Fast Approximate Random Walker Segmentation Using Eigenvector Precomputation

Leo Grady
Siemens Corporate Research
Princeton, NJ USA

Ali Kemal Sinop
Carnegie Mellon University
Pittsburg, PA USA

Abstract

Interactive segmentation is often performed on images that have been stored on disk (e.g., a medical image server) for some time prior to user interaction. We propose to use this time to perform an offline precomputation of the segmentation prior to user interaction that significantly decreases the amount of user time necessary to produce a segmentation. Knowing how to effectively precompute the segmentation prior to user interaction is difficult, since a user may choose to guide the segmentation algorithm to segment any object (or multiple objects) in the image. Consequently, precomputation performed prior to user interaction must be performed without any knowledge of the user interaction. Specifically, we show that one may precompute several eigenvectors of the weighted Laplacian matrix of a graph and use this information to produce a linear-time approximation of the Random Walker segmentation algorithm, even without knowing where the foreground/background seeds will be placed. Finally, we also show that this procedure may be interpreted as a seeded (interactive) Normalized Cuts algorithm.

1. Introduction

Interactive methods for image segmentation have been gaining popularity in recent years because they permit the targeted extraction of objects of interest with minimal guidance. The method of user interaction has traditionally taken one of two forms: 1) Providing a complete initial boundary near the desired boundary that is evolved to the correct segmentation, 2) Selecting pieces of the desired boundary which are then connected automatically. The first approach to user interaction is typically employed by active contour or level set methods, while the second approach to user interaction has been employed by intelligent scissors/live wire [14, 7] and fast marching [6]. Since both of these approaches to user interaction require explicit formulations of the object boundary, it can sometimes be difficult to extend them to higher dimension or more abstract problems like data clustering. After the advent of the Graph Cuts algorithm [2], a "scribble" interface has recently become popular in which a user marks some pixels (voxels) as belonging to the object foreground and others as belonging to the object background (known as **seeds**). Using this partial labeling (which is much less than the number of pixels), the remaining pixels are accordingly labeled. Since the user interaction involves labeling a few data points, these methods extend without modification to higher dimensions and more abstract clustering scenarios. Following the success of the Graph Cuts segmentation method, several other algorithms have been developed that employ the same user interface, but differ in the procedure for generating a complete pixel labeling from the seeds [9, 15]. Despite the popularity of these methods for image segmentation, the size of high-resolution images or medical volumes is such that the runtime can be prohibitive for employing these methods in an interactive fashion.

In this paper we propose to shift the computational burden of an interactive algorithm to an *offline* procedure that may be performed before any user interaction has taken place. An offline procedure is attractive since, in many cases, there is substantial time between the acquisition of the image and the segmentation of that image by a user. In particular, medical images (volumes) often exist for days or weeks on a data server before a user interacts with the image. Precomputation via an offline procedure is difficult to formulate since it is unknown where the user will choose to place seeds. In Section 2, we show that precomputing a small number of eigenvectors of the graph Laplacian matrix is sufficient to allow for a good approximation of the solution to the interactive Random Walker image segmentation algorithm [9], regardless of where the seeds are placed. Given this precomputation of a few eigenvectors of the graph Laplacian matrix, the interactive, online segmentation is performed in linear time.

It is generally agreed that offline computer time is less valuable than the time that a user spends during interaction. For example, in a hospital setting, offline precomputation

978-1-4244-2243-2/08/$25.00 ©2008 IEEE

may occur over long periods of time while image data is stored on a data server. Additionally, several applications (e.g., radiation therapy planning) require the online segmentation of many objects in the same volume. However, the hourly rate of physicians and technicians is so high that any time that can be saved during their usage of software to segment or otherwise interact with image data results in substantial cost savings. Despite the opportunity for efficiency gains, the precomputation of quantities that enhance the speed of user interaction is a relatively unexplored topic in computer vision. In contrast, precomputation has been utilized with great effectiveness in the context of querying a shortest path on a static network (e.g., a road network) [10, 13]. To date, these algorithms utilizing precomputation produce the fastest shortest path queries of any known approach.

In computer vision, traditional approaches to increasing the efficiency of interactive image segmentation algorithms have focused primarily on multiresolution approaches rather than performing an offline computation. For example, several multiresolution methods have been proposed to increase the efficiency of Graph Cuts [11, 19] that can reduce the computation time to near linear complexity. However, these approaches have three primary difficulties: 1) Using a very low resolution results in poor segmentations, while using a finer resolution may still be computationally expensive (e.g., the lowest resolution $(64 \times 64 \times 64)$ used by [11] still required roughly 10 seconds to produce a segmentation), 2) Thin objects disappear at lower resolution, 3) Special handling is required to prevent nearby seeds from being merged into the same region at lower resolution. In contrast, our approach to increase the efficiency of an interactive segmentation method via precomputation yields an interactive algorithm that operates on the full resolution with a linear time complexity.

