ADI algorithm and obtain convergence (for a prespecified tolerance) in very few iterations for different sizes of the **A** matrix. In addition, we can speed up the convergence rate of the ADI method by making use of the observation that the ADI parameters are related to different amounts of error reduction in different spectral ranges. We will address this issue in our future research.

References

[1] R.M. Bolle and B.C. Vemuri. "On three-dimensional surface reconstruction methods,". *IEEE Trans. Pattern Anal. Machine Intell.*, 13(1):1–13, 1991.

[2] R.W. Freund and N.M. Nachtigal. "QMR: a quasi-minimal residual method for non-Hermitian matrices,". Technical Report 90.51, RIACS, NASA Ames Research Center, Moffett Field, 1990.

[3] G.H. Golub and C.F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, 2nd edition, 1989.

[4] C. Lanczos. "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators,". *J. Res. Natl. Bur. Stand.*, 45:255–282, 1950.

[5] A. Lu and E. L. Wachspress. "Solution of Lyapunov equations by alternating direction implicit iteration,". *Computers Math. Applic.*, 21(9):43–58, 1991.

[6] L.L. Schumaker. "Fitting surfaces to scattered data,". In G.G. Lorentz, C.K. Chui, and L.L. Schumaker, editors, *Approximation Theory II*, pages 203–267. New York: Academic, 1976.

[7] R. Szeliski. "Fast surface interpolation using hierarchical basis functions,". *IEEE Trans. Pattern Anal. Machine Intell.*, 12(6):513–528, 1990.

[8] D. Terzopoulos. "Image analysis using multigrid relaxation methods,". *IEEE Trans. Pattern Anal. Machine Intell.*, 8:129–139, 1986.

[9] D. Terzopoulos. "The computation of visible-surface representations,". *IEEE Trans. Pattern Anal. Machine Intell.*, 10(4):417–438, 1988.

[10] B.C. Vemuri, A. Mitiche, and J.K. Aggarwal. "Curvature-based representation of objects from range data,". *Image and Vision Comput.*, 4(2):107–114, 1986.