A Lattice-Preserving Multigrid Method for Solving the Inhomogeneous Poisson Equations Used in Image Analysis

Leo Grady

Siemens Corporate Research
Department of Imaging and Visualization
755 College Road East
Princeton, NJ 08540

Abstract. The inhomogeneous Poisson (Laplace) equation with internal Dirichlet boundary conditions has recently appeared in several applications ranging from image segmentation [1, 2, 3] to image colorization [4], digital photo matting [5, 6] and image filtering [7, 8]. In addition, the problem we address may also be considered as the generalized eigenvector problem associated with Normalized Cuts [9], the linearized anisotropic diffusion problem [10, 11, 8] solved with a backward Euler method, visual surface reconstruction with discontinuities [12, 13] or optical flow [14]. Although these approaches have demonstrated quality results, the computational burden of finding a solution requires an efficient solver. Design of an efficient multigrid solver is difficult for these problems due to unpredictable inhomogeneity in the equation coefficients and internal Dirichlet boundary conditions with unpredictable location and value. Previous approaches to multigrid solvers have typically employed either a data-driven operator (with fast convergence) or the maintenance of a lattice structure at coarse levels (with low memory overhead). In addition to memory efficiency, a lattice structure at coarse levels is also essential to taking advantage of the power of a GPU implementation [15,16,5,3]. In this work, we present a multigrid method that maintains the low memory overhead (and GPU suitability) associated with a regular lattice while benefiting from the fast convergence of a data-driven coarse operator.

1 Introduction

The inhomogeneous Poisson (Laplace) equation with internal Dirichlet boundary conditions has recently appeared in several applications ranging from image segmentation [1,2,3] to image colorization [4], digital photo matting [5,6] and image filtering [7]. In addition, the problem we address may also be considered as the generalized eigenvector problem associated with Normalized Cuts [9], the linearized anisotropic diffusion problem [10, 11, 8] solved with a backward Euler method, visual surface reconstruction with discontinuities [12, 13] or optical flow [14]. Although these approaches have demonstrated quality results, the computational burden of finding a solution on high-resolution 2D or 3D images demands an efficient solver that is not currently available in the literature.

D. Forsyth, P. Torr, and A. Zisserman (Eds.): ECCV 2008, Part II, LNCS 5303, pp. 252–264, 2008.

Although some of these algorithms are framed in a general graph setting, they have been almost exclusively employed in a rectilinear coordinate system that results in the use of a (widely) banded Laplacian matrix. Despite the banded structure of the matrix representing the discrete Laplacian operator on a grid, traditional fast Laplace/Poisson solvers are inappropriate due to the inhomogeneity of the diffusion constants (i.e., edge weights, in a graph context). These diffusion constants can vary by many orders of magnitude at neighboring pixels if, for example, a high-contrast image gradient is nearby. Unfortunately, the inhomogeneity of the diffusion constants is a very important feature of the behavior of anisotropic diffusion and related algorithms that cannot be eliminated for the sake of numerical efficiency. The solution of Poisson equations on domains with uniform diffusion constants may be performed quite efficiently (e.g., [17]).

Multigrid approaches are widely considered to be the best methods for solving a Poisson equation in a homogeneous domain, resulting in linear time algorithms [18, 19, 20]. Traditional multigrid approaches to solving the Poisson equation fall into two broad categories: Geometric and Algebraic. Classic geometric multigrid algorithms apply when the matrix represents a known grid, using bilinear interpolation to project the coarse-level solutions. Unfortunately, bilinear interpolation causes smoothing of coarse solutions across object boundaries (represented by small diffusion constants) in an image, causing a poor convergence rate. In contrast, algebraic multigrid [21, 22] uses the diffusion constants to generate problem-specific interpolation operators and coarsened matrices. Unfortunately, the coarsened matrices produced via algebraic multigrid are not guaranteed to represent a banded (lattice) sparsity structure. Note that some work has been addressed at finding matrix-dependent geometric multigrid methods (see [20] and references therein). For high-resolution images or 3D volumes in which each pixel corresponds to one equation, the banded structure of the Laplacian matrix offers significant storage and processing savings. Similarly, coarsening strategies have been explored in the past for graph-based algorithms (specifically, Markov Random Fields), but the explosion of non-local connectivity at high levels has made these approaches unworkable for high-resolution images/volumes (e.g., [23, 24]).