The key computation in the Normalized Cuts algorithm [18] is to produce eigenvectors of the normalized graph Laplacian matrix. Since we are using eigenvectors of the graph Laplacian matrix to approximate the Random Walker solution, it is natural to look for an interpretation of our procedure in terms of the Normalized Cuts algorithm. We find that our use of the Laplacian eigenvectors may be interpreted as introducing an interactive seeding interface into Normalized Cuts.

The structure of the paper is as follows: In Section 2 we review the Random Walker algorithm and show how we can use precomputed eigenvectors to produce an approximate solution, even in the absence of knowing the location of the user input seeds. We then develop the connection between this precomputed Random Walker algorithm and Normalized Cuts. In Section 3 we give results demonstrating that the algorithm correctly converges to the Random Walker segmentation as more eigenvectors are used, and that our approximation preserves the quality segmentation properties of the Random Walker algorithm. Section 4 draws conclusions and discusses future work.

2. Method

We begin by fixing our notation. A **graph** consists of a pair $G = (V, E)$ with **vertices (nodes)** $v \in V$ and **edges** $e \in E \subseteq V \times V$, with $N = |V|$ and $M = |E|$. 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} and is assumed here to be nonnegative. 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., $w_{ij} = w_{ji}$). An image may be associated with a graph by identifying each pixel with a node and defining an edge set to represent the local neighborhood relationship of the pixels (e.g., a 4-connected lattice).

The Random Walker segmentation algorithm of [9] computes the probability, for each pixel, that a random walker leaving that pixel will first arrive at a foreground seed before arriving at a background seed. It was shown in [9] that these probabilities may be calculated analytically by solving a linear system of equations with the graph Laplacian matrix. The **Laplacian matrix** is defined as

$$L(i,j) = \begin{cases} d_i & \text{if } i = j, \\ -w_{ij} & \text{if } v_i \text{ and } v_j \text{ are adjacent nodes,} \\ 0 & \text{otherwise,} \end{cases} \quad (1)$$

where $L(i, j)$ is indexed by vertices v_i and v_j.

Given a set of foreground seeds, V_F, and background seeds, V_B, where $V_F \cap V_B = \emptyset, V_S = V_F \cup V_B$, we can compute the probabilities, x_i, that a random walker leaving node v_i arrives at a node in V_F before arriving at a node in V_B by solving

$$L_U x_U = -B x_S. \quad (2)$$

The variable x_U represents the set of probabilities corresponding to unseeded nodes, x_S is the set of probabilities corresponding to seeded nodes (i.e., '1' for foreground nodes and '0' for background nodes) and L_U, B correspond to the matrix decomposition of L

$$L = \begin{bmatrix} L_S & B \\ B^T & L_U \end{bmatrix}. \quad (3)$$

Our goal is to find an approximate solution to x_U *without knowing* V_F *and* V_B. The approach to this problem will be to observe that multiplying L with x (containing values corresponding to the seeds) will produce some f, i.e.

$$Lx = f. \quad (4)$$

Then, if we perform an eigenvector decomposition of L

$$Q\Lambda Q^T x = f, \tag{5}$$

we can compute x up to a constant from

$$x = Ef, \tag{6}$$

where E is the pseudoinverse of L, equaling $E = Q\Lambda^{-1}Q^T$ in which $\Lambda(1,1) = 0$. The reason that $\Lambda(1,1) = 0$ is because, for a connected graph, L has a single eigenvalue corresponding to zero and the corresponding eigenvector is constant [1]. However, since $f^T 1 = 0$, by virtue of (4), we can ignore the zero eigenvalue and constant eigenvector in (6), since this entry will contribute nothing to x. Due to the constant vector nullspace of L, this procedure for forming x in (6) is correct to a constant value. The issue of how to determine this constant will be addressed later.

In order to limit the computational burden, storage and time complexity of our online algorithm, we can define a K-approximation to x as

$$x_K = Q_K \Lambda_K^{-1} Q_K^T f, \tag{7}$$

in which K is the number of eigenvectors used to produce an approximation to x. If $K = N$ then $x_K = x$, up to a constant. Clearly, from (6), the most effective eigenvectors to use will be those that correspond to the smallest eigenvectors. To avoid notational clutter we will assume for the remainder of the exposition that some K has been fixed and write all the variables without the K subscripts. Note that, for a constant K and known f, the approximation of x described in (7) is computed in time that is linear in the number of nodes (pixels). Taking the approach of solving (7) allows us to perform our precomputation of the eigenvectors in advance of knowing the seed locations, since the seed locations only serve to produce a different f vector. The primary question we face in the next section is how to know what the f vector will be for an input set of seed locations.

