Three Interactive Graph-Based Segmentation Methods Applied to Cardiovascular Imaging

Leo Grady, Yiyong Sun, James Williams

ABSTRACT We examine the use of three techniques, graph cuts, isoperimetric minimization and random-walk partitioning for the interactive segmentation of cardiovascular medical images. These methods can often be used effectively without heavy reliance on learned or explicitly encoded priors. We illustrate, through the use of a toy problem, the basic difference in the performance characteristics of the methods. Subsequently, the suitability of each method to a particular segmentation application in the cardiovascular imaging domain is demonstrated.

1 Introduction

Isolation and quantification of structures in medical images is a continuous and varied source of segmentation problems. Segmentation methods which rely heavily on learned or explicit prior information often require significant customization before they can be applied to a specific problem. It is often preferable to use methods which can be quickly tested on the problem and then later enhanced with priors to improve accuracy. Graph partitioning algorithms are one such family of methods. In particular, we look at three segmentation techniques based on graph partitioning that at first glance may appear similar, but on closer inspection demonstrate unique behaviors. It is the distinct nature of these behaviors that can make one preferable over another for a specific application.

Although the graph cuts algorithm [17, 28] has been successfully employed in many applications, it is fundamentally a two-label algorithm. In fact, finding a minimal cut separating multiple terminals is an NP problem, although [7] provided an algorithm for getting within a bound of the optimal solution. The algorithm finds the smallest cut between two seed groups. In cases of weak object boundaries or small seed groups there is a tendency to find the cut that minimally encloses the seeds. The random walker algorithm proposed in [15] has a similar user interface (i.e., user "painting"), but does not suffer from the "small cut" problem and extends naturally to an arbitrary number of labels. The practical cost of this computation is currently higher than that of performing a binary graph cut

on a similarly sized image graph. As will be discussed later, this algorithm also has a formal relationship to the graph cuts algorithm.

Both graph cuts and the random walker algorithm require specification of seed points for each output label in the resulting segmentation. In the case where a foreground/background segmentation is desired, a user is often interested in specifying only a few pixels in the foreground region instead of labeling pixels in both the foreground and background. Additionally, if one wants to apply one of these algorithms by specifying seeds automatically, it is easier to automatically specify foreground seeds than both foreground and background seeds. The isoperimetric algorithm of [14] naturally extends the random walker algorithm to a situation where only foreground labels are provided. Given a foreground-labeled pixel (or pixel group), the isoperimetric algorithm may be derived by starting a random walker at each unlabeled pixel and calculating the expected number of steps before the walker reaches a labeled seed. As with the random walker algorithm, these probabilities may be calculated analytically with simulation of a random walk. The expected number of steps may be converted into a foreground/background segmentation by finding a threshold that produces the minimal *isoperimetric ratio*, from which this algorithm was originally derived [14]. Not surprisingly, there is a formal relationship between the random walker and the isoperimetric algorithm.

2 Characteristic Behaviors of the Algorithms

Table 1.1 illustrates differences between these three algorithms. These differences may appear to be subtle compared with the similarities. From a practical standpoint, one might wonder if these algorithms return similar results or whether we can expect essentially identical behavior. It is certainly true that applying each technique to a simple segmentation task (e.g., a black circle in a white background) will produce the same segmentation. However, Figures 1 and 2 are intended to illustrate the different "personalities" of each algorithm.

Figure 1 shows three concentric circles, with a foreground seed in the innermost circle and a background seed outside of the outermost circle. Given this user input, it is unclear how the "true" segmentation should be

Algorithm	Functional	Field	Constraints
Graph cuts	$Q(x) = x^T Lx$	$x = \{0, 1\}$	Seeds fixed to $\{0, 1\}$
Random walker	$Q(x) = x^T Lx$	$0 \leq x \leq 1$	Seeds fixed to $\{0, 1\}$
Isoperimetric	$Q(x) = \frac{x^T Lx}{x^T d}$	$0 \leq x$	Seeds fixed to $\{0\}$

TABLE 1.1. A tabulated comparison of the three algorithms. See text for details.