Both multiresolution methods (e.g., [25]) and multigrid numerical schemes have a long history in computer vision. Since at least as early as the work of Terzopoulos [26, 27], multigrid methods have been employed in computer vision in the context of visual surface reconstruction, shape-from-shading and optical flow. These early techniques employed the injection restriction operator and bilinear interpolation prolongation operator common to geometric multigrid methods. Although these operators are efficient and straightforward to implement, the convergence time for problems with strongly varying diffusion constants can be slower than necessary because bilinear interpolation smoothes the coarse-level solution over discontinuities (e.g., object boundaries in an image). Additionally, these restriction/prolongation operators are not adjoint to each other, which violates the Galerkin construction that is known theoretically to produce fast convergence [19]. Acton [28] applied similar multigrid operators to the anisotropic diffusion problem. Later work by Terzopoulos (and Szeliski) extended these ideas to handle discontinuities in surface reconstruction [12] and produce a parallel multigrid algorithm [13]. The technique for addressing discontinuities employed the concept of molecular inhibition across nodal discontinuities. Although similar in spirit

to the multigrid technique presented here, these nodal molecules did not guarantee a lattice structure at coarse levels.

The anisotropy of the optical flow problem has prompted investigations into data (matrix) driven multigrid operators. Ghosal and Vaněk [14] proposed using a smoothed aggregate restriction/prolongation operator inspired by the algebraic multigrid concept introduced by Ruge of *strongly coupled nodes* [29]. Although similar to the technique that we present here in the sense that a data-driven aggregation operator is proposed, an important difference between our work and [14] is that their aggregation technique will not preserve a lattice structure at coarse levels. Additionally, our method is parameter-free. Algebraic multigrid techniques have also been applied in computer vision by Kimmel and Yavneh [30], using a more traditional algebraic multigrid approach for designing prolongation/restriction operators. However, as above, our method has the important distinction from this work that we preserve the lattice structure at each coarse level and employ no parameters for the construction of the prolongation/restriction operators.

In this work, we introduce the Maximally Connected Neighbor coarsening operator which both uses data-driven operators to produce fast multigrid convergence and preserves the memory-efficient lattice structure at higher levels. In Section 2, we describe the Maximally Connected Neighbor multigrid method and show that the coarse-level operators maintain the interpretation as the (inhomogeneous) Laplacian matrix of a grid. In Section 3 we compare the performance of our multigrid method with preconditioned conjugate gradients and the multigrid method of [31]. We finalize conclusions in Section 4.

2 Method

The Poisson equation that we consider will be written using graph notation due to the graph formulation of many of the algorithms in question. Despite this graph formulation, it is important to recognize that the "graph formulation" on a lattice is equivalent to a finite differences discretization of the continuous Poisson equation on a square domain with Neumann boundary conditions on the borders of the image.

We begin by fixing our notation for a graph. A **graph** consists of a pair $G = (V, E)$ with **vertices (nodes)** $v \in V$ and **edges** $e \in E \subseteq V \times V$. An edge, e, spanning two vertices, v_i and v_j, is denoted by e_{ij}. A **weighted graph** assigns a value to each edge called a **weight**. The weight of an edge, e_{ij}, is denoted by $w(e_{ij})$ or w_{ij}. The **degree** of a vertex is $d_i = \sum w(e_{ij})$ for all edges e_{ij} incident on v_i. The following will also assume that our graph is connected and undirected (i.e., $w_{ij} = w_{ji}$). In the language of continuous PDEs, the edge weights may be identified with diffusion constants, which are assumed here to have arbitrary positive values, i.e., $w_{ij} > 0$. The graphs considered in this paper will be taken to represent a Cartesian lattice in which every pixel (voxel) is uniquely identified with a node and each node is connected via an edge to each of its neighboring nodes with either a 4-connected or 8-connected structure in 2D or a 6-connected or 26-connected structure in 3D.

2.1 Problem Formulation

In the greatest generality, we will consider finding a solution to the equation

$$(T + L)\, x = f, \tag{1}$$

in which x is the desired solution (assigning one value to each node), f is an arbitrary function (assigning one value to each node), T is a diagonal matrix with an arbitrary nonnegative function on the diagonal, $t = \mathrm{diag}\,(T)$, and L represents the Laplacian matrix defined as

$$L_{ij} = \begin{cases} d_i & \text{if } i = j, \\ -w_{ij} & \text{if } v_i \text{ and } v_j \text{ are adjacent nodes,} \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where L_{ij} is indexed by vertices v_i and v_j. Additionally, we would like to permit a set of internal Dirichlet boundary conditions at a set of nodes $V_B \subseteq V$ for which the solution of x_i, $v_i \in V_B$ is prescribed by function $x_i = h_i$ if $v_i \in V_B$. These internal boundary conditions occur frequently in the applications we consider such as segmentation [1, 2, 3], colorization [4], matting [5, 6] and filtering [7]. Note that (1) represents a discrete Poisson equation if $T = 0$ and a discrete Laplace equation if $f = 0$. In the following sections we will employ the notational convenience of $G = (T + L)$.