2.1. Determination of the right hand side

If the foreground and background seed sets consist of just a single node each, i.e., $|V_F| = |V_B| = 1$ then, for $v_f \in V_F$ and $v_b \in V_B$, f is known to be [1]

$$f_i = \begin{cases} \rho & \text{if } i = f, \\ -\rho & \text{if } i = b, \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

where ρ is the effective conductance between nodes v_f and v_b [1]. Therefore, when a single foreground and background seed are input, it is sufficient to treat $\rho = 1$, compute x with (6) and then normalize the computed x to lie between zero and one in order to get the random walker probabilities computed from (2).

In the more typical use case of more than a single foreground and background seed being input, the authors are unaware of any known method for determining f without knowing the variable x in advance. Consequently, we now present a method for approximating f in constant time (for a fixed number of seeds) with the precomputed eigenvectors in Q.

We begin by noting that application of E to both sides of (4) results in

$$\left(I - gg^T\right) x = Ef = Q\Lambda^{-1}Q^T f, \tag{9}$$

where g is the eigenvector of L corresponding to the zero eigenvalue. We can decompose f into $[f_S; f_U]$ (using "MATLAB notation"), corresponding to the seeded and unseeded nodes, and note that $f_U = 0$. Since $f_U = 0$, our primary interest is in calculating f_S, from which we can find x. In order to find f, we decompose E into

$$E = \begin{bmatrix} E_S & R \\ R^T & E_U \end{bmatrix}. \tag{10}$$

Note that

$$R^T = Q_U \Lambda^{-1} Q_S^T, \tag{11}$$

which means that R can be approximated by the computed eigenvectors comprising Q.

From (4) and (9) we know that

$$L_S x_S + B x_U = f_S, \tag{12}$$

$$x_U + g_U g_U^T x_U = R^T f_S, \tag{13}$$

which combine to form

$$\left(I - BR^T\right) f_S = L x_S - B g_U g_U^T x_U. \tag{14}$$

Let

$$P = \left(I - BR^T\right). \tag{15}$$

In order to find f_S, we need to solve the linear system of equations defined by (14) which is of order $|S|$. We may handle the unknown x_U on the RHS by replacing it with the single unknown $\alpha = g_U^T x_U$. Now, in order to solve (14) for f_S, we decompose $f_S = \hat{f} - \alpha\tilde{f}$ and solve

$$P\hat{f} = L_S x_S, \tag{16}$$

$$P\tilde{f} = B g_U. \tag{17}$$

Note that, by definition of g,

$$0 = g^T L x = g^T f = g^T \left(\hat{f} - \alpha\tilde{f}\right). \tag{18}$$

Therefore, we may solve for the unknown α by computing

$$\alpha = \frac{g^T \hat{f}}{g^T \tilde{f}}. \tag{19}$$

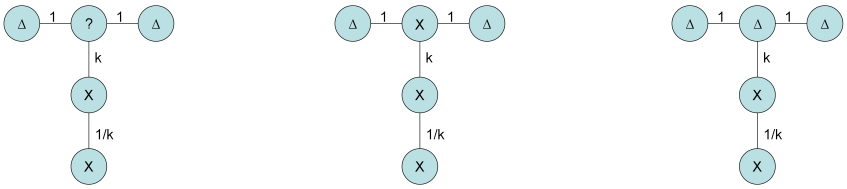


Figure 1. Illustration of problems with two approaches to seeded clustering in spectral coordinates with multiple seeds. (a) Example graph — Two nodes are seeded with the 'x' label (foreground) and two nodes are seeded with the 'triangle' label. The issue is to decide the label of the node marked with a question mark (the unlabeled node). The edges connecting the unlabeled node to the triangle labels have unit weight, while the edge connecting the unlabeled node to the 'x' label has weight k and the edge connecting the two 'x' labeled nodes has weight $\frac{1}{k}$. (b) Consider labeling the unlabeled node by assigning it the label of the closest seed (in spectral space). If $k = 1 + \epsilon$, then the unlabeled node takes the 'x' label, even though this labeling gives a suboptimal Normalized Cuts value of the partition. (c) Consider labeling the unlabeled node by assigning it the label of the seed group with an average spectral coordinate closer to the unlabeled node. If $k = 1000$, the unlabeled node is erroneously labeled 'triangle', even though it is strongly connected to an 'x' seed.

Given f_S, we may compute from (13) that

$$x_U = R^T f_S - \alpha g_U, \tag{20}$$

representing the approximate random walker probabilities, which may be thresholded at 0.5 in order to produce a final segmentation (23). In our experience, an excellent approximation of x may be obtained for even a small K (see Section 3). However, even a small error in the computation of α can lead to an inaccurate post-thresholded segmentation. Therefore, we have adopted the procedure of choosing the threshold of x that produces the partition with the best Normalized Cuts value. We note that, depending on the number of eigenvectors used in the approximation, it may be possible that the thresholded segmentation results in a small number of nodes disconnected from their respective seeds. In these cases, a connected component procedure could be employed.