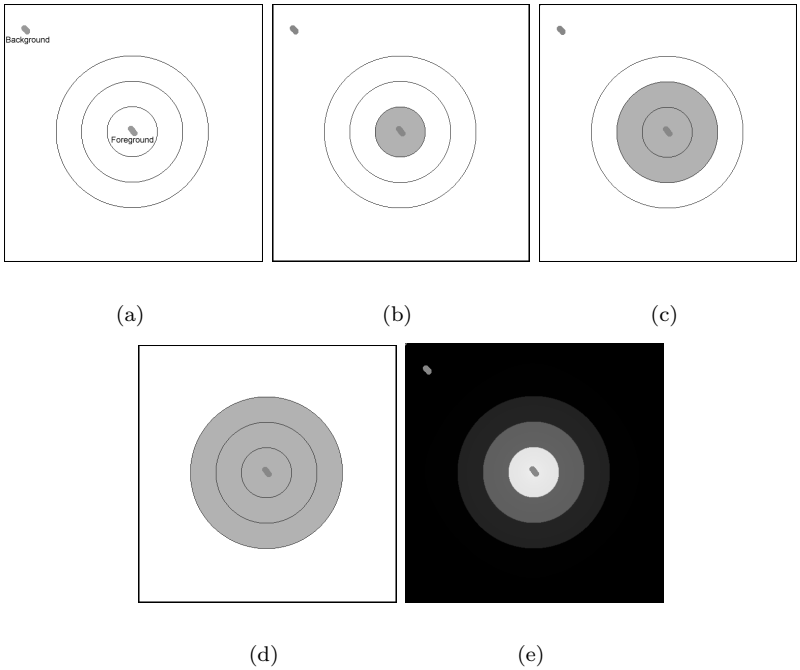
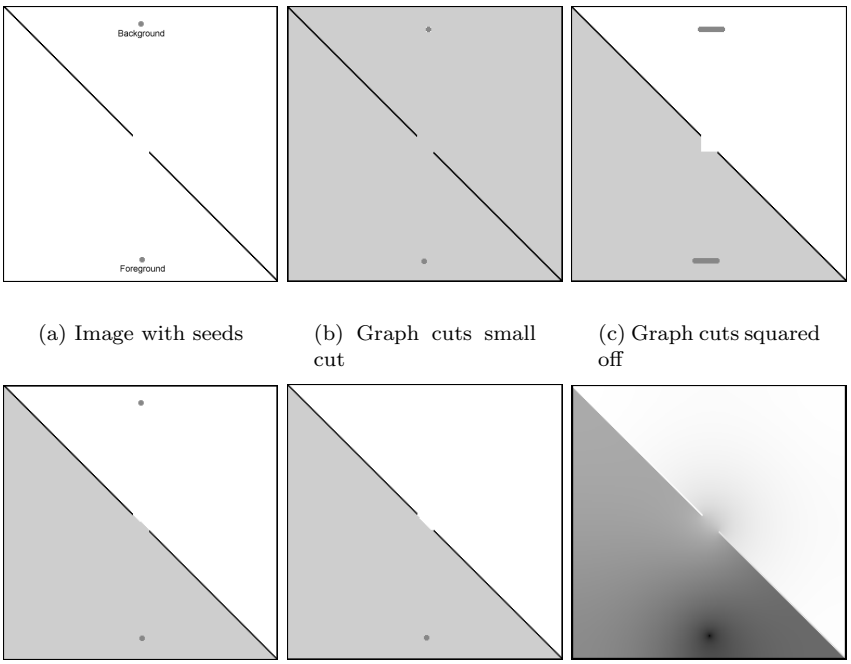


FIGURE 1. Different "personalities" of the three algorithms. (a) Input image with user-specified foreground and background seeds. (b) Graph cuts finds the innermost circle because it represents the smallest cut between seeds. (c) Random walk partitioning finds the middle circle because it is the most "equal" boundary between the two seeds. (d) The isoperimetric algorithm finds the outermost circle, since it minimizes the isoperimetric ratio in (1.8). Note that no background seeds were used when applying the isoperimetric algorithm. (e) Both the random walker and isoperimetric algorithms give a soft segmentation that is converted into a hard segmentation. For this image, each produce s the same soft segmentation (up to a scaling constant) that may be interpreted as a probability that a pixel lies in the foreground segment.

defined. Note that there is no ambiguity in the boundaries or difference in statistics of the regions. The real issue to be addressed by an algorithm that is given these seeds is: What does the user want? Depending on the user (or the goal), there are three valid outputs: The innermost object (small circle), the middle boundary between the foreground/background seeds (middle circle), or the entire group of objects (the outermost circle). These results are exactly those given by the three algorithms respectively. Note that, regardless of the number of concentric circles, graph cuts will always choose the smallest, random walker will choose the middle and isoperimetric will choose the largest.

Graph cuts chooses the innermost circle because it will find the cut hav-

ing the smallest cost. Each black/white transition of a ring will bear the same cost to cut, so the circle with smallest circumference will be preferred. This property is valuable because it finds the smallest object "unit" surrounding a foreground seed. The downside to this behavior is that a larger object (especially a textured object) requires many more seeds to locate. Additionally, this property is also the source of the "small cut" problem that can result in a return of the trivial boundary that minimally encloses the seeds.



(a) Image with seeds (b) Graph cuts small cut (c) Graph cuts squared off

(d) Random walker (e) Isoperimetric (f) Isoperimetric soft segmentation

FIGURE 2. Weak edge behavior of the three algorithms. (a) Diagonal line image with user-specified seeds. (b) "Small cut" behavior of graph cuts. (c) More seeds overcome small cut issue, but yields a "blocky" segmentation (see text). (d) Random walker. (e) The isoperimetric algorithm finds the desired cut, but without background seeds. (f) Solution to (1.11) thresholded to produce a hard segmentation of (e) having a minimum isoperimetric ratio.

