

A Seeded Image Segmentation Framework Unifying Graph Cuts And Random Walker Which Yields A New Algorithm*

Ali Kemal Sinop

Computer Science Department
Carnegie Mellon University, Pittsburgh, PA 15213
asinop@cmu.edu

Leo Grady

Department of Imaging and Visualization
Siemens Corporate Research, Princeton, NJ 08540
leo.grady@siemens.com

Abstract

In this work, we present a common framework for seeded image segmentation algorithms that yields two of the leading methods as special cases - The Graph Cuts and the Random Walker algorithms. The formulation of this common framework naturally suggests a new, third, algorithm that we develop here. Specifically, the former algorithms may be shown to minimize a certain energy with respect to either an ℓ_1 or an ℓ_2 norm. Here, we explore the segmentation algorithm defined by an ℓ_∞ norm, provide a method for the optimization and show that the resulting algorithm produces an accurate segmentation that demonstrates greater stability with respect to the number of seeds employed than either the Graph Cuts or Random Walker methods.

1. Introduction

Image segmentation is an important problem in computer vision. Traditionally, most segmentation algorithms have focused on unsupervised segmentation, grouping elements of the image according to a criterion such as homogeneity. Recently, supervised image segmentation methods have gained popularity since these methods give the user (or preprocessor) the ability to affect the segmentation as necessary for a particular application. Supervised segmentation algorithms generally come in three types: 1) The user is asked to provide pieces (usually points) of the desired boundary which are then completed by the algorithm, 2) The user is asked to specify an initial boundary that is “close” to the desired boundary which evolves to the desired boundary or 3) The user is asked to provide an initial labelling of some pixels as belonging to the desired object to be segmented or to the background, after which the algorithm completes the labeling for all pixels. Our focus in this work is on supervised segmentation algorithms of the last type.

An excellent example of the first type of supervised segmentation is the intelligent scissors (live wire) algorithm [10, 4]. In this algorithm, the image is treated as a graph and the user places several marks along the desired object boundary. Then, Dijkstra’s shortest path algorithm is used to find a minimum length path connecting all marks and this path is returned as the object boundary. However, these techniques are often prone to fail in the presence of noise and low contrast objects, since the shortest path is likely to “shortcut” the desired boundary. Additionally, a high level of accuracy is required from the user to place the marks precisely on the boundary.

The second type of supervised segmentation is exemplified by the family of active contour and level set methods [8, 13], in which the user is generally expected to initialize the algorithm with a boundary that is “close” to the desired boundary. After this initialization, the boundary evolves to a local minimum that is returned as the final segmentation. Many terms in the energy functional can be used to prefer certain boundaries over others. However, there are two main problems associated with these methods: 1) The energy terms are generally not convex, so that great care is necessary to avoid an incorrect local minimum, 2) If the segmentation is not correct, the standard avenues for correction are parameter adjustment or re-initialization, which may be difficult for a naive user.

In the third type of supervised segmentation methods, the user provides a partial labelling of the image (known as **seeds**), after which a complete labelling is constructed. Correction of an erroneous segmentation is accomplished by specifying additional labels for the initializing partial labeling. Two popular seeded segmentation methods are the Graph Cuts [3] and Random Walker [6] algorithms. Both of these methods treat the image as a graph and minimize certain energy functionals on this graph to produce a segmentation. Since these energy functionals are convex, it is possible to find the global optimum. Moreover, due to the algorithm definitions on general graphs, they may be applied without modification to an image of arbitrary dimen-

*Published in Proc. of ICCV 2007, Rio de Janeiro, Brazil, Oct., 2007

sion.

In both of the Graph Cuts and Random Walker methods, a weighted graph representation of the image is constructed. Nodes of this graph correspond to pixels in the image and edges are placed between nearby pixels. The edge weights are determined by a similarity measure on these pixels such that an edge connecting two pixels with a high similarity should have a large weight and vice versa.

In the Graph Cuts method [3], the foreground/background seeds are treated as source/sink nodes for a max-flow/min-cut operation. Using a max-flow/min-cut algorithm, a set of edges with the minimum total weight is found and then returned as the object boundary. A common problem with the Graph Cut method is the “small cut” behavior. Since this method tries to minimize the total edge weights in the cut, it may return very small segmentations as a result of low contrast, a small number of seeds or noise. The minimum-cut criterion may also cause problems by returning solutions in which the boundary takes a “shortcut” over a protruded section of the object in an attempt to minimize boundary length.

In the Random Walker method [6], the edge weights are treated as probabilities (when normalized by node degree) of a particle at one node traveling to a neighboring node. Given seeds, one may then compare the probability that a particle initiating at any node (pixel) travels first to the foreground or background seeds and assign that pixel to the corresponding label. It was shown in [6] that it is possible to compute these probabilities for each pixel analytically without any simulation of random walks.