2.2. Algorithm summary

The algorithm has two parts: 1) An "offline" algorithm that has no knowledge of the seeds that the user will use to segment an object, 2) An "online" algorithm that inputs seeds from a user and finds the segmentation.

The "offline" procedure:

1. Input an image and compute edge weights, according to e.g.,

$$w_i = \exp\left(-\beta(I_j - I_k)^2\right) \quad \text{for } \{v_j, v_k\} \in e_i, \tag{21}$$

   where I_j indicates the image (volume) intensity at voxel v_j.

2. Build the Laplacian matrix, L of (1).

3. Compute K eigenvectors of L, Q.

Efficient computation of the eigenvectors of L has been well-studied in the Normalized Cuts literature [18, 8].

Given user-placed seeds, the "online" procedure is:

1. Using the precomputed Q, generate a K-estimate of P in (15) using the K-estimated R^T of (11).

2. Solve (16) and (17) with the K-estimated P.

3. Calculate α from (19).

4. Generate x_U from (20).

5. Threshold x to produce the partition with the best Normalized Cuts value.

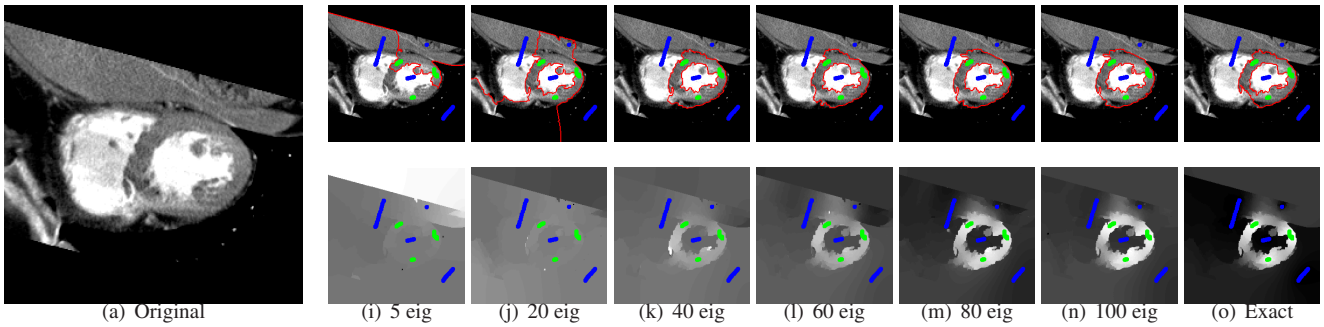
The most computationally intense step in the "online" procedure is the solution of (16) and (17). However, if the number of seeds is constant, this step requires constant time (for a given image resolution). Solving a full (i.e., non-sparse) set of equations for a matrix obtained from 500 seeds (i.e., having size 500×500) via LU decomposition requires roughly 0.04s in MATLAB on a Pentium 4 (2.8GHz) with 1GB of RAM. Note that any seeds for which all of the neighboring nodes are also seeds need not be used in (16) and (17), since the corresponding f_S entries are zero. Therefore, for a constant number of seeds and eigenvectors, the online procedure has a complexity that is linear in the number of nodes.

2.3. Relationship to Normalized Cuts

The above procedure for approximating the Random Walker solution by precomputing eigenvectors of the Laplacian matrix can also be interpreted as an interactive version of the Normalized Cuts algorithm [18]. To see this connection, recall that the Normalized Cuts method is used to find a two-way partition of a graph by thresholding the second smallest generalized eigenvector associated with the problem

$$Ly = \lambda Dy, \tag{22}$$

where y is the generalized eigenvector associating one value with each node, λ is the eigenvalue and D is a diagonal matrix such that $D(i,i) = d_i$.



(a) Original (i) 5 eig (j) 20 eig (k) 40 eig (l) 60 eig (m) 80 eig (n) 100 eig (o) Exact

Figure 2. Since our precomputation method produces an approximation to the Random Walker image segmentation algorithm, we want to know how many eigenvectors are necessary to precompute in order to get a quality segmentation. Foreground seeds are given in green, background seeds in blue and the segmentation boundary is outlined in red. Top row: Segmentation result with the use of 5, 20, 40, 60, 80 and 100 eigenvectors. The Random Walker segmentation is on the far right (equivalent to using N eigenvectors). Bottom row: The potential function corresponding to each of the segmentations, using progressively more eigenvectors.

