Combinatorial Preconditioners and Multilevel Solvers for Problems in Computer Vision and Image Processing [*]

Ioannis Koutis
CSD-UPRRP
ioannis.koutis@upr.edu

Gary L. Miller
CSD-CMU
glmiller@cs.cmu.edu

David Tolliver
CSD-CMU
tolliver@cs.cmu.edu

Abstract

Several algorithms for problems including image segmentation, gradient inpainting and total variation are based on solving symmetric diagonally dominant (SDD) linear systems. These algorithms generally produce results of high quality. However, existing solvers are not always efficient, and in many cases they operate only on restricted topologies. The unavailability of reliably efficient solvers has arguably hindered the adoptability of approaches and algorithms based on SDD systems, especially in applications involving very large systems.

A central claim of this paper is that SDD-based approaches can now be considered practical and reliable. To support our claim we present Combinatorial Multigrid (CMG), the first reliably efficient SDD solver that tackles problems in general and arbitrary weighted topologies. The solver borrows the structure and operators of multigrid algorithms, but embeds into them powerful and algebraically sound combinatorial preconditioners, based on novel tools from support graph theory. In order to present the derivation of CMG, we review and exemplify key notions of support graph theory that can also guide the future development of specialized solvers. We validate our claims on very large systems derived from imaging applications. Finally, we outline two new reductions of non-linear filtering problems to SDD systems and review the integration of SDD systems into selected algorithms.

1 Introduction and Motivation

The Laplace operator has played a central role in computer vision for nearly 40 years. In his early work Horn employed finite element methods for elliptical operators in shape from shading [Hor70], to produce albedo maps [Hor74], and flow estimates [Hor81]. In his seminal work, Witkin [Wit83] studied the diffusion properties of matrix equations derived from the Laplace operator for linear filtering, later generalized by Perona and Malik [PM90] to the anisotropic case. More recently, Laplacians of combinatorial graphs have formed the algorithmic core of spectral methods [SM00, NJW02, BN03, YS04, CLL$^+$05, TM06, CS07], random walks segmentation [Gra06], in-painting [Sze06, MP08, BCCZ08], and matting methods [LRAL07]. Further research, such as the work by Grady et.al [GA08] on Mumford-Shah segmentation, aims to address traditional image processing problems via algorithms that solve a number of symmetric diagonally dominant (SDD) systems [1].

Given the pervasiveness of SDD systems in computer vision applications, the design of SDD solvers is an important endeavor. We argue that a good SDD solver targeting computer vision applications should have the following characteristics:

1. *Speed and Scalability.* Many applications require timely performance. Images such as medical scans already provide enormous volumes of data, while increases in resolution are expected.

[*]This work was partially supported by NSF CCF-0635257 and CCF-1018463.

[1]A symmetric matrix A is diagonally dominant when $A_{ii} \geq \sum_{j \neq i} |A_{ij}|$.

2. *Reliability.* The solver speed should not be overly instance-dependent. For example, a medical scan analyzer should be expected to work reasonably fast on *all* scans.

3. *Black-box quality.* The solver must not require any user interaction.

4. *Support of general sparse weighted topologies.* Many applications, such as the spectral segmentation and convex programming, generate systems with wildly varying weights and often employ randomly sampled or loosely localized topologies. Algorithms for optimization problems initially defined on regular lattices can also benefit in terms of speed, as they often can be localized to subgraphs of lattices.

With computer vision-generated linear systems pushing the limits of computational feasibility, researchers inevitably have relied on iterative algorithms, such as a class of solvers known as Algebraic Multigrid (AMG) [RS87, Bra00] and specialized solvers developed by researchers in computer vision [SBB01, Sze06, GO07, MP08, BCCZ08, GS08, Gra08]. AMG algorithms have been developed and fine-tuned targeting engineering applications, requiring from the user the experience to deal with a large space of algorithmic knobs [HY02]. On the other hand the vision-specializing solvers don't require advanced skills from the user, but they operate only on restricted topologies that limit their applicability. Most importantly, all known solvers are heuristic and –as a result– none of them is reliable [2]. While their empirical performance is frequently very good, that is not always the case.

Is it even possible to design an SDD solver that concentrates all the desired characteristics? In this work we describe the Combinatorial Multigrid (CMG) solver that provides an affirmative answer. As its name suggests, CMG borrows the structure and operations of multigrid algorithms. What differentiates CMG from other multigrid solvers is its setup phase which is based on a sound algebraic machinery. This machinery is provided by *support theory*, a set of techniques developed for the construction of *combinatorial preconditioners.*

The rest of the paper is organized as follows. In Section 2 we give background material on certain useful fragments of support theory. The purpose of Section 2 is not only to explain the derivation of CMG but also stimulate further research by providing a lens through which the strengths and weaknesses of other solvers can be viewed, understood, and improved. We illustrate this through the discussion of Section 2.4. In Section 3 we give some background material on solvers and present CMG. The theoretical foundation of CMG has been laid in previous work [KM08], but the solver itself and its application to computer vision systems are new. In Section 4, after discussing a methodology for picking test SDD systems, we present experiments that compare CMG to other publicly available solvers. The experiments highlight the reliability of CMG and demonstrate its power as a software primitive.

Finally in Section 5 we provide timing and complexity bounds for selected computer vision methods that require the solutions to SDD systems at their core. Among else, we outline how non-linear filtering operations such as ℓ_2, ℓ_1 Total Variation [ROF92, CS05] and Non-Local Means [BCM08, BKC08] can also be formulated as optimizations with these linear systems at their core. To the best of our knowledge these reductions are novel.

2 Support Theory for graphs

2.1 Preconditioners - Motivating Support Theory

In this Section we review fragments of support theory that are relevant to the design of our SDD solver. We refer the reader to [BH03] for an extensive exposition of support theory. Iterative algorithms, such as the Chebyshev iteration or the Conjugate Gradient, converge to a solution using only matrix-vector products with A. It is well known that iterative algorithms suffer from slow convergence properties when

[2]An exception are solvers for regular unweighted lattices [MP08].

the conditioning of A, $\kappa(A)$, - defined as the ratio of the largest over the minimum eigenvalue of A - is large [Axe94].

Preconditioned iterative methods attempt to remedy the problem by changing the linear system to $B^{-1}Ax = B^{-1}b$. In this case, the algorithms use matrix-vector products with A, and solve linear systems of the form $By = z$. The speed of convergence now depends on the **condition number** $\kappa(A, B)$, defined as

$$\kappa(A, B) = \max_x \frac{x^T Ax}{x^T Bx} \cdot \max_x \frac{x^T Bx}{x^T Ax} \tag{2.1}$$

where x is taken to be outside the null space of A. In constructing a preconditioner B, one has to deal with two contradictory goals: (i) Linear systems in B must be easier than those in A to solve, (ii) The condition number must be small to minimize the number of iterations.

Historically, preconditioners were natural parts of the matrix A. For example, if B is taken as the diagonal of A we get the Jacobi Iteration, and when B is the upper triangular part of A, we get the Gauss-Seidel iteration.