A random walker starting from a given pixel will be more likely to reach the seed that requires crossing a smaller number of circles. Therefore, for an odd number of concentric circles, the random walker algorithm will always choose the middle circle. Note, however, that the probabilities form a "wedding cake" between the inner and outer circle that could be used (using another rule, e.g., minimum cut) to find any of the intermediate circles as seen in Figure 1. This behavior of the random walker algorithm is beneficial because the "small cut" problem is avoided and the most "equal" boundary between the seeds is found. However, this behavior can also result in the user having to place background seeds close to the foreground object in order to get the desired segmentation. Additionally, if an *even* number of concentric circles are present, the random walker algorithm will, in its neutrality between the two seeds, return a circle that threads the middle two circles instead of "snapping" the segmentation to the nearest circle.

The isoperimetric algorithm will also produce a "wedding cake" distribution , where each level corresponds to a circle. Note that the background seed is not employed in the isoperimetric algorithm. Given this solution, the isoperimetric algorithm looks for a threshold that produces a cut minimizing the isoperimetric ratio, defined as $h = \frac{x^T L x}{x^T d}$ (see below for detail). This ratio may be thought of intuitively as the ratio of the surface area (i.e., dual to the cut) to the volume. In a continuum setting, the isoperimetric ratio of a circle is $h = \frac{2\pi r}{\pi r^2} = \frac{2}{r}$ which will get smaller (and thus preferred) for a larger radius. Since the isoperimetric algorithm chooses the threshold that minimizes the isoperimetric ratio, the largest circle will be returned out of the "wedding cake" distribution produced by the solution.

Figure 2 illustrates another aspect of the personality of the algorithms. Here, a (broken) black line was drawn on a white image. All three algorithms exhibit the ability to locate the weak boundary even though there is no intensity cue at the gap and the statistics of both regions are identical. However, with graph cuts there are two issues. First, we initially see the "small cut" problem of when small seed groups are placed. However, even when the seed groups are made large enough, the algorithm finds a "squared off" cut that is unappealing. The reason for this squared off cut is because a 4-connected lattice is employed and, therefore, the squared off cut has the same cut cost as the diagonal cut, so one of these cuts is simply returned by the algorithm. This issue may be ameliorated by using a lattice with increased connectivity (e.g., 8-connected) at the cost of increased memory consumption. However, even for a 4-connected lattice, the random walker and isoperimetric algorithm neither exhibit a "squared off" solution nor suffer from the "small cut" problem.

3 Applications on CT Cardiovascular data

Computed tomography (CT) imaging has, in the last 5 years, undergone a revolution in resolution. Premium multi-slice scanners now have between 16 and 256 detector rows with gantry rotational latencies of less than half a second. These advances have meant not only an increase in spatial, but also in temporal resolution. CT angiography (CTA) uses injected contrast to opacify the cardiovascular system for high-resolution imaging. CTA is now the modality of choice for imaging 3D cardiovascular morphology. Due to the huge amount of data produced by these scanners (2GB volumes are now not uncommon), automated an semi-automated post-processing techniques are no longer a curiosity, they are indispensable tools for the radiologist.

3.1 Segmenting Individual Heart Chambers using Graph Cuts

Electrophysiological ablation procedures, like pulmonary vein isolation for curing atrial fibrillation, are today guided by a combination of electrophysiological and morphological criteria. Therefore it is helpful for the electrophysiologist to have 3D visualizations of the cardiac chamber which is subject to RF ablation available for pre-procedural planning, intra-procedural catheter guidance, and post-procedural follow-up. This requires the tools for heart chamber segmentation from CTA images.

The requirements of the segmentation tools are:

1. Accuracy: Segmentation shall be as close as possible to the ground truth provided by the user.
2. Easy to use: The tool shall need minimal user input.
3. Performance: The algorithm shall be fast and memory efficient.

Since fully automatic segmentation inherently has the problem of reliability and repeatability, an interactive segmentation is more attractive. Interactive methods take advantage of the user knowledge of the anatomy, and increase the overall procedure efficiency.

Even with contrast, accurate chamber segmentation with minimal user interaction is still a challenging problem. The difficulty is largely due to the weak boundaries between chambers. For example, the left atrium and left ventricle often have similar intensity due to direct blood pool connection through the mitral valve. Image noise and different imaging protocols across various sites also pose a challenge for the robustness of the segmentation algorithm.

3.2 Multi-Resolution Banded Graph Cuts

Boykov and Jolly [5] describe an interactive graph cuts algorithm. The algorithm assumes that some voxels have been identified as object or back-

ground seeds based on *a priori* knowledge from the anatomy. It computes a globally optimal binary segmentation that completely separates the object seeds and the background seeds.

Despite the power of finding a globally optimal solution, the major difficulty of the graph cuts algorithm lies in the enormous computational costs and memory consumption. Typical CT scans generate a 3D volume of hundreds of slices of images. For example, a volume of 512×512×300 has 75M voxels. A graph that stores the nodes and edges can easily consume over 1GB memory. Performing a graph cut segmentation on this volume using the max-flow algorithm [12] can take several minutes.

The approach we use to make the graph cuts algorithm practical in this context is a multi-resolution banded formulation. With the prior knowledge of the rough size of the chamber, the background seeds can be automatically detected after the user specifies an object seed inside the chamber. This makes it possible to achieve one-click-segmentation. What makes this possible is the compact shape of the heart chamber and the relatively homogeneous intensity in the chamber due to the contrast agent injection.