Although the Normalized Cut method traditionally employs only a single (generalized) eigenvector to bipartition a graph (corresponding to the smallest nonzero eigenvalue), Shi and Malik also suggest employing a K-means algorithm in the coordinate space defined by the "coordinates" assigned to each node in successive eigenvectors. Such coordinates are sometimes referred to as **spectral coordinates** and have previously been used for clustering (e.g., [3]). In the Normalized Cuts interpretation of our method, the spectral coordinates are used to define a distance measure, from which the distance from each node to the seeds may be computed and then partitioned according to which seed has shorter distance. Specifically, consider the case of two single seeds, $V_F = v_f, V_B = v_b$. We may define the desired partition as

$$\begin{aligned} \text{Foreground} &= \{v_i |\ \text{dist}(v_i, v_F) < \text{dist}(v_i, v_B)\}, \\ \text{Background} &= \{v_i |\ \text{dist}(v_i, v_B) \geq \text{dist}(v_i, v_F)\}, \end{aligned} \tag{23}$$

where we define

$$\text{dist}(v_i, v_j) = r_{ij}^T Y \Lambda^{-1} Y^T r_{ij}, \tag{24}$$

Y is the matrix of all generalized eigenvectors, taking column i as y_i, Λ^{-1} is a diagonal matrix with $\Lambda(i,i) = \frac{1}{\lambda_i}$ and r_{ij} is an indicator vector taking values

$$r_{ij}(k) = \begin{cases} 1 & \text{if } i = k, \\ -1 & \text{if } j = k, \\ 0 & \text{otherwise.} \end{cases} \tag{25}$$

The inclusion of Λ^{-1} in the definition of distance (24) is the key to the connection with the Random Walker algorithm (in [18], Shi and Malik perform clustering of the spectral coordinates of the nodes without weighting the spectral coordinates with their corresponding eigenvalues). By including Λ^{-1} in the definition of distance, we can interpret (24) as equivalent to the *effective resistance* (viewing the graph as a linear circuit where weights are equivalent to conductances) between two nodes. The effective resistance is also proportional to the *commute time* of a random walk on the weighted graph from node v_i to v_j. The commute time measures the expected number of steps that a random walker would take to pass from node v_i to v_j and then back again from v_j to v_i. The commute time has become a popular quantity for graph embedding (dimensionality reduction), graph matching and unsupervised clustering [17, 16].

To see the connection between the distance in spectral space defined by (24) and commute time, define the *normalized* Laplacian matrix as

$$\tilde{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}, \tag{26}$$

with corresponding eigenvector decomposition

$$\tilde{L} = Q \Lambda Q^T. \tag{27}$$

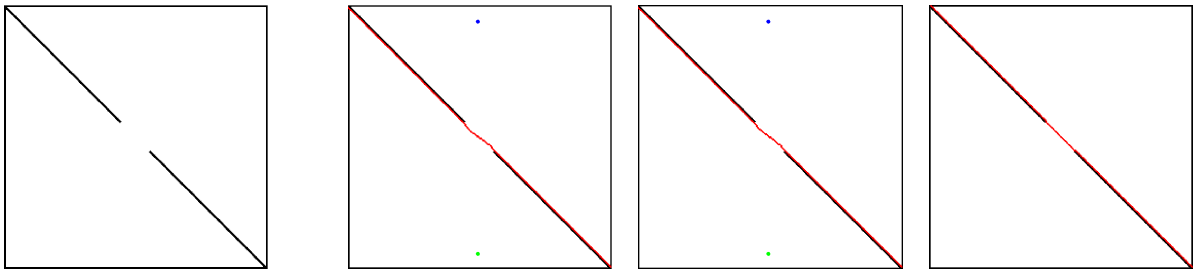
The eigenvectors, Q, are related to the generalized eigenvectors described by (22)

$$Y = D^{-\frac{1}{2}} Q. \tag{28}$$

Therefore, we may rewrite (24) in terms of the eigenvectors of $\tilde{L}$ as

$$\text{dist}(v_i, v_j) = r_{ij}^T D^{-\frac{1}{2}} Q \Lambda^{-1} Q^T D^{-\frac{1}{2}} r_{ij} = \sum_{k=2}^{N} \frac{1}{\lambda_k} \left(\frac{q_{ik}}{\sqrt{d_i}} - \frac{q_{jk}}{\sqrt{d_j}} \right)^2, \tag{29}$$

which equals the effective resistance between two nodes and is proportional to the commute time between v_i and v_j [12]. It has been shown that the Random Walker algorithm of [9] is equivalent to finding the smallest effective resistance between each pixel and the foreground seeds (considered as merged into a single node) and the background seeds



(a) Original (b) Random Walker (c) Approx. RW (40 eigs) (d) NCuts

Figure 3. The diagonal line example of [9]. Foreground seeds are given in green, background seeds in blue and the segmentation boundary is outlined in red. The ability of both the Random Walker and the Normalized Cuts algorithms to correctly handle this image suggests that the two methods are related. The automatic Normalized Cuts result was obtained after the third cut (i.e., the NCuts algorithm initially segments the disconnected black lines). Although a relatively small number of precomputed eigenvectors are used to approximate the Random Walker solution, the ability of the algorithm to segment objects with weak boundaries is preserved.