The cornerstone of combinatorial preconditioners is the following intuitive yet paradigm-shifting idea explicitly proposed by Vaidya [Vai91]: *A preconditioner for the Laplacian of a graph A should be the Laplacian of a simpler graph B, derived in a principled fashion from A.*

2.2 Graphs as electric networks - Support basics

There is a fairly well known analogy between graph Laplacians and resistive networks [DS00]. If G is seen as an electrical network with the resistance between nodes i and j being $1/w_{i,j}$, then in the equation $Av = i$, if v is the vector of voltages at the node, i is the vector of currents. Also, the quadratic form $v^T Av = \sum_{i,j} w_{i,j}(v_i - v_j)^2$ expresses the *power dissipation* on G, given the node voltages v. In view of this, the construction of a good preconditioner B amounts to the construction of a simpler resistive network (for example by deleting some resistances) with an energy profile close to that of A.

The **support** of A by B, defined as $\sigma(A/B) = \max_v v^T Av/v^T Bv$ is the number of copies of B that are needed to support the power dissipation in A, for all settings of voltages. The principal reason behind the introduction of the notion of support, is to express its local nature, captured by the Splitting Lemma.

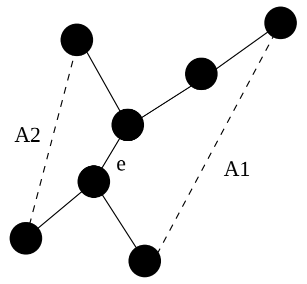
Lemma 2.1 (Splitting Lemma) *If $A = \sum_{i=1}^m A_i$ and $B = \sum_{i=1}^m B_i$, where A_i, B_i are Laplacians, then $\sigma(A, B) \leq max_i \sigma(A_i, B_i)$.*

The Splitting Lemma allows us to bound the support of A by B, by splitting the power dissipation in A into small local pieces, and "supporting" them by also local pieces in B.

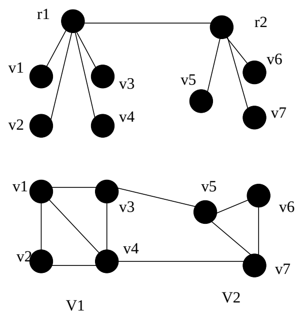
For example, in his work Vaidya proposed to take B as the maximal weight spanning tree of A. Then, it is easy to show that $\sigma(B, A) \leq 1$, intuitively because more resistances always dissipate more power. In order to bound $\sigma(A, B)$, the basic idea to let the A_i be edges on A (the ones not existing in B), and let B_i be the unique path in the tree that connects the two end-points of A_i. Then one can bound separately each $\sigma(A_i, B_i)$. In fact, it can be shown that any edge in A that doesn't exist in B, can be supported *only* by the path B_i.

As a toy example, consider the example in Figure 1(a) of the two (dashed) edges A_1, A_2 and their two paths in the spanning tree (solid) that share one edge e.

In this example, the **dilation** of the mapping is equal to 3, i.e. the length of the longest of two paths. Also, as e is uses two times, we say that the **congestion** of the mapping is equal to 2. A core Lemma in Support Theory [BGH$^+$05, BH03] is that the support can be upper bounded by the product **congestion$*$dilation**.



(a) A graph and its spanning tree - obtained by deleting the dashed edges.



(b) A graph and its Steiner preconditioner

Figure 1:

2.3 Steiner preconditioners

Steiner preconditioners, introduced in [Gre96] and extended in [KM08] introduce external nodes into preconditioners. The proposed preconditioner is based on a partitioning of the n vertices in V into m vertex-disjoint clusters V_i. For each V_i, the preconditioner contains a star graph S_i with leaves corresponding to the vertices in V_i rooted at a vertex r_i. The roots r_i are connected and form the **quotient** graph Q. This general setting is illustrated in Figure 1(b).

Let D' be the total degree of the leaves in the Steiner preconditioner S. Let the **restriction** R be an $n \times m$ matrix, where $R(i,j) = 1$ if vertex i is in cluster j and 0 otherwise. Then, the Laplacian of S has $n + m$ vertices, and the algebraic form

$$S = \begin{pmatrix} D' & -D'R \\ -R^T D' & Q + R^T D' R \end{pmatrix}. \tag{2.2}$$

A worrisome feature of the Steiner preconditioner S is the extra number of dimensions/vertices. So how do we even use it? Gremban and Miller [Gre96] proposed that every time a system of the form $Bz = y$ is solved in an usual preconditioned method, the system

$$S \begin{pmatrix} z \\ z' \end{pmatrix} = \begin{pmatrix} y \\ 0 \end{pmatrix}$$

should be solved instead, for a set of *don't care* variables z'. They also showed that the operation is equivalent to preconditioning with the dense matrix

$$B = D' - V(Q + D_Q)^{-1} V^T \tag{2.3}$$

where $V = D'R$, and $D_Q = R^T D' R$. The matrix B is called the Schur complement of S with respect to the elimination of the roots r_i. It is a well known fact that B is also a Laplacian.

The analysis of the support $\sigma(A/S)$, is identical to that for the case of subgraph preconditioners. For example, going back to Figure 1(b), the edge (v_1, v_4) can only be supported by the path (v_1, r_1, v_4), and the edge (v_4, v_7) only by the path (v_4, r_1, r_2, v_7). Similarly we can see the mappings from edges in A to paths in S for every edge in A. In the example, the **dilation** of the mapping is 3, and it can be seen that to minimize the **congestion** on every edge of S (i.e. make it equal to 1), we need to take $D' = D$, where D are the total degrees of the nodes in A, and $w(r_1, r_2) = w(v_3, v_5) + w(v_4, v_7)$. More generally, for two roots r_i, r_j we should have

$$w(r_i, r_j) = \sum_{i' \in V_i, j' \in V_j} w_{i,j}.$$

Under this construction, the algebraic form of the quotient Q can be seen to be $Q = R^T AR$.

So far no special properties of the clustering have been used. Those come into play in bounding the support of S by A, $\sigma(S/A)$. In [KM08] it was shown that the support $\sigma(S/A)$ reduces to bounding the support $\sigma(S_i, A[V_i])$, for all i, where $A[V_i]$ denotes the graph induced in A by the vertices V_i. How can we bound $\sigma(S_i, A[V_i])$? Before we answer this question, let us recall the definition of **conductance**.

Definition 2.2 *The conductance $\phi(A)$ of a graph $A = (V, E, w)$ is defined as*

$$\phi(A) = \min_{S \subseteq V} \frac{w(S, V - S)}{\min(w(S), w(V - S))}$$

where $w(S, V - S)$ denotes the total weight connecting the sets S and $V - S$, and where $w(S)$ denotes the total weight incident to the vertices in S.

The main result of [KM08] is captured by the following Theorem.

Theorem 2.3 *The support $\sigma(S/A)$ is bounded by a constant c independent from n, if and only if for all i the conductance of the graph $A^o[V_i]$ induced by the nodes in V_i augmented by the edges leaving V_i is bounded by a constant c'.*

2.4 Support Theory for predicting the performance of solvers

Theorem 2.3 doesn't give a way to pick clusters, but it does provide a way to *avoid* bad clusterings. In recent work [Gra08], Grady proposed a multigrid method where the construction of the "coarse" grid follows exactly the construction of the **quotient** graph in the previous section. Specifically, Grady's algorithm proposes a clustering such that every cluster contains exactly one pre-specified 'coarse' nodes. It then defines the restriction matrix R and he lets the coarse grid be $Q = R^T AR$, identically to the construction of the previous Section. The algorithm is iterated to construct a hierarchy of grids. The question then is whether the proposed clustering provides the guarantees that by Theorem 2.3 are necessary for the construction of a good Steiner preconditioner. In the following Figure, we replicate Figure 2 of [Gra08], with a choice of weights that force the depicted clustering.