In this paper, we propose a general seeded image segmentation algorithm. We then show that this general algorithm takes the form of either the Graph Cuts or Random Walker algorithms, depending on the choice of one parameter — The norm by which the solution gradient is computed. After establishing that the general algorithm will yield these two image segmentation methods, we proceed to examine a previously unexplored algorithm resulting from a different choice of this same parameter.

The paper is organized as follows: In Section 2, we give the general seeded segmentation algorithm on a combinatorial (graph-based) domain, show that the Graph Cuts and Random Walker algorithms can both be derived from this framework and examine the third case suggested. In Section 3, we perform a quantitative stability comparison of all three algorithms and qualitatively compare the segmentation results on a series of images. We conclude in Section 4 with a summary of this framework and the new algorithm, with suggestions for future work.

2. Method

This section is divided into two parts. In the first part, we will give a general algorithm for seeded segmentation and

Given Foreground F and Background B Seeds, $q \in \mathbb{R}^+$

Let $x \leftarrow \arg \min \|C^{\frac{1}{2}}Ax\|_q^q =$

$$\left[\sum_{e_{ij} \in E} (w_{ij}|x_i - x_j|)^q \right]^{\frac{1}{q}},$$

subject to $x(F) = 1$ and $x(B) = 0$.

Output Segmentation $s_i = \begin{cases} 1 & \text{if } x_i \geq \frac{1}{2}, \\ 0 & \text{if } x_i < \frac{1}{2}. \end{cases}$

Figure 1. The Graph Cuts and Random Walker segmentation algorithms may be viewed as particular instantiations of a more general algorithm, termed here as “Algorithm A”. Algorithm A gives the segmentation results of Graph Cuts when $q = 1$ and Random Walker when $q = 2$. We explore the new algorithm derived in the $q = \infty$ case.

then show that the Graph Cuts and Random Walker segmentation algorithms are special cases of this general algorithm. In the second part, we investigate a previously unexplored third case of the general algorithm.

2.1. A general seeded segmentation algorithm

Given the combinatorial formulation of the Graph Cuts and Random Walker algorithms, we first fix our notation before proceeding to establish the general algorithm. 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 following will also assume that our graph is connected and undirected (i.e., $w_{ij} = w_{ji}$). In this paper, we will employ an 8-connected lattice as our neighborhood structure.

Consider the following algorithm on an image domain, V , termed **Algorithm A**: 1) Assume that we are given (interactively or via a preprocessor) a set of pixels, F , labeled foreground and a set of pixels, B , labeled background, such that $F, B \subseteq V, F \cap B = \emptyset$. 2) Find a solution to

$$\begin{aligned} \min \quad & \|C^{\frac{1}{2}}Ax\|_q^q = \left[\sum_{e_{ij} \in E} (w_{ij}|x_i - x_j|)^q \right]^{\frac{1}{q}}, \\ \text{s.t.} \quad & x_F = 1, \\ & x_B = 0. \end{aligned} \tag{1}$$

where A is the $m \times n$ node-edge incidence matrix defined as

$$A_{e_{ij}v_k} = \begin{cases} +1 & \text{if } i = k, \\ -1 & \text{if } j = k, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

and the $m \times m$ matrix C is the constitutive matrix, which is a diagonal matrix with the *square* weights of each edge along the diagonal. 3) Assign a set of segmentation labels,

Choice of q	1	2	∞
Name	Graph Cut	Random Walk	This Algorithm
Objective Function	$\sum_{e_{ij} \in E} w_{ij} x_i - x_j $	$\sum_{e_{ij} \in E} w_{ij}^2 x_i - x_j ^2$	$\max_{e_{ij} \in E} w_{ij} x_i - x_j $
Obj. Func. Interpretation	Boundary Cut	Effective Conductance	Minimal Lipschitz Extension
Optimization Method	Maximum Flow	Solution of a Sparse Linear System	Shortest Path
Uniqueness	Not Unique*	Unique	Not Unique

Figure 2. Comparison of different algorithms obtained from “Algorithm A” in Figure 1. Uniqueness — Unlike the $p = \infty$ case, arbitrarily small perturbations of the weights will resolve the $p = 1$ degeneracy. However, this resolution will be arbitrary.

$s : V \mapsto \{0, 1\}$ such that

$$s_i = \begin{cases} 1 & \text{if } x_i \geq \frac{1}{2}, \\ 0 & \text{if } x_i < \frac{1}{2}. \end{cases} \quad (3)$$

This general algorithm is summarized in Figure 1. We note that the above formulation could be interpreted from a continuous perspective in which (1) is replaced by

$$\min \|\nabla \psi\|_q^q \quad (4)$$

where the sets F and B represent internal Dirichlet boundary conditions and the gradient operator is modified by an inhomogeneous metric tensor that changes spatially in response to intensity changes [9]. Note that the above formulation of Algorithm A offers a natural method by which the Graph Cuts and Random Walker algorithms could be combined in the sense that the corresponding energy functionals could be added together and jointly minimized.