Although we motivate this work in terms of a linear system solver, note also that (1) may be viewed as applying to generalized eigenvector problems of the form

$$Lx = \lambda T x, \tag{3}$$

in which λ represents an eigenvalue and $f = 0$. Problems of the form in (3) have been studied extensively with respect to the Normalized Cuts algorithm [9].

In the various imaging applications mentioned above the edge weights (diffusion constants) are used to encode the image structure. This is a common feature of graph based algorithms for image analysis and several weighting functions are commonly used in the literature [9, 32, 33]. Although this paper is concerned with solving linear systems of the form in (1) with arbitrary positive weights, an example of a very common weighting function is the Gaussian weighting given by

$$w_{ij} = \exp\left(-\beta(g_i - g_j)^2\right), \tag{4}$$

where g_i indicates the image intensity at pixel i and β represents a free parameter.

2.2 Incorporation of the Dirichlet Boundary Conditions

Since we are primarily concerned with maintaining a lattice structure, the internal Dirichlet boundary conditions can present a problem. Therefore, our first procedure will be to enforce the boundary conditions via a diagonal matrix. Specifically, we replace our problem in (1), including internal Dirichlet boundary conditions at nodes V_B with the problem

$$\left((T + L) + \alpha \mathrm{diag}\,(b)\right) x = f + \alpha h, \tag{5}$$

in which b is an indicator vector denoting membership in the boundary set (i.e., $b_i = 1$ if $v_i \in V_B$ and $b_i = 0$ otherwise) and h represents the boundary values ($h_i = 0$ if $v_i \notin V_B$). As $\alpha \to \infty$ the boundary values are enforced exactly with this construction. Since this method of enforcing the boundary conditions simply results in a modified diagonal matrix, T, and modified right hand side, f, we will persist in using the notation $Gx = f$ to represent the problem (1) with the understanding that any internal Dirichlet boundary conditions are enforced.

Algorithm 1. Basic steps of a multigrid method

1. `multigrid(L,x,f)`
2. **while** not converged to tolerance **do**
3. Relax $G^0 x^0 = f^0$ with v iterations (relaxation e.g., via Gauss-Seidel)
4. $r^0 = f - G^0 x^0$ (compute the residual)
5. $r^1 = R r^0$ (restrict the residual)
6. $c = 0$ (initialize the correction to zero)
7. `multigrid_recursion(1,`c, r^1`)`
8. $x^k = x^k + Pc$ (update the solution with the prolongated correction)
9. **end while**

2.3 Multigrid Methods

Standard relaxation methods, such as Jacobi iteration or Gauss-Seidel relaxation, have been shown to perform well in correcting high-frequency error [18]. However, low-frequency error requires an excessive number of iterations to produce convergence with these methods. The multigrid method works by producing a correction of the current solution that is derived from a coarsened form of the system. The principle is that the coarse-grid updates correct low-frequency errors, while fine grid relaxations correct high-frequency errors. In order to produce the coarse grid correction, the multigrid method is applied recursively, forming what is known as a **V-cycle**, described by Algorithm 1. Denote the relaxation iterations parameter by v, the residual vector with r, the current (intermediate) solution at level k as x^k, the coarsened operator at level k as L^k, the restriction operator as R and the prolongation operator as P. Since our prolongation/restriction operators are linear, we view them as matrices. Starting with $k = 0$, the multigrid method is described in pseudocode in Algorithm 1 and the recursive function is given in Algorithm 2.

Given the above formalism for multigrid, the main issues that must be addressed in order to design a multigrid method are:

1. Determining the sampling structure of coarse nodes.
2. Specifying the restriction operator.
3. Specifying the prolongation operator.
4. Producing a coarsened operator.

The design of restriction, prolongation and coarse operator typically follows the Galerkin construction

$$G^{k+1} = R G^k P, \tag{6}$$

$$P = R^T, \tag{7}$$

which is known to have favorable theoretical properties [19]. Consequently, the specification of the restriction operator is sufficient to fully describe the prolongation and coarsened operators of a multigrid method.