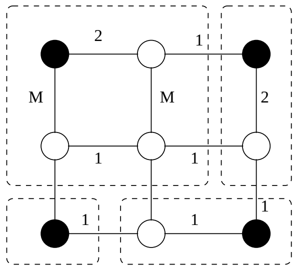


Figure 2: A bad clustering

Every cluster in Figure 2 contains exactly one black/coarse node. The problem with the clustering is that the top left cluster, has a very low conductance when $M >> 1$. In general, in order to satisfy the requirement of Theorem 2.3, there are cases where the clustering has to contain clusters with *no* coarse nodes in them. As we will discuss in Section 3.4 the behavior of the multigrid algorithm proposed in [Gra08] is closely related to the quality of the Steiner preconditioner induced by the clustering. This implies that the multigrid of [Gra08] can suffer bad convergence.

The canonical clustering in Grady's algorithm is very suitable for GPU implementations, when other solvers may be less suitable. This gives to it an advantage on this type of hardware. Even in the presence of a number of relatively bad clusters, it can be faster relative to a solver that uses better clusters. However the advantage is lost when the computed clusters cross a negative threshold in quality, a threshold that depends on several hardware-dependent factors. The value of Support Theory is evident in this case. Grady's algorithm can be instrumented with a very fast routine that measures the quality of the formed clusters and predicts its performance, and reverts to another solver when needed. One can also imagine hybrid clustering algorithms where the majority clusters are formed using the algorithm [Gra08] and the 'sensitive' parts of the system are treated separately.

3 The Combinatorial Multigrid Solver

In this section we describe the Combinatorial Multigrid Solver (CMG). We start with a short review of multigrid algorithms and other SDD solvers, which is necessary to explain the differences of CMG from previous multigrid algorithms.

3.1 Related work on SDD solvers

Multigrid was originally conceived as a method to solve linear systems that are generated by the discretization of the Laplace (Poisson) equation over relatively nice domains [TSO00]. The underlying geometry of the domain leads to a hierarchy of grids $A = A_0, \ldots, A_d$ that look similar at different levels of detail; the picture that the word multigrid often invokes to mind is that of a tower of 2D grids, with sizes $2^{d-i} \times 2^{d-i}$ for $i = 0, \ldots, d$. Its provably asymptotically optimal behavior for certain classes of problems soon lead to an effort -known as *Algebraic Multigrid* (AMG)- to generalize its principles to arbitrary matrices. In contrast to classical Geometric Multigrid (GMG) where the hierarchy of grids is generated by the discretization process, AMG constructs the hierarchy of 'coarse' grids/matrices based only on the algebraic information contained in the matrix. Various flavors of AMG –based on different heuristic coarsening strategies– have been proposed in the literature. AMG has been proven successful in solving more problems than GMG, though some times at the expense of robustness, a by-product of the limited theoretical understanding.

A solver with provable properties for arbitrary SDD matrices, perhaps the 'holy grail' of the multigrid community, was discovered only recently. The path to it was Support Theory [BH03], a set of mathematical tools developed for the study of combinatorial subgraph preconditioners, originally introduced by Vaidya [Vai91, Jos97]. It has been at the heart of the seminal work of Spielman and Teng [ST06] who proved that SDD systems can be solved in nearly-linear time. Koutis and and Miller [KM07] proved that SDD matrices with planar connection topologies (*e.g.* 4-connectivity in the image plane) can be solved asymptotically optimally, in $O(n)$ time for n-dimensional matrices. The complexity of the Spielman and Teng solver was recently significantly improved by Koutis, Miller and Peng [KMP10, KMP11], who described an $O(m \log n)$ algorithm for the solution of general SDD systems with m non-zero entries.

It is fair to say that these theoretically described solvers are still impractical due to the large hidden constants, and the complicated nature of the underlying algorithms. Combinatorial Multigrid (CMG) [KM09] is a variant of multigrid that reconciles theory with practice. Similarly to AMG, CMG builds a hierarchy of matrices/graphs. The essential difference from AMG is that the hierarchy is constructed by viewing the matrix as a graph, and using the *discrete geometry* of the graph, for example notions like graph separators and expansion. It is, in a way, a hybrid of GMG and AMG, or a discrete-geometric MG. The re-introduction of geometry into the problem allows us to prove sufficient and necessary conditions for the construction of a good hierarchy and claim strong convergence guarantees for symmetric diagonally dominant (SDD) matrices based on recent progress in Steiner preconditioning [Gre96, Kou07, KM08].

3.2 SDD linear systems as graphs

In this Section we discuss how SDD linear systems can be viewed entirely as graphs. Combinatorial preconditioning advocates a principled approach to the solution of linear systems. The core of CMG and all other solvers designed in the context of combinatorial preconditioning is in fact a solver for a special class of matrices, graph Laplacians. The *Laplacian* A of a graph $G = (V, E, w)$ with *positive* weights, is defined by:

$$A_{i,j} = A_{j,i} = -w_{i,j} \text{ and } A_{i,i} = -\sum_{i \neq j} A_{i,j}.$$

More general systems are solved via light-weight transformations to Laplacians. Consider for example the case where the matrix A has a number of positive off-diagonal entries, and the property $A_{i,i} = \sum_{i \neq j} |A_{i,j}|$. Positive off-diagonal entries have been a source of confusion for AMG solvers, and various heuristics have been proposed. Instead, CMG uses a reduction known as double-cover [Gre96]. Let $A = A_p + A_n + D$, where D is the diagonal of A and A_p is the matrix consisting only of the positive off-diagonal entries of A. It is easy to verify that

$$Ax = b \Leftrightarrow \begin{pmatrix} D + A_n & -A_p \\ -A_p & D + A_n \end{pmatrix} \begin{pmatrix} x \\ -x \end{pmatrix} = \begin{pmatrix} b \\ -b \end{pmatrix}.$$

In this way, we reduce the original system to a Laplacian system, while at most doubling the size. In practice it is possible to exploit the obvious symmetries of the new system, to solve it with an even smaller space and time overhead.

Matrices of the form $A + D_e$, where A is a Laplacian and D_e is a positive diagonal matrix have also been addressed in various ways by different AMG implementations. In CMG, we again reduce the system to a Laplacian. If d_e is the vector of the diagonal elements of D, we have

$$Ax = b \Leftrightarrow \begin{pmatrix} A + D_e & 0 & -d_e \\ 0 & A + D_e & -d_e \\ -d_e^T & -d_e^T & \sum_i d_e(i) \end{pmatrix} \begin{pmatrix} x \\ -x \\ 0 \end{pmatrix} = \begin{pmatrix} b \\ -b \\ 0 \end{pmatrix}.$$

Again it's possible to implement the reduction in a way that exploits the symmetry of the new system, and with a small space and time overhead work only implicitly with the new system.

A symmetric matrix A is diagonally dominant (SDD), if $A_{i,i} \geq \sum_{i \neq j} |A_{i,j}|$. The two reductions above can reduce any SDD linear system to a Laplacian system. Symmetric positive definite matrices with non-positive off-diagonals are known as M-matrices. It is well known that if A is an M-matrix, there is a positive diagonal matrix D such that $A = DLD$ where L is a Laplacian. Assuming D is known, an M-system can also be reduced to a Laplacian system via a simple change of variables. In many application D is given, or it can be recovered with some additional work [SD08].