(similarly merged). When using a K-approximation to the Random Walker potentials, the summation in (29) occurs over K eigenvectors instead of N.

In the more realistic situation of multiple seeds, the method in Section 2.1 is equivalent to treating all foreground seeds as a single node and all background seeds as a single node. In the context of the distance function described above, it might seem more natural to adopt another approach to treating multiple seeds. Specifically, two other approaches for treating multiple foreground/background seeds appear obvious. The first approach is to assign the node, $v_i \in V_U$, to foreground (background) if the distance between the node is smaller to a foreground (background) seed than to any background (foreground) seed, i.e., $\text{Foreground} = \{v_i | \exists\, v_f \in V_F \text{ s.t. } \text{dist}(v_i, v_f) < \text{dist}(v_i, v_b) \quad \forall\, v_b \in V_B\}$. The second approach is to assign the unseeded node to foreground if the average distance between the node and all foreground seeds is smaller than the average distance between the node and all background seeds, i.e., $\text{Foreground} = \{v_i | \text{ average}\,(\text{dist}(v_i, v_f)) < \text{average}\,(\text{dist}(v_i, v_b)), \forall\, v_f \in V_F, v_b \in V_B\}$. Unfortunately, both of these approaches have serious drawbacks, which we now illustrate.

The first (minimum) approach produces poor segmentations because a node might be labeled background if the node has a shorter distance to a single, slightly closer background seed, even if the node has only a slightly more distant relationship with several foreground seeds. This situation is illustrated in Figure 1(b). Even though the unlabeled node is very close to two background seeds, the minimum approach would assign the node to the foreground because it is ϵ closer to a single foreground seed. As shown in the figure, such a rule would result in a partition with a suboptimal Normalized Cut value. Therefore, it seems that the correct treatment of multiple seeds is to simultaneously take into account all of the foreground/background seeds.

The second (averaging) approach also produces poor segmentations, but for different reasons. In this approach, the average distances are unduly biased by outlier seeds that are at a great distance (and possibly included by a user to address difficulties in the segmentation in other regions of the image). A succinct example of this problem is given by Figure 1(c). Even though the unlabeled node is nearly equivalent with the foreground seed (due to the near-infinite weight), the presence of a distant foreground seed causes the unlabeled seed to be labeled background. Using this construction the Normalized Cut value of this partition can be made arbitrarily poor for the averaging approach by increasing the value of k.

Using the approach of merging the nodes corresponding to all foreground seeds and all background seeds that is implicit in Section 2.1 produces the cut in the graph of Figure 1 with the lowest Normalized Cut value. Due to this connection between the precomputed Random Walker algorithm and a seeded version of Normalized Cuts, we suggest cutting the value of the approximated potential function at the value producing the best Normalized Cut value.

2.4. Generalization of results

The development here for using precomputed eigenvectors to approximately solve a linear system generalizes beyond the solution of (4). Specifically, if the matrix has a single zero eigenvalue and $f_U = 0$, then the procedure above can be used to find f_S and consequently x for any input values of the seed points and specification of set S. We now note that the Random Walker system of equations in (4) can alternately be formulated in terms of the normalized Laplacian as

$$D^{-\frac{1}{2}} L D^{-\frac{1}{2}} y = D^{-\frac{1}{2}} f, \tag{30}$$

in which

$$D^{-\frac{1}{2}} y = x. \tag{31}$$

Consequently, we could choose to precompute the eigenvectors of either the normalized or unnormalized Laplacian