The sampling structure of the coarse nodes used in the geometric multigrid has been to select every second node in each direction along the axes. Geometric

Algorithm 2. Recursive call of a multigrid method

1. `multigrid_recursion(`k, y, d`)`
2. **if** $k = k_{\mathrm{MAX}}$ **then**
3. Solve $G^k y = d$ exactly
4. **else**
5. Relax $G^k y^k = d^k$ with v iterations (relaxation)
6. $r^k = d^k - G^k y^k$ (compute the residual)
7. $r^{k+1} = R r^k$ (restrict the residual)
8. $c = 0$ (initialize the correction to zero)
9. `multigrid_recursion(`$k+1, c, r^{k+1}$`)` (continue recursion)
10. **end if**
11. $y^k = y^k + Pc$ (update the solution with the prolongated correction)

multigrid has a tremendous advantage for high-resolution images/volumes because the sampling structure and coarsened operator also represent a lattice and therefore the storage (and smoothing) at coarse levels has minimal indexing overhead. In addition to memory efficiency, a lattice structure at coarse levels is useful because it is possible to take advantage of the power of a GPU implementation [15, 16, 5, 3]. Unfortunately, matrix-independent geometric multigrid is inappropriate for inhomogeneous domains since the projection operators (bilinear interpolation) do not take the weights (diffusion constants) into account, permitting the smoothing of a coarse solution over areas of low diffusivity. The advantage of algebraic multigrid is that the weights are used to produce the prolongation/restriction and coarse operators, which does not smooth a coarse solution over areas of low diffusivity. Unfortunately, the algebraic multigrid procedure has a matrix-driven sampling structure that leads to coarse operators which have an arbitrary sparsity structure, forcing a high memory consumption and limiting possible application to high resolution images/volumes. Our approach in this paper is to suggest a new coarsening operator that adopts the lattice-preserving sampling structure of geometric multigrid while still using a matrix-driven (inhomogeneity-driven) restriction operator, chosen in such a way as to preserve a lattice structure at higher levels. Therefore, one may consider our method as a matrix-dependent geometric multigrid solver.

In the next section, we detail the proposed restriction operator and show that the resulting coarsened operators still maintain a lattice sparsity structure.

2.4 Maximally Connected Neighbor Coarsening

If R has only a single '1' entry on each row, then the coarse-level operator, RGR^T, represents the Laplacian (plus diagonal) of a coarse-level graph, where each node on the fine-level node is subsumed into a large node on coarse-level. Such a prolongation/restriction operator is termed an *agglomerative* operator [19] since it may be viewed as grouping together fine-level nodes into a single coarse-level node. The advantage of an agglomerative operator is that the coarse-level matrix is guaranteed to be the Laplacian of a graph, and therefore the approach is guaranteed to be stable. In contrast, the coarse-level operator RGR^T that uses weighted bilinear interpolation as the prolongation operator (as suggested by [31]) is not guaranteed to produce the Laplacian of some graph on the coarse level, and therefore exhibits instability.

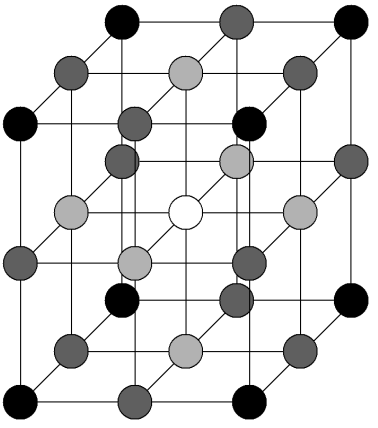

Fig. 1. Coarse-lattice neighbors in 3D. Black nodes correspond to coarse-lattice samples, dark-gray nodes to 1st order neighbors, light-gray nodes to 2nd order neighbors and white nodes to 3rd order neighbors.

In the context of the present applications, an agglomerative operator should exhibit two properties:

1. Coarse-level boundaries should correspond to fine-level boundaries (i.e., the agglomeration should represent a rough presegmentation).
2. The coarse-level graph topology should be regular (i.e., be a lattice), so that it may be stored/processed efficiently.

We now define the maximally connected neighbor (MCN) restriction operator to meet these objectives.

We will restrict our attention to 4- or 8- connected lattices in 2D, 6-, 10- or 26-connected lattices in 3D or, more generally, $\leq (3^p - 1)$-connected lattices in p-D. Without loss of generality, we will provide our definitions for a 3D lattice. We begin by first writing each node in terms of its 3D coordinates (starting with $(0, 0, 0)$ and moving in the positive directions). Define a node as belonging to the k-th order distance set, $\mathcal{N}^k \subset V$, if the number of odd digits in its coordinates are equal to k. Our coarse sampling set will consist of all nodes in $\mathcal{N}^0$. Note that, for p-D, $(\mathcal{N}^0 \cup \mathcal{N}^1 \cup \ldots \cup \mathcal{N}^p) = V$. The k-th order distance sets are represented pictorially in Figure 1. A node in the $\mathcal{N}^k$th distance set will be referred to as a ***k*th-order neighbor** of a coarse node.