The reduction of SDD systems to Laplacians allows us to concentrate on them for the rest of the paper. There is a one-to-one correspondence between Laplacians and graphs, so we will be often using the terms interchangeably.

3.3 A graph decomposition algorithm

According to the discussion of §2.3, the crucial step for the construction of a good Steiner preconditioner is the computation of a group decomposition that satisfies, as best as possible, the requirements of Theorem 2.3. Before the presentation of the **Decompose-Graph** algorithm, that extends the ideas of [KM08], we need to introduce a couple of definitions. Let $vol_G(v)$ denote the total weight incident to node v in graph

G. The *weighted degree* of a vertex v is defined as the ratio

$$wd(v) = \frac{vol(v)}{\max_{u \in N(v)} w(u,v)}.$$

The *average weighted degree* of the graph is defined as

$$awd(G) = (1/n) \sum_{v \in V} wd(v).$$

```
Algorithm **Decompose-Graph**
Input: Graph $A = (V, E, w)$
Output: Disjoint Clusters $V_i$ with $V = \bigcup_i V_i$
1. Let $\kappa > 4$ be a constant and $W \subseteq V$ be the set of nodes satisfying
       $wd(v) > \kappa \cdot awd(A)$
2. Form $F \subset G$ by keeping the heaviest incident edge for each $v \in V$
       $F$ is a forest of trees
3. For every vertex $w \in W$ such that
       $vol_T(w) < vol_G(w)/awd(A)$ :
   Remove from $F$ the edge contributed by $w$ in Step 2.
4. Decompose each tree $T$ in $F$ into vertex-disjoint trees, trying to optimize the maximum
   conductance over the vertex-disjoint trees.
```

It is not very difficult to prove that the algorithm **Decompose-Graph** produces a partitioning where the conductance of each cluster depends only on $awd(A)$ and the constant κ. In fairly general sparse topologies that allow high degree nodes, $awd(A)$ is constant and the number of clusters m returned by the algorithm is such that $n/m > 2$ (and in practice larger than 3 or 4). There are many easy ways to implement Step 3. Our current implementation makes about three passes of A. Of course, one can imagine variations of the algorithm (i.e. a correction step, etc) that may make the clustering phase a little more expensive with the goal of getting a better conductance and an improved condition number, if the application at hand requires many iterations of the solver.

3.4 From Steiner preconditioners to Multigrid

In this subsection we outline the intuition behind the fact that Steiner preconditioners and multigrid. Details and proofs can be found in [Kou07]. Algebraically, any of the classic preconditioned iterative methods, such as the Jacobi and Gauss-Seidel iteration, is nothing but a matrix $\mathcal{S}$, which gets applied implicitly to the current error vector e, to produce a new error vector $e' = \mathcal{S}e$. For example, in the Jacobi iteration we have $\mathcal{S} = (I - D^{-1}A)$. This has the effect that it reduces effectively only part of the error in a given iterate, namely the components that lie in the low eigenspaces of $\mathcal{S}$ (usually referred to as high frequencies of A). The main idea behind a two-level multigrid is that the current *smooth* residual error $r = b - Ax$, can be used to calculate a correction $R^T Q^{-1} Rr$, where Q is a smaller graph and R is an $m \times n$ restriction operator. The correction is then added to the iterate x. The hope here is that for smooth residuals, the low-rank matrix $R^TQ^{-1}R$ is a good approximation of A^{-1}. Algebraically, this correction is the application of the operator $T = (I - R^TQ^{-1}RA)$ to the error vector e. The choice of Q is most often not independent from that of R, as the *Galerkin condition* is employed:

$$Q = RAR^T.$$

The Galerkin condition ensures that T is a projection operator with respect to the A-inner product. Two-level convergence proofs are then based on bounds on the angle between the subspace $Null(P)$ and the high frequency subspace of $\mathcal{S}$.

At a high level, the key idea behind CMG is that the provably small condition number $\kappa(A, B)$ where B is given in expression 2.3, is equal to the condition number $\kappa(\hat{A}, \hat{B})$ where $\hat{A} = D^{-1/2}AD^{-1/2}$ and $\hat{B} = D^{-1/2}BD^{-1/2}$. This in turn implies a bound on the angle between the low frequency of $\hat{A}$ and the high frequency of $\hat{B}$ [KM08]. The latter subspace is $Null(R^TD^{1/2})$. This fact suggests to choose $R^TD^{1/2}$ as the projection operator while performing relaxation with $(I - \hat{A})$ on the system $\hat{A}y = D^{-1/2}b$, with $y = D^{1/2}x$. Combining everything, we get the following two-level algorithm.

```
**Two-level Combinatorial Multigrid**

Input: Laplacian $A = (V, E, w)$, vector $b$, approximate solution $x$, $n \times m$ restriction matrix $R$
Output: Updated solution $x$ for $Ax = b$

1. $D := diag(A)$;  $\hat{A} := D^{-1/2}AD^{-1/2}$;
2. $z := (I - \hat{A})D^{1/2}x + D^{-1/2}b$;
3. $r := D^{-1/2}b - \hat{A}z$;  $w := RD^{1/2}r$;
4. $Q := RAR^T$;  Solve $Qy = w$;
5. $z := z + D^{1/2}R^Ty$
6. $x := D^{-1/2}((I - \hat{A})z + D^{-1/2}b)$
```

The two-level algorithm can naturally be extended into a full multigrid algorithm, by recursively calling the algorithm when the solution to the system with Q is requested. This produces a hierarchy of graphs $A = A_0, \ldots, A_d$. The full multigrid algorithm we use, after simplifications in the algebra of the two-level scheme is as follows.

```
**function** $x := CMG(A_i, b_i)$
1. $D := diag(A)$
2. $x := D^{-1}b$
3. $r_i := b_i - A_i(D^{-1}b)$
4. $b_{i+1} := Rr_i$
5. $z := CMG(A_{i+1}, b_{i+1})$
6. **for** $i = 1$ **to** $t_i - 1$
7.        $r_{i+1} := b_{i+1} - A_{i+1}z$
8.        $z := z + CMG(A_{i+1}, r_{i+1})$
9. **endfor**
10. $x := x + R^Tz$
11. $x := r_i - D^{-1}(A_ix - b)$
```

If $nnz(A)$ denotes the number of non-zero entries in matrix A, we pick

$$t_i = \max\{\lceil \frac{nnz(A_i)}{nnz(A_{i+1})} - 1 \rceil, 1\}.$$

This choice for the number of recursive calls, combined with the fast geometric decrease of the matrix sizes, targets a geometric decrease in the total work per level, while optimizing the condition number.

4 Experiments with CMG

The quality of SDD-based approaches and algorithms for computer vision problems has been demonstrated in the several papers that we discuss in the introduction and, in more detail, in the next Section. The purpose of this Section is to lend experimental support to our claim that SDD-based algorithms should also be considered extremely practical and reliable, suitable for very large volumes of data in commercial applications.

4.1 Selecting SDD systems for testing

The task of evaluating and comparing SDD solvers for computer vision problems can perhaps be approached via the compilation of a large set of SDD systems arising in various computer visions and algorithms. However, compiling such a set requires a daunting amount of work. This is not only because the number of SDD-based algorithms is large, but also because certain algorithms iterate on systems and generate more than one qualitatively different systems.

We argue that testing a solver on a large set of systems is probably wasteful too, especially if we want to verify the speed and reliability of the solver. In such a case we follow a 'selective' approach that uses mathematical intuition and previous experimental experience to select a number of instances that are hard.