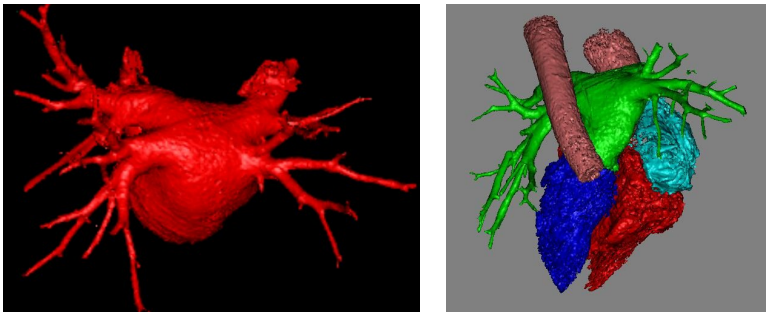
The idea of this algorithm is first to get a rough segmentation using graph cuts on a reduced resolution graph. The low resolution estimate is then used to guide a high resolution banded cut.

The algorithm contains five steps:

1. Apply a seeded region growing [1] from the object seeds in the low resolution volume. The growing stops when reaching a predefined maximal distance that depends on the *a priori* knowledge of the typical chamber size. The result may contain outliers due to leaking to left ventricle, the pulmonary arteries, and the bones.
2. Dilate a layer from the boundary of the region growing. The outer layer of the dilation is marked as background seeds.
3. Apply graph cuts in low resolution to get a rough segmentation of the left atrium. It is fast to solve because there are significantly fewer nodes for the low resolution graph.
4. Dilate a layer from the boundary of the rough segmentation to form a band whose inner boundary is marked as object seeds and outer boundary is marked as background seeds.
5. In high resolution, apply the graph cuts to get an accurate segmentation result. It is also fast to solve because the graph is built on the narrow band that only contains a few layers of voxels.

3.3 Empirical Results

Using the multi-resolution and banded graph cuts, the segmentation of the heart chambers can be achieved with a single mouse click inside the left atrium.



(a) (b)

FIGURE 3. Renderings of cardiac segmentations. (a) Left atrium. (b) Multiple chambers.

Fig. 3(a) shows the one click segmentation of the left atrium using the multi-resolution and banded graph cuts algorithm. It is segmented from a CT volume with the resolution of 512×512×370 and it takes less than 15 seconds to segment the left atrium on a Pentium 4 2.4GHz computer and uses less than 200MB memory. Fig. 3(b) shows the separate heart chambers and vessels, including left atrium, left ventricle, right atrium, right ventricle, and aorta. These chambers are segmented individually using the multi-resolution, banded graph cuts algorithm.

3.4 Random Walks for Simultaneous Chamber Segmentation

Where graph cuts are well suited to extraction of a single foreground object from a general background, extension to parallel multi-label segmentation does not follow naturally. Alternatively, the random walker algorithm can segment multiple regions in a single interactive step.

Assume that the user has provided K labeled pixels (hereafter referred to as **seed points** or **seeds**). For each unlabeled pixel, we ask: Given a random walker starting at this location, what is the probability that it first reaches each of the K seed points? It will be shown that this calculation may be performed exactly without the simulation of a random walk. By performing this calculation, we assign a K-tuple vector to each pixel that specifies the probability that a random walker starting from each unlabeled pixel will first reach each of the K seed points. A final segmentation may be derived from these K-tuples by selecting for each pixel the most probable seed destination for a random walker. In this approach, we treat an image (or volume) as a purely discrete object — a graph with a fixed number of vertices and edges. Each edge is assigned a real-valued weight corresponding to the likelihood that a random walker will cross that edge (e.g., a weight

of zero means that the walker may not move along that edge).

It has been established [21, 11] that the probability of a random walker first reaching a seed point exactly equals the solution to the Dirichlet problem [8] with boundary conditions at the locations of the seed points and the seed point in question fixed to unity, while the others are set to zero. A steady-state, DC circuit analogy is also given in [15]. Using the principle of superposition from circuit theory, it can be easily shown that the probabilities at each node sum to unity (as expected). For this reason, we need only solve $K-1$ systems, given K labels, since the remaining system is known via the unity constraint (i.e., at each node subtract the sum of the $K-1$ solutions at that node from unity to recover the solution to the remaining system).

3.5 The Random Walker Algorithm

We begin by defining a precise notion for a graph. A **graph** [20] consists of a pair $G = (V, E)$ with **vertices (nodes)** $v \in V$ and **edges** $e \in E \subseteq V \times V$. An edge, e, spanning two vertices, v_i and v_j, is denoted by e_{ij}. A **weighted graph** assigns a value to each edge called a **weight**. The weight of an edge, e_{ij}, is denoted by $w(e_{ij})$ or w_{ij}. The **degree** of a vertex is $d_i = \sum w(e_{ij})$ for all edges e_{ij} incident on v_i. Given a set of nonnegative weights, the probability that a random walker at node v_i transitions to node v_j is given by $p_{ij} = \frac{w_{ij}}{d_i}$. The following will also assume that our graph is connected.