We now proceed to define the maximally connected neighbor concept. Define the operator

$$\begin{aligned} M(v_i \in \mathcal{N}^k) &= v_j, \\ \text{s.t.} \quad v_j &\in \mathcal{N}^{k-1}, \\ \exists\, e_{ij} &\in E, \\ w_{ij} &\geq w_{ib},\ \forall\, v_b. \end{aligned} \tag{8}$$

In the case of a tie (in terms of weight) for maximal neighbors of a node, we define $M(v_i) = v_j$ if $(i - j)$ is minimum for all eligible nodes having equal, maximal weight

with v_i. We term v_j the **maximally connected neighbor** of v_i. This definition of $M(\cdot)$ may be applied recursively until a coarse node (a node in $\mathcal{N}^0$) is reached. Let $M^*(v_i)$ denote this recursive operator, i.e., $M^*(v_i) = M(M(\ldots M(v_i))) = v_j \in \mathcal{N}^0$.

Given the set of coarse nodes, $c_i \in \mathcal{N}^0$, we may therefore represent the $M(\cdot)$ operator in matrix form as:

$$R_{v_i c_j} = \begin{cases} 1 & \text{if } M^*(v_i) = c_j, \\ 0 & \text{else.} \end{cases} \tag{9}$$

2.5 Coarse Graphs

We first show that the coarse-level matrix, $RGR^T = R\,(L+T)\,R^T$, is in fact a matrix of the same form as G. Clearly, $RTR^T = T_c$ results in a T_c that is also diagonal with a nonnegative diagonal. Consequently, our concern is whether or not $L_c = RLR^T$ is also a coarse Laplacian matrix.

Any graph Laplacian matrix may be decomposed [34] into

$$L = A^T C A, \tag{10}$$

where C is an $m \times m$ diagonal matrix, indexed by edge, e_i, such that $C_{ii} = w_i$ and A is the $m \times n$ edge-node **incidence matrix** defined as

$$A_{e_{ij} v_k} = \begin{cases} +1 & \text{if } i = k, \\ -1 & \text{if } j = k, \\ 0 & \text{otherwise,} \end{cases} \tag{11}$$

for every vertex v_k and edge e_{ij}, where e_{ij} has been arbitrarily assigned an orientation. Note that neither P nor A contain information about the edge weights. Therefore, if $B = AP$ is an incidence matrix for some graph, then

$$L_c = RLR^T = RA^T C A R^T = B^T C B, \tag{12}$$

is also a graph Laplacian, which is necessarily isomorphic to a graph [34] (i.e., the matrix is a unique representation of a graph).

This result follows from the following lemma:

Lemma 1. *For $m \times n$ edge-node incidence matrix A and $n \times q$ matrix Q, the product $B = AQ$ is an edge-node incidence matrix if Q is a binary matrix (i.e., all values are zero or one) where each row has one and only one nonzero entry.*

Proof. A matrix is an incidence matrix if each row has no greater than one $\{-1, 1\}$ pair, with all other entries being zero.

Consider the ith row of A, a_i, which must contain either all zeros or a single $\{-1, 1\}$ pair. If all zeros, then the corresponding row of B, B_i is all zeros. If there is a $\{-1, 1\}$ pair, denote the nonzero entries by A_{ij} and A_{ik}, respectively. Since each row of Q contains a single '1', then B_i will either be all zeros (if $Q_{js} = Q_{ks} = 1$ for some column s) or will contain a $\{-1, 1\}$ pair (if $Q_{js} = Q_{kr} = 1$ for $s \neq r$). □

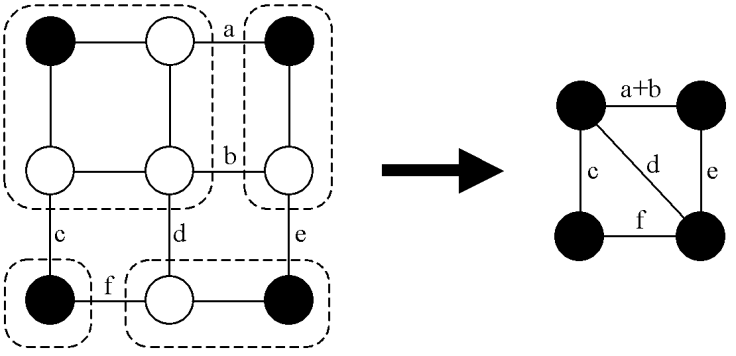

Fig. 2. Coarsening weights from a fine lattice to a coarse lattice. Black circles represent selected coarse nodes while open circles indicate fine-level nodes. Dashed lines show groups of fine-level nodes that have a common maximally connected neighbor. The graph on the right indicates the coarse-level weighting derived from the fine-level MCN groupings. Although a single diagonal edge has been introduced in the coarse-level graph, the coarse-level graphs will have connectivity no greater than an 8-connected lattice (or a 26-connected lattice in 3D). Weights on internal edges in the MCN grouping have no influence on the coarse-level weighting and are therefore unlabeled to avoid confusion.