In this work we follow the selective approach. To construct the test set, we set forth a set of requirements:

1. The test systems/graphs must be very large. Large sizes can reveal a bad underlying asymptotic behavior. In addition, heuristic mistakes are more probable in large graphs.

2. The very rich topological properties of 3D graphs relative to those of 2D graphs also increase the probability of heuristic mistakes. So a number of 3D systems must be included in the test set.

3. The test set must contain graphs with a very large variation of weights and steep discontinuities in local and long-range scales, as both experience and theory support the intuition that unweighted graphs are relatively easier.

4. The test set must contain graphs with non-regular weighted topologies that are relevant to computer vision applications.

4.2 Our test set of SDD systems

Previous related works (e.g. [Sze06, Gra08]) evaluate solvers on a very small number of systems coming from natural images. In this paper we present more extensive experiments on affinity graphs/systems such as those proposed in [SM00]. Most of our experiments are with 3-dimensional images; up to our knowledge no previous experimental work has considered 3-dimensional systems.

We exclude natural images from our experiments, because our experience indicates that they are relatively easier. Our experience seems to match that of other research groups [SK11]. For example, the solver of [Sze06] works very well on natural images, but its performance degrades significantly on images derived from the application of the random walker method on CT scans [Gra06].

In our tests we use systems derived from the application of the spectral rounding algorithm [TM06] on EM microscopy images of the neural system of a sea specimen, provided to us by Eduardo Molinar's Biological Imaging Group at the University of Puerto Rico-Rio Piedras. The images are two-dimensional cross-sectional scans of three-dimensional objects. The 2D images can be stacked into 3D images with no volume registration. The images contain interesting 'non-natural' structures of different scales including local noise, often uncorrelated between neighboring 2D frames. These features satisfy the third requirement in the list of the previous subsection. One image similar to those used in our experiments is shown in Figure 3.

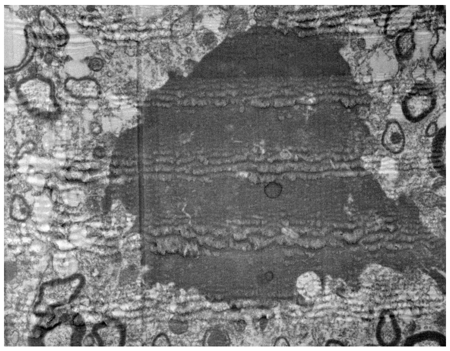


Figure 3: An EM microscopy scan. Notice the presence of noise.

We experiment with 3D images consisting of different numbers of frames, including 2D images. This creates a range of different graphs, with up to 16 million nodes, that allow us to observe the speed and scalability of the solvers but also how the performance is affected by the transition from a 2D frame to a –topologically richer– 3D images. In prior work [TKI$^+$08], we found useful to add to the graph one high-degree node that joins all the nodes on one face of the 3D lattice. We've repeated this approach for the EM microscopy scans on the largest examples. We include these graphs as examples of systems with a non-regular topology.

4.3 Experiments

To the best of our knowledge the only published variant of Algebraic Multigrid running in MATLAB is AGMG [MN08, Not10]. We also compare with the classical Ruge-Stüben AMG [RS87], and Smoothed Aggregation Multigrid [VBM01], implemented as part of the package PyAMG [BOS11], written in a combination of Python and C++ for efficiency.

All solvers in our experiments consist of a setup and a solve phase. We do not report the setup times because they consist only a negligible fraction of the total time to solve the system. The AGMG and Smoothed Aggregation Multigrid multigrid solvers *failed* to converge (in a reasonable number of iterations) on *all* our test systems. So, our report includes only CMG and the Ruge-Stüben variant of AMG.

The experiments with the solve phases of CMG and RS-AMG are reported in Figure 4. The systems $Ax = b$ were solved for a randomly picked b-side, and the *stopping criterion* for convergence was taken to be $||Ax - b|| < 1e - 05 * ||b||$. The `'na'` symbol means that the corresponding number is not available. This is not a problem with the AMG solver but it is 'due to technical difficulties in loading the MATLAB-generated .mat files containing the systems into PyAMG; the corresponding routine appeared to stagnate or quit for the larger sizes, at least in our systems.

The leftmost column gives the dimensions of the lattice. The two other columns give the number of iterations until convergence for CMG and RS-AMG respectively. The *times per iteration* for CMG and RS-AMG are within 5% of each other, without a clear winner. Thus the number of iterations is strongly correlated with the actual running times. The CMG solver is 3-4 faster than the RS-AMG solver on all systems with the exception of the lattice including a high-degree node. This case highlights the heuristic and inconsistent nature of RS-AMG. On the other hand it is clear that the performance of CMG is consistent across sizes and topologies.

Matrix	**CMG_iter**	**AMG_iter**
1024^2x1	25	77
1024^2x2	23	65
1024^2x4	20	82
1024^2x4+1	18	15
1024^2x8	23	na
1024^2x16	23	na
512^2x1	22	36
512^2x2	20	65
512^2x4	21	75
512^2x8	23	67
512^2x16	21	70
512^2x32	23	na
512^2x64	23	na

Figure 4: Experiments with CMG and RS-AMG.

5 SDDs arising in Computer Vision

Many computer vision problems naturally suggest a graph structure - for example the vertices often correspond to samples (*e.g.* pixels, patches, images), the edge set establishes pairwise comparisons or constraints encoded in the graph and the weights are either data driven (for clustering) or the result of an ongoing optimization procedure (weights in the t^{th} iteration of Newton's method).

The reformulation of fundamental objective functions in computer vision, such as the recent reduction of the Mumford-Shah functional in [GA08], to optimizations on combinatorial graphs opens the door to faster and more accurate algorithms. In this section we illustrate the pervasiveness and utility of SDDs with a collection of related problems that reduce to solutions to SDDs in the inner loop. For the two Total Variation applications we demonstrate that the systems reduce to SDDs at their core - for the others this fact is obvious.

5.1 Gradient Inpainting

Recent work on gradient inpainting [Sze06, MP08, BCCZ08] has centered around the development of specialized solvers for 4-connected meshes (either weighted or un-weighted). The gradient inpainting problem seeks to integrate out an image from a (potentially sparse) set of image gradients. Formulated as a least squares optimization, given a vector of gradient constraints Δ^c for each channel c, the corresponding set of relationships encoded in the edge-node incidence matrix Γ and a possible weighting of the constraints W - the least squares image is obtained by solving:

$$\Gamma^T W \Gamma x^c = \Gamma^T W^{-1/2} \Delta^c \tag{5.4}$$

for each color channel. As $\Gamma^T W \Gamma$ is a weighted Laplacian the solution can be computed in $\tilde{O}(n)$ work for the general case and $O(n)$ work the weighted planar case (as addressed in [Sze06, MP08, BCCZ08]) - the image reconstruction step requires less work than linear filtering.

The gradient inpainting problem is evocative of Total Variation (TV) approaches for image processing problems. In the next section we outline optimization of related denoising functionals which can be used to condition least squares inpainting.

5.2 Non-linear Filtering and Convex Optimization

Non-linear methods in image processing based on Total Variation[3] (TV) arise in image denoising [ROF92], super resolution [PETM09], and inpainting [PBC08] in computer vision. In this section we demonstrate that ℓ_2, ℓ_2 and anisotropic (*i.e.* Manhattan) ℓ_2, ℓ_1 TV-like functionals reduce to solving SDD systems and can thus be integrated into CV pipelines at a cost comparable to applying a non-separable filterbank.