In order to represent the image structure (given at the pixels) by random walker biases (i.e., edge weights), one must define a function that maps a change in image intensities to weights. Since this is a common feature of graph based algorithms for image analysis, several weighting functions are commonly used in the literature [26, 6, 14]. Additionally, it was proposed in [29] to use a function that maximizes the entropy of the resulting weights. In this work we have preferred (for empirical reasons) the typical Gaussian weighting function given by

$$w_{ij} = \mathrm{e}^{-\beta(g_i - g_j)^2}, \tag{1.1}$$

where g_i indicates the image intensity at pixel i. The value of β represents the only free parameter in this algorithm. In practice, we employ

$$w_{ij} = \mathrm{e}^{-\frac{\beta}{\rho}(g_i - g_j)^2} + \epsilon, \tag{1.2}$$

where ϵ is a small constant (we take $\epsilon = 10^{-6}$) and ρ is a normalizing constant $\rho = \max(g_i - g_j), \forall\, i, j$. The purpose of (1.2) is to keep the choice of β relevant to images of different quantization and contrast, as well as make sure that none of the weights go identically to zero (resulting in a possible disconnection).

The discrete Dirichlet problem has been discussed thoroughly in the literature [4, 11] and a convenient form for the solution is given in [16]. We will now review the method of solution.

Define the discrete Laplacian matrix [22] as

$$L_{v_i v_j} = \begin{cases} d_i & \text{if } i = j, \\ -w_{ij} & \text{if } v_i \text{ and } v_j \text{ are adjacent nodes,} \\ 0 & \text{otherwise,} \end{cases} \tag{1.3}$$

where $L_{v_i v_j}$ is used to indicate that the matrix L is indexed by vertices v_i and v_j.

Partition the vertices into two sets, V_M (marked/seed nodes) and V_U (unmarked nodes) such that $V_M \cup V_U = V$ and $V_M \cap V_U = \emptyset$. Note that V_M contains all seed points, regardless of their label. Then, we may reorder the matrix L to reflect the subsets

$$L = \begin{bmatrix} L_M & B \\ B^T & L_U \end{bmatrix}. \tag{1.4}$$

Denote the probability assumed at each node, v_i, for each label, s, by x_i^s. Define the set of labels for the seed points as a function $\phi(v_j) = s$, $\forall v_j \in V_M$, where $s \in \mathbb{Z}, 0 < s \leq K$. Define the $|V_M| \times 1$ (where $|\cdot|$ denotes cardinality) marked vector for each label, s, at node $v_j \in V_M$ as

$$m_j^s = \begin{cases} 1 & \text{if } \phi(v_j) = s, \\ 0 & \text{if } \phi(v_j) \neq s. \end{cases} \tag{1.5}$$

As demonstrated in [16], the solution to the combinatorial Dirichlet problem may be found by solving

$$L_U x^s = -B m^s, \tag{1.6}$$

which is just a sparse, symmetric, positive-definite, system of linear equations with $|V_U|$ number of equations and the number of nonzero entries bounded from above by $2|E| + |V|$. Since L_U is guaranteed to be nonsingular for a connected graph [3], the solution, x^s, is guaranteed to exist and be unique. Therefore, the potentials for all the labels may be found by solving the system

$$L_U X = -BM, \tag{1.7}$$

where X has columns taken by each x^s and M has columns given by each m^s. As mentioned above, one must solve only $K - 1$ systems, given K labels, since the probabilities at each node must sum to unity.

3.6 Numerical solution

Many good methods exist for solving large, sparse, symmetric, linear systems of equations (e.g., [13, 24]). A direct method, such as LU decomposition with partial pivoting has the advantage that the computation necessary

to solve (1.7) is only negligibly increased over the amount of work required to solve (1.6). Unfortunately, current medical data volumes frequently exceed $256 \times 256 \times 256 \approx 16M$ voxels, and hence require the solution of an equal number of equations. Furthermore, there is no reason to believe that the resolution will not continue to increase. Most contemporary computers simply do not have enough memory to allow an LU decomposition with that number of equations.

The standard alternative to the class of direct solvers for large, sparse systems is the class of iterative solvers [19]. These solvers have the advantages of a small memory requirement and the ability to represent the matrix-vector multiplication as a function. In particular since, for a lattice, the matrix L_U has a circulant nonzero structure (although the coefficients are changing), one may avoid storing the matrix entirely. Instead, a vector of weights may be stored (or computed on the fly, if memory is at a premium) and the operation $L_U x_U^s$ may be performed very cheaply. Furthermore, sparse matrix operations (like those required for conjugate gradients) may be efficiently parallelized [10, 18](e.g., for use on a GPU). Because of the relationship of (1.6) to a finite differences approach to solving the Dirichlet problem on a hypercube domain, the techniques of numerical solution to PDEs may also be applied. Most notably, the algebraic multigrid method [25, 9] achieves near-optimal performance for the solution to equations like (1.6).