For many graph based segmentation algorithms [14, 3, 6, 7], the edge weights encode image intensity changes. A standard weighting function in these algorithms that we shall employ here is

$$w_{ij} = \exp\left(-\sqrt{\beta|g_i - g_j|^2 + |h_i - h_j|^2}\right), \quad (5)$$

where g_i indicates the image intensity at pixel i , h_i indicates the spatial position of pixel i and β is a free parameter. In the context of (4), these weights would be equivalent to a spatially varying metric. Note that pixel affinity could equally be derived from changes in color, texture, etc.

The definition of Algorithm A leaves only one remaining choice — What value of q should be employed? In the following sections, we show that the choice of $q = 1$ leads to the Graph Cuts algorithm and the choice of $q = 2$ leads to the Random Walker algorithm. This situation is summarized in Figure 2. Following the justification of the above statements, we proceed to consider the choice $q = \infty$. Note that a unary (data) term is often explicitly included in the formulation of Algorithm A for various metrics (e.g., in Graph Cuts [3] or Random Walker [5]). However, since a data term may be considered from the standpoint of “t-links” (i.e., a pairwise term between the unary node and a “phantom” seed), Algorithm A equally applies to the use of a unary (data) term for any of the three algorithms.

2.1.1 The $q = 1$ case — Graph Cuts

If we substitute $q = 1$ into (1), the second step of Algorithm A requires the solution of

$$\begin{aligned} \min \quad & |C^{\frac{1}{2}} Ax| = \sum_{e_{ij} \in E} w_{ij} |x_i - x_j|, \\ \text{s.t.} \quad & x_F = 1, \\ & x_B = 0. \end{aligned} \quad (6)$$

The dual formulation of (6) is

$$\begin{aligned} \max \quad & y_B, \\ \text{s.t.} \quad & A^T y = \delta_F y_F + \delta_B y_B, \\ & 0 \leq y_{ij} \leq w_{ij} \quad \forall e_{ij} \in E, \end{aligned} \quad (7)$$

where $\delta_{F,B}$ are vectors of length n with 1’s in entries corresponding to foreground and background seeds respectively, and zeros elsewhere.

Upon closer inspection, it can be seen that (7) is the maximum-flow problem, where y_B is the total flow, subject to the flow conservation principle (e.g., [12]). Therefore, formulation (1) gives rise to the min-cut/max-flow problem for $q = 1$. The primal problem (6) also demonstrates the minimum cut problem — Minimize a weighted cut, represented by x , between terminals F and B . The solution of (6) will be a binary solution, $x_i \in \{0, 1\}$, due to the totally unimodular property of the min-cut problem [12]. Since the solution of (6) will necessarily be binary, the thresholding step of Algorithm A will have no consequence, and the final labelling will be equal to the optimal solution of (6), i.e., $s = x$. As a result, the Graph Cuts image segmentation algorithm [3] gives the same solution as Algorithm A when $q = 1$.

2.1.2 The $q = 2$ case - Random Walker

It is even clearer to see that the $q = 2$ case corresponds to the Random Walker segmentation algorithm. For $q = 2$, (1) becomes

$$\begin{aligned} \min \quad & \|C^{\frac{1}{2}} Ax\|_2^2 = \sum_{e_{ij} \in E} w_{ij}^2 (x_i - x_j)^2 \\ & = x^T A^T C A x, \\ \text{s.t.} \quad & x_F = 1, \\ & x_B = 0. \end{aligned} \quad (8)$$

A minimization of this form is exactly described in the Random Walker algorithm formulation of [6] (for the case

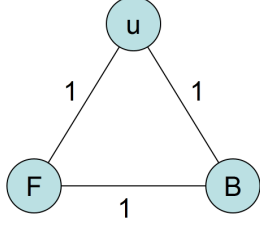


Figure 3. The solution to the $q = 1$ and $q = \infty$ cases are not unique, as illustrated by this graph. F is the foreground node (set to unity), B is the background node (set to zero). The remaining node, u , may take any value ranging between $[0, 1]$ without affecting the objective function in (1). For this reason, regularizing the $q = \infty$ case with an ℓ_1 term does not remove the degeneracy and we therefore use an ℓ_2 regularizer. Each edge has unity weight.

of two labels). Solution of (8) was shown in [6] to yield the desired random walker probabilities, which are then thresholded to produce the foreground/background labeling. Therefore, the Random Walker image segmentation algorithm with two labels gives exactly the same solution as Algorithm A when $q = 2$.