We begin with the discrete (anisotropic) form of the Rudin, Osher, Fatemi TV functional [ROF92], for $p-$norms along the ℓ_1 to ℓ_2 continuum:

$$\underset{x}{min} \ : \ ||x - s||_2^2 + \lambda \, ||\nabla x||_p \tag{5.5}$$

where $\ell_p \mid 2 \geq p \geq 1$ maintains the structural property of convexity, the existence of efficient algorithms, and generalizes to weighted norms. We outline a method for the $p = 1$ case as work on the statistics of natural images [RB94] and the success of sparse representations motivate the use of ℓ_1 for penalizing the image gradients. Write the primal for ℓ_2, ℓ_1 as:

$$\min_x \frac{1}{2}(x - s)^T(x - s) + 1^T|\Gamma x| \tag{5.6}$$

where Γ is the node-edge incidence matrix, $1^T|\Gamma x|$ measures change across edges in ℓ_1 and $(x - s)^T(x - s)$ measures the deviation from the source s in ℓ_2. Problem 5.6 can be formulated as follows:

$$\begin{array}{ll} \min_{x,t} & 1^T t + 1/2(x - s)^T(x - s) \\ & \left. t \geq \Gamma x \qquad\qquad\qquad \right\} y^+ \\ & \left. t \geq -\Gamma x \qquad\qquad\quad\;\; \right\} y^- \end{array}$$

By introducing the Lagrangian variables y^+, y^-, we can write the dual as:

$$\min_{x,t} \max_{y^+ \geq 0, y^- \geq 0} \quad 1^T t + 1/2(x - s)^T(x - s) + {y^+}^T(-t - \Gamma x) + {y^-}^T(-t + \Gamma x)$$

Taking derivative with respect to x yields

$$x - s - \Gamma^T(y^+ - y^-) = 0$$

Taking derivative with respect to t yields

$$y^+ + y^- = 1$$

Let $y = y^+ - y^-$. Since $y^+, y^- \geq 0$, we have $|y| \leq 1 \iff y^+ + y^- = 1$ (to see this let $y^+ = \frac{y+1}{2}$ and $y^- = \frac{-y+1}{2}$.) Plugging this back into the original formulation yields

$$\xi(x) = \max_{|y| \leq 1} -\frac{1}{2} y^T \Gamma \Gamma^T y + y^T \Gamma s. \tag{5.7}$$

It is well known that interior point methods can be applied to the above constrained problem by creating an unconstrained function $\hat{\xi}$ that replaces the linear constraint $|y| \leq 1$ with a log-barrier term[4], $\lambda \log(1 - y) + \lambda \log(1 + y)$. The interior point optimization now amounts to Newton's method[Boy04] on $\hat{\xi}(x)$. To uncover the computational complexity of the procedure we begin by calculating the gradient and hessian

[3] See [CS05] for a thorough survey of TV and related mathematical image processing methods.

[4] Recall: As $\lambda \to 0$, we have $\lambda \log(x) = \begin{cases} 0, & x > 0 \\ -\infty, & x \leq 0 \end{cases}$, here we assume $\log(x) = -\infty$ if $x \leq 0$.

of $\hat{\xi}(x)$ and examine their structure:

$$\frac{\partial \hat{\xi}}{\partial y} = -\Gamma\Gamma^T y + \Gamma s + \lambda \frac{1}{y-1} + \lambda \frac{1}{y+1} \tag{5.8}$$

$$\frac{\partial^2 \hat{\xi}}{\partial y^2} = -\Gamma\Gamma^T - \lambda \mathtt{d}(y-1)^{-2} - \lambda \mathtt{d}(y+1)^{-2} \tag{5.9}$$

where $\mathtt{d}(x)$ promotes the vector x to a diagonal matrix. Recall for Newton's method that the computational bottleneck is solution to linear system $Q^t y^{(t+1)} = y^t$, where the hessian, Q^t at iteration t is given by eq. 5.9.
Hence the algorithm is:

Algorithm 1: Solving TV-regularized smoothing problem.

```
Input: $\Gamma, s, \beta, \epsilon$
$t \leftarrow 0$;
$y^{(t)} \leftarrow 0$;
$\lambda \leftarrow 1$;
while $\lambda > \epsilon$ do
    repeat
        $Q^{(t)} \leftarrow \Gamma\Gamma^T + \lambda \mathtt{diag}(y^{(t)} - 1)^{-2} + \lambda \mathtt{diag}(y^{(t)} + 1)^{-2}$;
        $\Delta y^{(t)} \leftarrow {Q^{(t)}}^{-1} y^{(t)}$;
        $\alpha \leftarrow 1$;
        repeat
            $y^{(t+1)} \leftarrow y^{(t)} + \alpha \Delta y^{(t)}$;
            $\alpha \leftarrow \alpha\beta$;
        until $f(y^{(t+1)}, \lambda) \leq f(y^{(t)}, \lambda)$ ;
    until ${y^{(t)}}^T \Delta y^{(t)} \geq \epsilon$ ;
    $t \leftarrow t + 1$;
    $\lambda \leftarrow \lambda \frac{1}{1+1/\sqrt{n}}$;
end
```

Observe that the hessian matrix is factored as $Q^{(t)} = (\Gamma\Gamma^T + C)$ where C is a diagonal matrix. Using Binomial inverse theorem, we have ${Q^{(t)}}^{-1} = C^{-1} - C^{-1}(I + \Gamma^T C^{-1}\Gamma)^{-1} C^{-1}$, where I is the identity matrix. Since C is a diagonal matrix, C^{-1} is easy to compute. Further note that $\Gamma^T C^{-1} \Gamma$ is a weighted Laplacian matrix, so $I + \Gamma^T C^{-1} \Gamma$ is SDD. Given that Newton's method takes $O(\sqrt{n})$ many iterations [Boy04], the total running time is at most $\tilde{O}(n^{3/2})$ for the optimization of ℓ_2, ℓ_1.

5.2.1 Non-Local Means

Motivated by Efros' texture work [EL99], the non-local means (NLM) [BCM08] energy functional has received a great deal of recent attention [BKC08, PBC08, PETM09] due to its empirical performance in textured regions. We observe that the discrete instance of the functional can be written in terms of solving a weighted SDD. The NLM energy functional takes the following form:

$$\underset{x}{min} \ : \ ||x - s||_2^2 + \lambda \sum_{(ij) \in N} ||p_i - p_j||_W^2 \tag{5.10}$$

where s is the observed signal, x the de-noised signal, p_i is the image patch centered at location i, and W and an empirical weighting function over the patch pairs (ij).

The second term $\sum_{(ij)\in N} ||p_i - p_j||_W^2$ in Eq 5.10 can be written out as a symmetric quadratic form. Where R_i is the k^2 by n restriction matrix, as in §2.3, that selects the pixels from patch i and places them in a vector. The non-local term can now be written as

$$\begin{aligned}\sum_{(ij)\in N} w_{ij}\left(R_i x - R_j x\right)^T\left(R_i x - R_j x\right) &= x^T\left(\sum_{(ij)\in N} w_{ij}(R_i - R_j)^T(R_i - R_j)\right)x \\ &= x^T M x.\end{aligned}$$

The matrix M is clearly positive semi-definite as it is the sum of products, further, we see that it is a weighted Laplacian matrix as $R_i - R_j$ terms are exactly edge-node incidence matrices over the patch pixels.