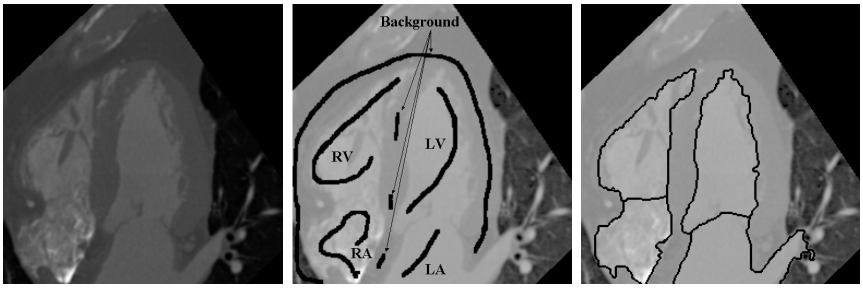
We have implemented the standard conjugate gradients algorithm with a modified incomplete Cholesky preconditioning [2], representing the matrix-vector multiplication implicitly, as described above on an Intel Xeon 2.4 GHz dual-processor with 1GB of RAM. Solution of (1.6) using conjugate gradients (tolerance = 10^{-4}, sufficient for the algorithm) for a 256×256 image with two randomly placed seed points required approximately 3 seconds.

To summarize, the steps of the random walker algorithm are:

1. Obtain a set, V_M, of marked pixels (seeds) with K labels from the user.
2. Using (1.1), map the image intensities to edge weights in the lattice.
3. Solve (1.7) outright for the probabilities or solve (1.6) for each label except the final one, f (for computational efficiency). Set $x_i^f = 1 - \sum_{s<f} x_i^s$.
4. Obtain a final segmentation by assigning to each node, v_i, the label corresponding to $\max_s (x_i^s)$.

3.7 Empirical Results

Using (1.1), we transformed a four-chamber view of a CTA heart volume into a weighted graph and applied the random walker algorithm. Results



(a) Input (b) Seeds (c) Segmentation

FIGURE 4. Random walker algorithm applied to a four-chamber slice of a cardiac CTA volume. (a) Original four-chamber slice. (b) User-specified seeds of each chamber (i.e., left ventricle, left atrium, right ventricle, right atrium, background). (c) Resulting segmentation boundaries.

are displayed in Figure 3.7. The random walker algorithm was chosen for this problem because five labels are required to segment the four chambers of the heart (each chamber plus the background). Segmentation of this 256×256 image required approximately 20 seconds of computation time. We have found this algorithm reliable on a large variety of CTA cardiac data with varying levels of noise.

3.8 Isoperimetric algorithm

Neither the graph cuts nor the random walker algorithm can be used when only foreground seeds are specified by the user. The isoperimetric algorithm of [14] may be interpreted as a natural extension of the random walker algorithm to a single seed group. The segmentation is based on computation of the expected number of steps a random walker will take, starting from each pixel, to find the user-specified seeds. However, in the original formulation [14], the isoperimetric algorithm was derived from a segmentation goal of minimizing the **isoperimetric ratio**

$$Q(x) = \frac{x^T L x}{\min(x^T d, (1^T - x^T) d)}, \tag{1.8}$$

where 1^T is the vector of all ones and d is the vector of node degrees. The indicator vector, x, is defined as

$$x_i = \begin{cases} 0 & \text{if } v_i \in \overline{S}, \\ 1 & \text{if } v_i \in S, \end{cases} \tag{1.9}$$

where S indicates the set of foreground nodes. Unfortunately, a combinatorial minimization of this problem is NP-Hard [23]. Consequently, the vector

x was relaxed to take real values and the "volume" (represented combinatorially by the denominator) was fixed to a constant, i.e., $x^T d = k$ (see [14] for a full exposition).

Using a Lagrange multiplier to perform a constrained minimization gives the energy as

$$Q(x) = x^T L x - \lambda(x^T d - k), \tag{1.10}$$

and the resulting minimum as

$$Lx = \frac{1}{2}\lambda d. \tag{1.11}$$

Since we are only concerned with relative values of the solution, and in order that (1.11) represents the expected number of steps required to find a seed, we ignore the scalar factor $\frac{\lambda}{2}$, setting $\frac{\lambda}{2} = 1$.

Although the Laplacian matrix in (1.11) is singular, the incorporation of user-specified seeds, i.e., $x_i = 0, \forall v_i \in V_M$ removes the singularity. The solution, x_i, at node, v_i, obtained through solution of (1.11) gives the expected number of steps that a random walker would take to find a seed node (see [27] for justification of this interpretation). Indeed, if one were to solve (1.11) for two seeds, v_1 and v_n, then

$$Lx^1 = \begin{bmatrix} -1^T d_U - d_n \\ d_U \\ d_n \end{bmatrix}, \tag{1.12}$$

where 1^T represents the vector of all ones, d_U represents the vector containing the degrees of unlabeled nodes. The reason that (1.12) holds is because premultiplication of both sides by 1^T produces zero on the left hand side, so the right hand side must be balanced. Then,

$$L(x^2 - x^1) = \begin{bmatrix} d_1 \\ d_U \\ -1^T d_U - d_1 \end{bmatrix} - \begin{bmatrix} -1^T d_U - d_n \\ d_U \\ d_n \end{bmatrix} = \begin{bmatrix} 1^T d_U + d_n + d_1 \\ 0 \\ -1^T d_U - d_n - d_1 \end{bmatrix}. \tag{1.13}$$