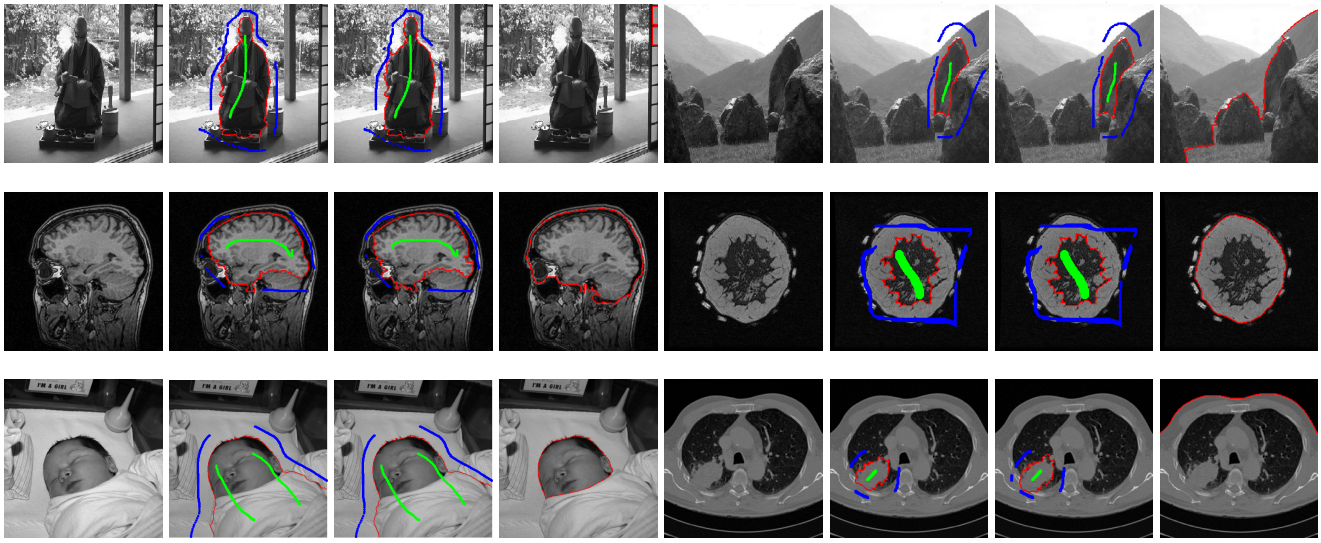


Figure 4. Several examples of our fast approximate Random Walker image segmentation algorithm. Foreground seeds are given in green, background seeds in blue and the segmentation boundary is outlined in red. For each group, the left image is the original, then the standard Random Walker segmentation, our fast approximation Random Walker algorithm using 40–80 eigenvectors and the (automatic) Normalized Cut segmentation. In our algorithm, the majority of the computation (i.e., production of eigenvectors) may be precomputed (without knowing the seed locations), making the "online" runtime fast.

matrix to obtain our approximation of x. Although either choice will give similar results, we have noticed that the eigenvalues of the normalized Laplacian seem to offer a somewhat faster convergence to the true x (likely due to the better behaved spectrum described in [5]). Therefore, in our experiments, we chose to employ eigenvectors of the normalized Laplacian.

3. Results

Our experiments are intended to study three issues: 1) How many eigenvectors must be precomputed in order to produce a good segmentation and how quickly does the solution converge to the Random Walker segmentation? 2) How fast does the online computation run? 3) Are important properties of the full Random Walker segmentation, such as weak boundary detection and quality segmentations, preserved?

In order to address the first question, we segmented the heart image of [9] with a number of precomputed eigenvectors equal to 5, 20, 40, 60, 80 and 100. Both the estimated potential function and the resulting segmentation were compared with the segmentation obtained from the Random Walker algorithm. The results in Figure 2 show that a small number of eigenvectors (less than 40) are insufficient to produce a quality segmentation. By using 40 or more eigenvectors, it is possible to obtain a segmentation that is qualitatively equivalent to the Random Walker result. However, in order to produce a potential function (probability distribution) that is similar to the Random Walker potential function, 80 or more eigenvectors were necessary.

An important property that contributes to the success of the Random Walker algorithm is the ability of the segmentations to "complete" weak or missing boundaries [9]. Additionally, the Normalized Cuts criterion inherently tolerates weak or missing boundaries if the segmentation is otherwise coherent. Consequently, we expect that our precomputed approximate Random Walker algorithm also exhibits robustness to weak or missing boundaries, even when using a small number of eigenvectors. We applied the conventional Random Walker segmentation to the weak boundary example of [9], the fast approximation method of this paper using 40 eigenvectors and the standard (unsupervised) Normalized Cuts algorithm. Figure 3 shows that our fast approximate Random Walker algorithm preserves the property that the produced segmentations are capable of completing a weak boundary.

Figure 4 shows several examples of our fast approximation Random Walker algorithm using 40–80 eigenvectors, compared with the standard Random Walker and (unsupervised) Normalized Cuts segmentation results. Although the segmentations we obtained were qualitatively similar to the Random Walker results, the online computation was much faster. For example, the segmentation of a 512×512 image with 40 eigenvectors required roughly 0.7 seconds (using unoptimized MATLAB code run on a Pentium 4 (2.8GHz) with 1GB of RAM) between the time that the seeds were placed and the production of the segmentation. The precomputation step of the corresponding eigenvectors required roughly 130s in MATLAB using the same ma-

chine. Using the MATLAB code for the Random Walker algorithm that is available on Grady's webpage, this same segmentation required 10.4s.

4. Conclusion

In this paper we addressed the question of splitting the computation time for the interactive Random Walker algorithm into a costly pre-interaction "offline" eigenvector computation and a very fast post-interaction "online" segmentation. We demonstrated that roughly 40–80 eigenvectors were sufficient to produce a segmentation of qualitatively the same quality as the full Random Walker solution with a speed advantage of over two orders of magnitude. Additionally, we showed that it was possible to view our approximation to the Random Walker potentials from the standpoint of distance in the "spectral coordinates" space defined by the weighted generalized eigenvectors employed by the Normalized Cuts algorithm. In the context of Normalized Cuts, the present paper could also be considered as a principled method for introducing user interaction into the popular Normalized Cuts algorithm.