We now proceed to show that the coarse level graph will maintain a lattice structure. Given the formulation of the coarse-level Laplacian matrix in (12), the weight between two coarse nodes on the upper-level graph is given by

$$w_{ij} = \sum_{M^*(v_s)=v_i, M^*(v_q)=v_j, \forall v_s, v_j} w_{sq}. \tag{13}$$

Therefore, the weight between neighboring coarse nodes is nonzero only if $\exists\, e_{ij} \in E$, s.t. $M^*(v_i) \neq M^*(v_j)$. A k-th order neighbor may be mapped to a set of 2^k coarse nodes through $M^*(v_i)$, which we may denote as the **reach** of v_i through M^*, $H(v_i)$. Given two neighboring nodes, v_i and v_j, of kth and $(k-1)$th order respectively, we note that $H(v_j) \subset H(v_i)$. Since $|H(\cdot)|$ is maximal for a node $v_i \in \mathcal{N}^k$, and contains all nodes in the hypercube surrounding v_i, only coarse-level nodes at the corners of this hypercube may become connected. Consequently, if a fine-level lattice is connected within a coarse-level hypercube, i.e., $(3^p - 1)$-connected in p-D, then the coarse-level lattice will remain $(3^p - 1)$-connected in p-D for every subsequent coarse level. Figure 2 illustrates an example of the MCN coarsening in which a fine-level four-connected 2D lattice is coarsened to an eight-connected 2D lattice.

2.6 Efficient Implementation

A description of an efficient method for effecting the three operators (restriction, coarse-level and prolongation) is given by:

1. Find M^* of each node in the fine-level graph by examining kth-level neighbors from $k = 0$ to $k = P$, where P represents the data dimension.

2. The restriction may be effected by summing together the solution at each fine-level node that maps through M^* to the same coarse-level node to produce the coarse-level solution.
3. Produce the weights for the coarse-level graph, L_c, from (13), i.e., for every two neighbors with different M^* nodes, add the weight between them to the weight between their M^* nodes.
4. Produce the diagonals for the coarse-level graph, T, by summing together the diagonal at each fine-level node that maps through M^* to the same coarse-level node.
5. The prolongation operator may be efficiently implemented by copying the solution at a coarse-level node to each of the fine-level nodes that were mapped by M^* to that coarse node.

Note that each of these steps require linear computational complexity. In the next section, we examine the speed effects of our method with respect to an optimized conjugate gradients and the pertinent multigrid method described in [31].

3 Results

The generalized inhomogeneous Poisson equation in (1) has found application for several problems, as reviewed above. The multigrid method will produce the same solution to the linear system, only faster. Consequently, it is sufficient to benchmark the method for speed on any model problem of the form in (1) if it is run to the same level of convergence as the competing numerical methods. The application chosen for the experiments was the segmentation method of [2]. Five high-resolution images were seeded and segmented using [2], followed by progressive downsampling of both the images and seeds to lower resolution. Our goal was to replicate the same segmentation problem at multiple resolutions.

We compared our multigrid approach for solving (1) to a conjugate gradients method and the multigrid method of [31] on 2D images of increasing size. The conjugate gradients method was heavily optimized and employed an incomplete Cholesky preconditioner that was modified to keep the dropped values along the diagonal (see [35] for more details). Both of the multigrid methods were run in a standard V-cycle with two Gauss-Seidel smoothing iterations for the multigrid method of [31] and one Gauss-Seidel smoothing iteration for the MCN multigrid. All three methods were run until the norm of the residual of the solution was within a tolerance value that was kept constant for all experiments. The computations used to provide a solution to (1) were all performed with double precision. Note that the conjugate gradients algorithm increased computation time with roughly $\mathcal{O}\left(n^{1.4}\right)$ complexity, despite the visualization which may make it appear to increase with a linear dependence on resolution. The MCN multigrid did in fact increase computation with a roughly linear dependence on resolution.

For each image at each resolution, all three algorithms were applied and the total time reported for execution on an Intel Xeon 2.4GHz machine with 3GB of RAM. The error bars represent one standard deviation of the runtime of the image set. While both multigrid approaches significantly outperform the conjugate gradients approach, the MCN multigrid is faster than the multigrid method of [31] by approximately five