To solve the NLM problem replace the patch term $\lambda\sum_{(ij)\in N} ||p_i - p_j||_W^2$, in eq 5.10, with $\lambda x^T M x$ and take the derivative with respect to x, setting to zero we arrive at linear system $(I + \lambda M)x = s$. The matrix $(I + \lambda M)$ is SDD as it the sum of SDD matrices, therefore x can be found in $\tilde{O}(n)$ work. Note that the above is a single step due to the ℓ_2 smoothness term, ℓ_1 penalization of the patch smoothness is easily achieved by adapting the duality algorithm to NLM.

5.3 Clustering, Maps, Matting and Segmentation

In recent years data clustering, embedding, image matting and image segmentation problems have been formulated as optimizations on combinatorial graphs representing the data [SM00, NJW02, BN03, YS04, CLL+05, LRAL07, Gra06, CS07]. Laplacians, normalized Laplacians and related linear operators of graphs arise naturally in formulating the objective functions and optimization procedures. In this section we briefly relate a handful of recent approaches that ultimately reduce to solving SDD linear systems.

As an example, the resistive network analogy (see §2.2) motivated an assisted segmentation method for images and volumes. Grady *et al.* [Gra06, SG07] exploited the relationship between graph Laplacians and random walks to segment images given sparse labels. This method requires a single solve and a sort to achieve its solution (a cost is dominated the vertex sort in the planar case).

Similar work [YS04, CS07, EOK07] extends the NCuts and related objectives, discussed below, to include membership constraints yielding assisted clustering procedures. These approaches differ from the Random Walks procedure, and Linear Programming[5] based $k-$way min-cut approaches in that the relative sizes of the partitions are explicitly balanced.

5.3.1 Eigencalculations in Vision

Calculating a minimal, say $k-$dimensional, eigenspace forms the computational core of the spectral relaxation for NCuts [SM00], spectral clustering [NJW02], Laplacian eigenmaps [BN03], diffusion maps [CLL+05], and the typical case for Levin *et. al.*'s image matting algorithm [LRAL07].

Recall that any symmetric matrix has n distinct pairs (λ_i, x_i) such that $Ax_i = \lambda_i x_i$. Pairs of symmetric positive definite matrices, (A, B) also have n distinct pairs (λ_i, x_i) such that $Ax_i = \lambda_i x_i$. The case where A is a Laplacian and $B = D$ is its diagonal is of great interest. In this case the generalized problem $Ax = \lambda Dx$ can be reduced to the simple eigenvalue problem for the normalized Laplacian $\mathcal{A} = D^{-1/2}AD^{-1/2}$, which has the same eigenvalues as A, and eigenvectors $y = D^{1/2}x$.

Iterative algorithms -such the power method or Lanczos' method- for the computation of approximate subspaces of $\mathcal{A}$, are based on matrix-vector multiplications with $\mathcal{A}$. For example, if x is a random vector orthogonal to the null space of $\mathcal{A}$, the vector $(I - \mathcal{A}/2)^k x$ converges to the eigenvector of $\mathcal{A}$ corresponding to its smallest non-trivial eigenvalue. For most interesting cases, iterative methods need a very large number of iterations with $\mathcal{A}$ to converge. However, it is easy to show (see for example [ST06]) that only $O(\log n \log(1/\epsilon))$ iterations of the power method with $\mathcal{A}^{-1}$ are required for the computation of a

[5]It is worth noting that the best theoretical upper bound for computing min-cut\max-flow can be obtained by solving SDD systems [CKM+10].

vector x such that $x^T \mathcal{A} x$ is within a $(1+\epsilon)$ factor from λ_2, when the exact eigenvector x_2 for λ_2 satisfies $x_2^T A x_2 = \lambda_2$. An approximation of this kind is sufficient for most applications. The discussion about the second eigen-pair extends to the first few eigenpairs.

Thus the complexity of finding a few (approximate) eigenvectors of $\mathcal{A}$ is $O(m \log^2 n)$ using the solver of [KMP10]. In practice using the CMG solver which runs in linear time, we achieve approximate NCuts solutions in time roughly proportional to sorting the vertices (pixels) by value.

6 Acknowledgment

We would like to thank Eduardo Rosa-Molinar and his Biological Imaging Group at the University of Puerto Rico-Rio Piedras that provided us the serial block-face imaging dataset used in this paper. Work on the dataset was supported by MH-086994, NSF-1039620, and NSF-0964114.

References

[Axe94] Owe Axelsson. *Iterative Solution Methods*. Cambridge University Press, New York, NY, 1994. 2.1

[BCCZ08] P. Bhat, B. Curless, M. Cohen, and C. L. Zitnick. Fourier analysis of the 2D screened Poisson equation for gradient domain problems. In *ECCV*, 2008. 1, 1, 5.1, 5.1

[BCM08] A. Buades, B. Coll, and J.M Morel. Nonlocal image and movie denoising. *IJCV*, 76(2):123–139, 2008. 1, 5.2.1

[BGH+05] Marshall Bern, John R. Gilbert, Bruce Hendrickson, Nhat Nguyen, and Sivan Toledo. Support-graph preconditioners. *SIAM J. Matrix Anal. Appl.*, 27:930–951, 2005. 2.2

[BH03] Erik G. Boman and Bruce Hendrickson. Support theory for preconditioning. *SIAM J. Matrix Anal. Appl.*, 25(3):694–717, 2003. 2.1, 2.2, 3.1

[BKC08] T. Brox, O. Kleinschmidt, and D. Cremers. Efficient nonlocal means for denoising of textural patterns. *Trans. on Image Processing*, 2008. 1, 5.2.1

[BN03] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003. 1, 5.3, 5.3.1

[BOS11] W. N. Bell, L. N. Olson, and J. B. Schroder. PyAMG: Algebraic multigrid solvers in Python v2.0, 2011. Release 2.0. 4.3

[Boy04] K. Boyd. *Convex Optimization*. Cambridge University Press, 1st edition, 2004. 5.2, 5.2

[Bra00] A. Brandt. General highly accurate algebraic coarsening. *Electronic Transactions on Numerical Analysis*, 2000. 1

[CKM+10] Paul Christiano, Jonathan A. Kelner, Aleksander Madry, Daniel A. Spielman, and Shang-Hua Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. *CoRR*, abs/1010.2921, 2010. 5

[CLL+05] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker. Diffusion maps geometric diffusions as a tool for harmonic analysis and structure definition of data. *PNAS*, 102(21):7426–7431, May 2005. 1, 5.3, 5.3.1

[CS05] Tony Chan and Jianhong Shen. *Image Processing And Analysis: Variational, Pde, Wavelet, And Stochastic Methods*. SIAM, 2005. 1, 3

[CS07] Timothee Cour and Jianbo Shi. Solving markov random fields with spectral relaxation. *AISTATS*, 2007. 1, 5.3

[DS00] Peter G. Doyle and J. Laurie Snell. Random walks and electric networks, 2000. 2.2

[EL99] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. *ICCV*, 2:1033–1038, 1999. 5.2.1

[EOK07] Anders P. Eriksson, Carl Olsson, and Fredrik Kahl. Normalized cuts revisited: A reformulation for segmentation with linear grouping constraints. In *ICCV*, pages 1–8. IEEE, 2007. 5.3

[GA08] L. Grady and C. Alvino. Reformulating and optimizing the Mumford-Shah functional on a graph - A faster, lower energy solution. *ECCV*, 5302:248–261, 2008. 1, 5