Employing our precomputation approach is especially useful for the segmentation of high resolution images or 3D volumes, for which the exact (i.e., unapproximated) Random Walker algorithm can be too slow. On the 512×512 image in Section 3, we gained more than an order of magnitude in speed over the Random Walker algorithm, at the cost of a precomputation step. Additionally, the precomputation method presented in this paper could easily be coupled with any agglomeration ("supernode") method for coarsening a graph (e.g., watersheds [4], mean-shift [19]) to produce extremely fast "online" segmentations. Note that agglomeration is fundamentally different than the approach presented here, since the agglomeration super-pixels may cross weak boundaries (e.g., the entire white area in Figure 3 would be a single watershed basin), which would prevent smart algorithms like the Random Walker from detecting these weak boundaries. Another difference between our approach and agglomeration is that two seeds could potentially be grouped into the same super-pixel (e.g., watershed basin) by an agglomeration method, while our approach keeps each pixel separate and capable of attaining different labels.

Future work will focus on bounding the approximation to the Random Walker algorithm for a given set of eigenvectors and finding an image-dependent method for determining the number of eigenvectors to precompute, rather than employing a predefined number.

References

[1] N. Biggs. Algebraic potential theory on graphs. *Bulletin of London Mathematics Society*, 29:641–682, 1997.

[2] Y. Boykov and M.-P. Jolly. *Interactive graph cuts* for optimal boundary & region segmentation of objects in N-D images. In *Proc. of ICCV 2001*, pages 105–112, 2001.

[3] T. F. Chan, J. R. Gilbert, and S.-H. Teng. Geometric spectral partitioning. Technical Report CSL-94-15, Palo Alto Research Center, Xerox Corporation, 1994.

[4] C. Chefd'hotel and A. Sebbane. Random walk and front propagation on watershed adjacency graphs for multilabel image segmentation. In *Proc. of ICV*, Oct. 2007.

[5] F. R. K. Chung. *Spectral Graph Theory*. AMS, 1997.

[6] L. Cohen and R. Kimmel. Global minimum for active contours models: A minimal paths approach. *Int. J. of Comp. Vis.*, 24(1):57–78, 1997.

[7] A. X. Falcão, J. K. Udupa, S. Samarasekera, S. Sharma, B. H. Elliot, and R. de A. Lotufo. User-steered image segmentation paradigms: Live wire and live lane. *Graphical Models and Image Processing*, 60(4):233–260, 1998.

[8] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *IEEE PAMI*, 26(2):214–225, Feb. 2004.

[9] L. Grady. Random walks for image segmentation. *IEEE PAMI*, 28(11):1768–1783, Nov. 2006.

[10] U. Lauther. An extremely fast, exact algorithm for finding shortest paths in static networks with geographical background. In *Geoinformation und Mobilitat — von der Forschung zur praktischen Anwendung*, pages 219–230. Natur u. Wissenschaft, Institut fur Geoinformatik, 2004.

[11] H. Lombaert, Y. Sun, L. Grady, and C. Xu. A multilevel banded graph cuts method for fast image segmentation. In *Proc. of ICCV*, pages 259–265, Bejing, China, Oct. 2005.

[12] L. Lovász. Random walks on graphs: A survey. In *Combinatorics, Paul Erdõs is Eighty*, volume 2, pages 353–398. János Bolyai Math. Soc., 1996.

[13] R. H. Möhring, H. Schilling, B. Schütz, D. Wagner, and T. Willhalm. Partitioning graphs to speedup Dijkstra's algorithm. *J. of Experimental Algorithmics*, 11(2), 2006.

[14] E. Mortensen and W. Barrett. Interactive segmentation with intelligent scissors. *GMIP*, 60(5):349–384, 1998.

[15] A. Protiere and G. Sapiro. Interactive image segmentation via adaptive weighted distances. *IEEE Trans. on Image Proc.*, 16(4):1046–1057, April 2007.

[16] H. Qiu and E. R. Hancock. Image segmentation using commute times. In *Proc. of BMVC*, pages 929–938, 2005.

[17] M. Saerens, F. Fouss, L. Yen, and P. Dupont. The principal component analysis of a graph and its relationships to spectral clustering. In *Proc. of ECML*, pages 371–383, 2004.

[18] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE PAMI*, 22(8):888–905, Aug. 2000.

[19] J. Wang, P. Bhat, A. Colburn, M. Agrawala, and M. Cohen. Interactive video cutout. In *Proc. of SIGGRAPH*, volume 24, pages 585–594, 2005.