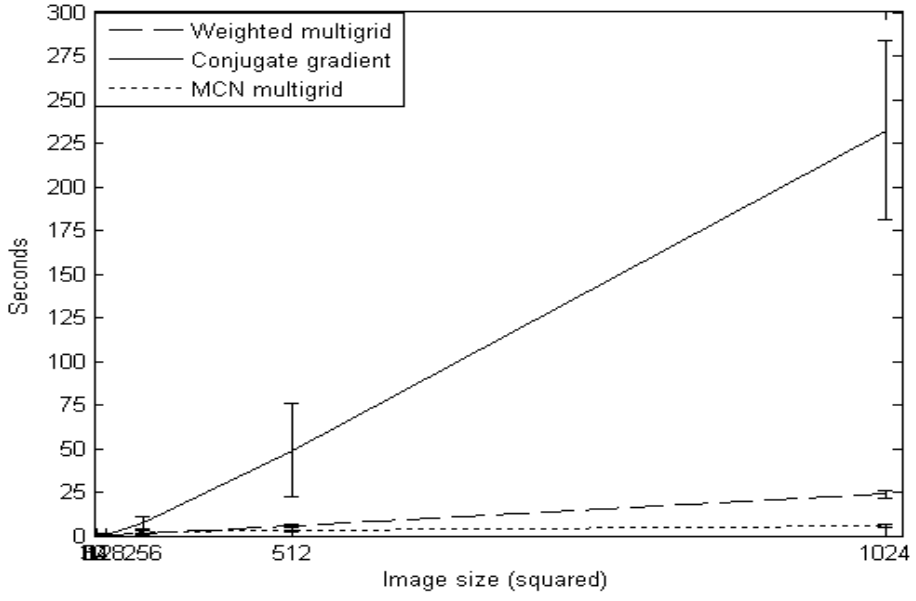

Fig. 3. Comparison of the conjugate gradient method (with incomplete Cholesky preconditioner) to the proposed multigrid method for solving (1) when segmenting images [2]. All images were square, with the length of one side given by the x-axis.

times. Due to the lattice structure of our MCN multigrid method, a GPU could easily be applied to produce additional speed gains [15].

4 Conclusion

We have presented a multigrid method for solving problems of the general form given in (1). Problems with this structure have become increasingly important in image processing applications such as segmentation [1, 2, 3], colorization [4], matting [5, 6] and filtering [7, 8]. Similar systems have also been studied in the context of visual reconstruction [12] and optical flow [14].

Previous approaches to multigrid solvers have typically employed either a matrix-dependent coarsening operator with fast convergence or the maintenance of a lattice structure at coarse levels with low memory overhead (see [20] for some exceptions). A coarse-level lattice structure is essential for memory demands (and fast processing) of high-resolution images or volumes. In addition to memory efficiency, a lattice structure at coarse levels is also essential to taking advantage of the power of a GPU implementation [15, 16, 5, 3]. However, matrix-dependent restriction/prolongation operators are essential for fast convergence of the multigrid method in the presence of discontinuities. The Maximally Connected Neighbor multigrid algorithm presented here both preserves a lattice structure at higher levels as well as employs the matrix-dependent restriction/prolongation operators that result in fast convergence. Our experiments compared

our method with optimized preconditioned conjugate gradients code and another Laplacian-specific multigrid method. Our method strongly outperforms the speed of both methods when applied to real images. Future work includes a GPU implementation of our fast inhomogeneous Poisson solver which is expected to produce even greater efficiency gains and the technique of recombining iterants [36] to further accelerate the convergence speed of the algorithm.

Acknowledgments

The author would like to thank Oren Livne for several useful discussions.