[GO07] G. Gilboa and S. Osher. Nonlocal operators with applications to image processing. *Multiscale Modeling and Simulation*, July 2007. 1

[Gra06] Leo Grady. Random walks for image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2(11):1768–1783, 2006. 1, 4.2, 5.3

[Gra08] L. Grady. A lattice-preserving multigrid method for solving the inhomogeneous poisson equations used in image analysis. *ECCV*, 5303:252–264, 2008. 1, 2.4, 2.4, 4.2

[Gre96] Keith Gremban. *Combinatorial Preconditioners for Sparse, Symmetric, Diagonally Dominant Linear Systems*. PhD thesis, Carnegie Mellon University, Pittsburgh, October 1996. CMU CS Tech Report CMU-CS-96-123. 2.3, 2.3, 3.1, 3.2

[GS08] Leo Grady and Ali Kemal Sinop. Fast approximate random walker segmentation using eigenvector precomputation. In *CVPR*. IEEE Computer Society, IEEE, June 2008. 1

[Hor70] B.K.P. Horn. Shape from shading: A method for obtaining the shape of a smooth opaque object from one view. Technical Report 232, MIT AI Laboratory, November 1970. 1

[Hor74] B.K.P. Horn. Determining lightness from an image. *Computer Graphics and Image Processing*, 3(1):277–299, 1974. 1

[Hor81] B.K.P. Horn. Determining optical flow. *MIT AI Laboratory*, 17(1):185–203, 1981. 1

[HY02] Van Emden Henson and Ulrike Meier Yang. BoomerAMG: A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics: Transactions of IMACS*, 41(1):155–177, 2002. 1

[Jos97] Anil Joshi. *Topics in Optimization and Sparse Linear Systems*. PhD thesis, University of Illinois at Urbana Champaing, 1997. 3.1

[KM07] Ioannis Koutis and Gary L. Miller. A linear work, $O(n^{1/6})$ time, parallel algorithm for solving planar Laplacians. In *Proc. 18th ACM-SIAM Symposium on Discrete Algorithms (SODA 2007)*, 2007. 3.1

[KM08] Ioannis Koutis and Gary L. Miller. Graph partitioning into isolated, high conductance clusters: Theory, computation and applications to preconditioning. In *Symposiun on Parallel Algorithms and Architectures (SPAA)*, 2008. 1, 2.3, 2.3, 2.3, 3.1, 3.3, 3.4

[KM09] Ioannis Koutis and Gary Miller. The combinatorial multigrid solver. Conference Talk, March 2009. 3.1

[KMP10] Ioannis Koutis, Gary L. Miller, and Richard Peng. Approaching optimality for solving SDD systems. In *FOCS '10: Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science*. IEEE Computer Society, 2010. 3.1, 5.3.1

[KMP11] Ioannis Koutis, Gary L. Miller, and Richard Peng. Solving sdd linear systems in time $\tilde{O}(m \log n \log(1/\epsilon))$. *CoRR*, abs/1102.4842, 2011. 3.1

[Kou07] Ioannis Koutis. *Combinatorial and algebraic algorithms for optimal multilevel algorithms*. PhD thesis, Carnegie Mellon University, Pittsburgh, May 2007. CMU CS Tech Report CMU-CS-07-131. 3.1, 3.4

[LRAL07] Anat Levin, A. Rav-Acha, and D. Lischinski. Spectral matting. In *CVPR*, 2007. 1, 5.3, 5.3.1

[MN08] Adrian C. Muresan and Yvan Notay. Analysis of aggregation-based multigrid. *SIAM J. Scientific Computing*, 30(2):1082–1103, 2008. 4.3

[MP08] James McCann and Nancy S. Pollard. Real-time gradient-domain painting. *SIGGRAPH*, 27(3), 2008. 1, 1, 2, 5.1, 5.1

[NJW02] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2002. 1, 5.3, 5.3.1

[Not10] Yvan Notay. An aggregation-based algebraic multigrid method. *Electronig Transactions of Numerical Analysis*, 2010. To appear. 4.3

[PBC08] G. Peyre, S. Bougleux, and L. Cohen. Non-local regularization of inverse problems. In *ECCV*, volume 5304, pages 57–68, 2008. 5.2, 5.2.1

[PETM09] M. Protter, M. Elad, H. Takeda, and P. Milanfar. Generalizing the nonlocal-means to super-resolution reconstruction. *Trans. on Image Processing*, 18(1):36–51, 2009. 5.2, 5.2.1

[PM90] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *PAMI*, 7(12):629–639, 1990. 1

[RB94] D. L. Ruderman and W. Bialek. Statistics of natural images: scaling in the woods. *NIPS*, 1994. 5.2

[ROF92] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithm. *Physica D*, 1(60):259–268, 1992. 1, 5.2

[RS87] J. W. Ruge and K. Stüben. Algebraic multigrid (AMG). In S. F. McCormick, editor, *Multigrid Methods*, volume 3 of *Frontiers in Applied Mathematics*, pages 73–130. SIAM, Philadelphia, PA, 1987. 1, 4.3

[SBB01] E. Sharon, A. Brandt, and R. Basri. Segmentation and boundary detection using multiscale intensity measurements. In *CVPR*. IEEE Computer Society, IEEE, 2001. 1

[SD08] Daniel A. Spielman and Samuel I. Daitch. Faster approximate lossy generalized flow via interior point algorithms. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, May 2008. 3.2

[SG07] Ali Kemal Sinop and Leo Grady. A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In *ICCV*. IEEE Computer Society, IEEE, Oct. 2007. 5.3

[SK11] Rick Szelisky and Dilip Krishnan, 2011. Private communication. 4.2

[SM00] J. Shi and J. Malik. Normalized cuts and image segmentation. In *PAMI*, 2000. 1, 4.2, 5.3, 5.3.1

[ST06] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems, 2006. 3.1, 5.3.1

[Sze06] Richard Szeliski. Locally adapted hierarchical basis preconditioning. *SIGGRAPH*, 25(3):1135–1143, August 2006. 1, 1, 4.2, 5.1, 5.1

[TKI$^+$08] D. A. Tolliver, I. Koutis, H. Ishikawa, J. S. Schuman, and G. L. Miller. Automatic multiple retinal layer segmentation in spectral domain oct scans via spectral rounding. In *ARVO Annual Meeting*, May 2008. 4.2

[TM06] David Tolliver and Gary L. Miller. Graph partitioning by spectral rounding: Applications in image segmentation and clustering. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006)*, pages 1053–1060, 2006. 1, 4.2

[TSO00] Ulrich Trottenberg, Anton Schuller, and Cornelis Oosterlee. *Multigrid*. Academic Press, 1st edition, 2000. 3.1

[Vai91] Preadeep M. Vaidya. Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners. A talk based on this manuscript was presented at the IMA Workshop on Graph Theory and Sparse Matrix Computation, October 1991. 2.1, 3.1

[VBM01] Petr Vanek, Marian Brezina, and Jan Mandel. Convergence of algebraic multigrid based on smoothed aggregation. *Numerische Mathematik*, 88(3):559–579, 2001. 4.3

[Wit83] A.P. Witkin. Scale-space filtering. *IJCAI*, pages 1019–1022, August 1983. 1

[YS04] S. Yu and J. Shi. Segmentation given partial grouping constraints. *PAMI*, 26(2):173–183, 2004. 1, 5.3