Since it is known [4] that multiplication of the solution to the random walker problem, x_{RW}, (given the same two seeds as above) by the Laplacian results in

$$Lx_{\mathrm{RW}} = \begin{bmatrix} \rho \\ 0 \\ -\rho \end{bmatrix}, \tag{1.14}$$

where ρ represents the **effective conductance** between nodes v_1 and v_n, the solution to two (or more) isoperimetric systems (1.11) also yields the random walker probabilities (up to a scaling and shift). It is intuitive that this should be true, since we would expect that a random walker having fewer expected steps to reach one seed over another would also be most likely to reach that seed first.

Computation of a solution to (1.11) yields a notion of how "far away" a given node is to a seed point, but it does not give a hard segmentation. Therefore, in accordance with [14], we convert the solution to (1.11) to a hard segmentation by thresholding the solution, x, at the value that produces a hard segmentation minimizing (1.8). Only n thresholds must be evaluated (i.e., one for each node) and the values of (1.8) may be evaluated quickly, leading to a fast production of a hard segmentation (see [14] for more details). Producing a hard segmentation in this way guarantees that all nodes belonging to the foreground segment are connected or, if more than one seed group is present, each group of foreground pixels is connected to a seed [14]. Note that this procedure for converting a soft segmentation into a hard segmentation is very similar to what is performed in the NCuts algorithm [26].

To summarize, the steps of the isoperimetric algorithm are:

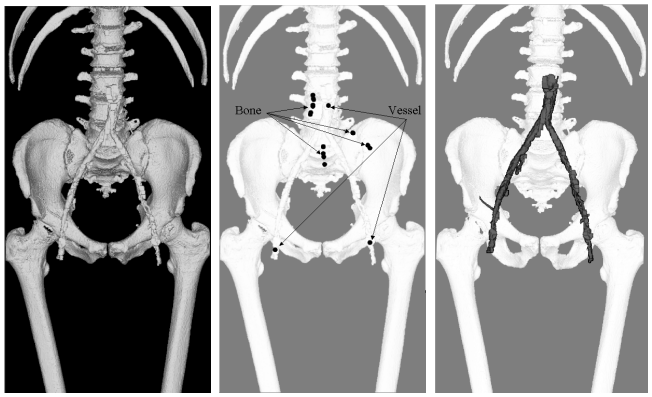
1. Obtain a set, V_M, of marked pixels (seeds) indicating foreground.
2. Using (1.1), map the image intensities to edge weights in the lattice.
3. Solve (1.11) for the expected number of steps taken by a random walker starting from each pixel to reach a node in V_M.
4. Obtain a hard segmentation by trying n thresholds, α, of x and choosing the segmentation that produces the smallest ratio given in (1.8). Assign each node, v_i, to foreground if $x_i \leq \alpha$ and to background if $x_i > \alpha$.

3.9 Bone-Vessel Separation

In CTA scans, contrast enhanced and calcified blood vessels can appear with the same intensity profile as bones. Further confounding an automatic bone/vessel segmentation is the fact that bones and vessels often touch each other and partial volume effects produce a gradual, diffuse boundary. Due to the difficulty of this situation, a user-guided mode is required to accurately separate bones and blood vessels.

The ideal user interface for this application is for a user to provide a single click on a blood vessel, without taking the time to label the bone. We first used an automatic segmentation algorithm to produce an initial bone/vessel segmentation. After this initial stage was completed, the user was able to click on a blood vessel and the isoperimetric algorithm was run on the subgraph of the original volume defined by the initial, automatic, segmentation. Application of the isoperimetric algorithm to this problem often gives the correct answer after placement of one or two vessel seed points. However, there were cases in which the vessel segmentation "bled" into the bone. To handle these circumstances, we allowed the user to enter "bone" seeds as well. Inclusion of a bone seed at node, v_i, had two effects:

1. Scaling the d_i entry on the right hand side of (1.11) by a large factor, $C >> 1$. This scaling has the effect of "pushing back" the solution to



(a) Input (b) Seeds (c) Vessel

FIGURE 5. Isoperimetric algorithm applied to vessel/bone separation. This problem is difficult because bone and vessel have similar intensities and touch each other (i.e., have a weak boundary) at several areas. (a) Original 3D image. (b) User-specified seeds of vessel (foreground) and bone (background). See text for meaning of background seeds in this application. (c) Resulting segmentation vessel segmentation.

(1.11) from node v_i. Alternatively, the scaling may also be interpreted as an additional injection of current at v_i in the circuit analogy of the isoperimetric algorithm presented in [14].

2. Limiting the threshold, α, to $\alpha < x_i$. With this limit, the resulting hard segmentation never encompasses the bone seed.

The performance of the isoperimetric algorithm is data dependent in this semi-automatic application. However, the algorithm supported interactive performance (sub-second) for the $64 \times 64 \times 64$ sub-volumes used in our application.