References

1. Grady, L., Schwartz, E.L.: Isoperimetric graph partitioning for image segmentation. IEEE Trans. on Pat. Anal. and Mach. Intel. 28(3), 469–475 (2006)
2. Grady, L.: Random walks for image segmentation. IEEE Trans. on Pat. Anal. and Mach. Intel., 1768–1783 (November 2006)
3. Bhusnurmath, A.: Applying Convex Minimization Techniques to Energy Minimization Problems in Computer Vision. Ph.D thesis, U. Pennsylvania (2008)
4. Levin, A., Lischinski, D., Weiss, Y.: Colorization using optimization. In: Proc. of SIGGRAPH, vol. 23, pp. 689–694. ACM, New York (2004)
5. Grady, L., Schiwietz, T., Aharon, S., Westermann, R.: Random walks for interactive alpha-matting. In: Proc. of VIIP, pp. 423–429. ACTA Press (September 2005)
6. Levin, A., Lischinski, D., Weiss, Y.: A closed form solution to natural image matting. In: Proc. of CVPR 2006, New York (June 2006)
7. Grady, L., Schwartz, E.: Anisotropic interpolation on graphs: The combinatorial Dirichlet problem. Technical Report CNS-TR-03-014, Boston University (2003)
8. Zhang, F., Hancock, E.R.: Graph spectral image smoothing. In: Escolano, F., Vento, M. (eds.) GbRPR. LNCS, vol. 4538, pp. 191–203. Springer, Heidelberg (2007)
9. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Trans. on Pat. Anal. and Mach. Intel., 888–905 (August 2000)
10. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE Trans. on Pat. Anal. and Mach. Intel. 12(7), 629–639 (1990)
11. Desbrun, M., Meyer, M., Schröder, P., Barr, A.H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In: Proc. of CGIT, pp. 317–324 (August 1999)
12. Terzopoulos, D.: The computation of visible-surface representations. IEEE Trans. on Pat. Anal. and Mach. Intel. 10(4), 417–438 (1988)
13. Szeliski, R., Terzopoulos, D.: Parallel multigrid algorithms and computer vision applications. In: Mandel, J., et al. (eds.) Proc. of the Fourth Copper Mountain Conf. on Multigrid Methods, pp. 383–398. SIAM, Philadelphia (1989)
14. Ghosal, S., Vaněk, P.: A fast scalable algorithm for discontinuous optical flow estimation. IEEE Trans. on Pat. Anal. and Mach. Intel. 18(2), 181–194 (1996)
15. Bolz, J., Farmer, I., Grinspun, E., Schröder, P.: Sparse matrix solvers on the GPU: Conjugate gradients and multigrid. ACM Trans. on Graphics, 917–924 (July 2003)
16. Krüger, J., Westermann, R.: Linear algebra operators for GPU implementation of numerical algorithms. ACM Trans. on Graphics, 908–916 (July 2003)
17. Sapiro, G.: Inpainting the colors. Technical Report IMA Preprint Series #1979, University of Minnesota, Institute for Mathematics and its Applications (2004)

18. Briggs, W.L., Hensen, V.E., McCormick, S.F.: A Multigrid Tutorial, 2nd edn. SIAM, Philadelphia (2000)
19. Trottenberg, U., Oosterlee, C.W., Schuller, A.: Multigrid. Academic Press, San Diego (2000)
20. Wesseling, P.: An Introduction to Multigrid Methods. R.T. Edwards (2004)
21. Dendy, J.E.: Black box multigrid. J. of Computational Physics 48, 366–386 (1982)
22. Brandt, A.: Algebraic multigrid theory: The symmetric case. Appl. Math. Comput. 19, 23–56 (1986)
23. Gidas, B.: A renormalization group approach to image processing problems. IEEE Trans. on Pat. Anal. and Mach. Intel. 11(2), 164–180 (1989)
24. Pérez, P., Heitz, F.: Restriction of a Markov random field on a graph and multiresolution statistical image modeling. IEEE Trans. on IT 42(1), 180–190 (1996)
25. Burt, P.J., Adelson, E.H.: The Laplacian pyramid as a compact image code. IEEE Trans. on Comm. COM-31(4), 532–540 (1983)
26. Terzopoulos, D.: Multilevel computational processes for visual surface reconstruction. Comput. Vision, Graphics, and Image Processing 24, 52–96 (1983)
27. Terzopoulos, D.: Image analysis using multigrid relaxation methods. IEEE Trans. on Pat. Anal. and Mach. Intel. 8(2), 129–139 (1986)
28. Acton, S.T.: Multigrid anisotropic diffusion. IEEE Trans. on Image Proc. 7(3), 280–291 (1998)
29. Ruge, J., Stüben, K.: Algebraic multigrid. In: McCormick, S. (ed.) Multigrid Methods. Frontiers in Applied Mathematics, vol. 3, pp. 73–130. SIAM, Philadelphia (1987)
30. Kimmel, R., Yavneh, I.: An algebraic multigrid approach for image analysis. SIAM J. of Sci. Comput. 24(4), 1218–1231 (2003)
31. Grady, L., Tasdizen, T.: A geometric multigrid approach to solving the 2D inhomogeneous Laplace equation with internal boundary conditions. In: Proc. of ICIP 2005, vol. 2, pp. 642–645. IEEE, Los Alamitos (2005)
32. Boykov, Y., Jolly, M.P.: Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In: Proc. of ICCV 2001, pp. 105–112 (2001)
33. Black, M.J., Sapiro, G., Marimont, D.H., Heeger, D.: Robust anisotropic diffusion. IEEE Trans. on Image Proc. 7(3), 421–432 (1998)
34. Biggs, N.: Algebraic Graph Theory. Cambridge Tracts in Mathematics, vol. 67. Cambridge University Press, Cambridge (1974)
35. Barrett, R., Berry, M., Chan, T.F., Demmel, J., Donato, J.M., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., Van der Vorst, H.: Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. SIAM, Philadelphia (1993)
36. Brandt, A., Mikulinksy, V.: On recombining iterants in multigrid algorithms and problems with small islands. SIAM J. of Sci. Comput. (1995)