4 Conclusions

The three algorithms described in this section exhibit distinct behaviors that make each suitable for a given problem. In practice, once a particular method is found to work well on a given problem without priors, priors are then encoded or learned from examples to improve overall performance. It remains a challenge to improve the computational efficiency of these techniques. Already we have seen performance improvements of one to two orders of magnitude by mapping the core solvers for these problems into commodity graphics hardware (GPU).

5 References

[1] R. Adams and L. Bischof. Seeded Region Growing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):641–647, 1994.

[2] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. M. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. Number 43 in Miscellaneous Titles in Applied Mathematics Series. SIAM, November 1993.

[3] N. Biggs. *Algebraic Graph Theory*. Number 67 in Cambridge Tracts in Mathematics. Cambridge University Press, 1974.

[4] N. Biggs. Algebraic Potential Theory on Graphs. *Bulletin of London Mathematics Society*, 29:641–682, 1997.

[5] Y. Boykov and M.-P. Jolly. Interactive Organ Segmentation using Graph Cuts. In *Medical Image Computing and Computer-Assisted Intervention*, pages 276–286, Pittsburgh, PA, October 2000.

[6] Y. Boykov, O. Veksler, and R. Zabih. A New Algorithm for Energy Minimization with Discontinuities. In H.-E.-R. Pelillo-M., editor, *Energy Minimization Methods in Computer Vision and Pattern Recognition. Second International Workshop, EMMCVPR'99, York, UK, 26-29 July 1999.*, pages 205–220, 26-29 July 1999.

[7] Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, November 2001.

[8] R. Courant and D. Hilbert. *Methods of Mathematical Physics*, volume 2. John Wiley and Sons, 1989.

[9] J. E. Dendy. Black Box Multigrid. *Journal of Computational Physics*, 48:366–386, 1982.

[10] J. J. Dongarra, I. S. Duff, D. C. Sorenson, and H. A. van der Vorst. *Solving Linear Systems on Vector and Shared Memory Computers*. Society for Industrial and Applied Mathematics, Philadelphia, 1991.

[11] P. Doyle and L. Snell. *Random Walks and Electric Networks*. Number 22 in Carus mathematical monographs. Mathematical Association of America, Washington, D.C., 1984.

[12] A. Gibbons. *Algorithmic Graph Theory*. Cambridge University Press, 1989.

[13] G. Golub and C. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996.

[14] L. Grady. *Space-Variant Computer Vision: A Graph-Theoretic Approach*. PhD thesis, Boston University, Boston, MA, 2004.

[15] L. Grady and G. Funka-Lea. Multi-Label Image Segmentation for Medical Applications Based on Graph-Theoretic Electrical Potentials. In M. Šonka, I. A. Kakadiaris, and J. Kybic, editors, *Computer Vision and Mathematical Methods in Medical and Biomedical Image Analysis, ECCV 2004 Workshops CVAMIA and MMBIA*, number LNCS3117 in Lecture Notes in Computer Science, pages 230–245, Prague, Czech Republic, May 2004. Springer.

[16] L. Grady and E. Schwartz. Anisotropic Interpolation on Graphs: The Combinatorial Dirichlet Problem. Technical Report CAS/CNS-TR-03-014, Department of Cognitive and Neural Systems, Boston University, Boston, MA, 2003.

[17] D. Greig, B. Porteous, and A. Seheult. Exact Maximum a posteriori Estimation for Binary Images. *Journal of the Royal Statistical Society, Series B*, 51(2):271–279, 1989.

[18] K. Gremban. *Combinatorial Preconditioners for Sparse, Symmetric Diagonally Dominant Linear Systems*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, October 1996.

[19] W. Hackbusch. *Iterative Solution of Large Sparse Systems of Equations*. Springer-Verlag, 1994.

[20] F. Harary. *Graph Theory*. Addison-Wesley, 1994.

[21] S. Kakutani. Markov Processes and the Dirichlet Problem. *Proceeding of the Japenese Academy*, 21:227–233, 1945.

[22] R. Merris. Laplacian Matrices of Graphs: A Survey. *Linear Algebra and its Applications*, 197,198:143–176, 1994.

[23] B. Mohar. Isoperimetric Numbers of Graphs. *Journal of Combinatorial Theory, Series B*, 47:274–291, 1989.

[24] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2nd edition, 2002.

[25] Y. Shapira. *Matrix-Based Multigrid: Theory and Applications*, volume 2 of *Numerical Methods and Algorithms*. Kluwer Academic Publishers, 2003.

[26] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

[27] P. Tetali. Random Walks and the Effective Resistance of Networks. *Journal of Theoretical Probability*, 4(1):101–109, 1991.

[28] Z. Wu and R. Leahy. An Optimal Graph Theoretic Approach to Data Clustering: Theory and its Application to Image Segmentation. *IEEE Pattern Analysis and Machine Intelligence*, 11:1101–1113, 1993.

[29] X. Zhu, J. Lafferty, and Z. Ghahramani. Combining Active Learning and Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *Proceedings of the ICML 2003 workshop on The Continuum from Labeled to Unlabel Data in Machine Learning and Data Mining*, pages 58–65, 2003.