Hence, Algorithm A is shown to produce the Graph Cuts and Random Walker segmentation solutions when $q = 1$ and $q = 2$, respectively. We now proceed to examine the previously unexplored algorithm that is obtained from Algorithm A by allowing $q \rightarrow \infty$. The resulting segmentation algorithm is the subject of the remainder of this paper.

2.2. The $q = \infty$ case

If we let $q \rightarrow \infty$, (1) becomes

$$\begin{aligned} & \lim_{q \rightarrow \infty} \sqrt[q]{\sum_{e_{ij} \in E} w_{ij}^q |x_i - x_j|^q} \\ &= \underbrace{\max_{e_{ij} \in E} w_{ij} |x_i - x_j|}_{\rho} \lim_{q \rightarrow \infty} \underbrace{\sqrt[q]{\sum_{e_{ij} \in E} \left(\frac{w_{ij} |x_i - x_j|}{\rho} \right)^q}}_1 \quad (9) \\ &= \rho. \end{aligned}$$

Consequently, the overall optimization problem may be written as

$$\begin{aligned} & \min \max_{e_{ij} \in E} w_{ij} |x_i - x_j|, \\ & \text{st} \quad x_F = 1, \\ & \quad \quad x_B = 0. \end{aligned} \quad (10)$$

This form can be recognized as a combinatorial formulation of the minimal Lipschitz extension [2]. A method for solving (10) is the subject of the next section.

2.2.1 Solution

In the following sections, let $u \rightsquigarrow v$ denote a path in G from node $u \in V$ to $v \in V$. Also, let $\rho = \max_{e_{ij} \in E} w_{ij} |x_i - x_j|$ for a solution x to (10).

For any path $\pi : F \rightsquigarrow B$, we have the inequalities:

$$\sum_{e_{ij} \in \pi} |x_i - x_j| \geq |x_F - x_B| = 1, \quad (11)$$

$$\sum_{e_{ij} \in \pi} |x_i - x_j| \leq \sum_{e_{ij} \in \pi} w_{ij}^{-1} \rho. \quad (12)$$

Combining these two inequalities yields

$$\rho \geq \left(\sum_{e_{ij} \in \pi} w_{ij}^{-1} \right)^{-1}, \quad \forall \pi : F \rightsquigarrow B. \quad (13)$$

A maximum lower bound for ρ is attained with the shortest path between a foreground and background seed over the reciprocal weights (w_{ij}^{-1}).

Let d_i^F and d_i^B denote the shortest path length from node $v_i \in V$ to a foreground and background node using reciprocal weights respectively.

Theorem 2.1. *Let x be defined as*

$$x_i = \min\{\rho d_i^B, 1\}. \quad (14)$$

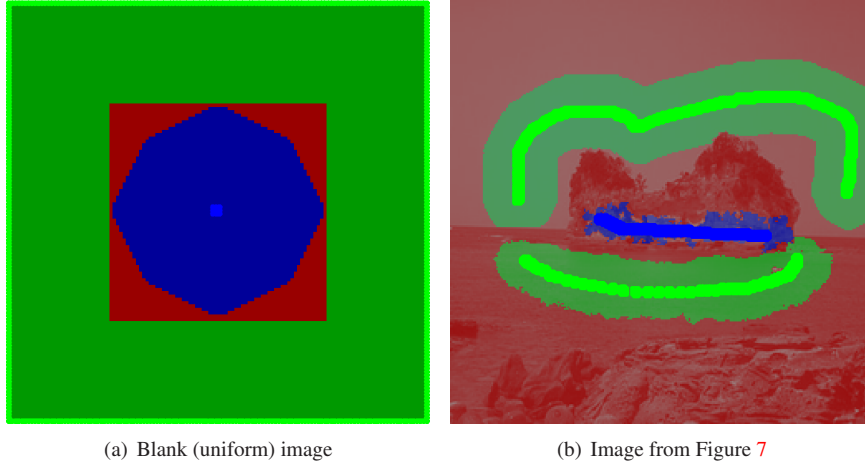
This potential vector is an optimal solution for (10).

Proof. We begin by checking the boundary conditions given in (10).

- $x_i = 1, i \in F$. Since $\rho^{-1} \leq d_i^B$ for all $i \in F$, $\rho d_i^B \geq 1$, which implies $x_i = 1$.
- $x_i = 0, i \in B$. For all $i \in B$, $d_i^B = 0$, therefore $x_i = 0$.
- $w_{ij} |x_i - x_j| \leq \rho$. Wlog, assume $x_i \geq x_j$, which implies that $d_i^B \geq d_j^B$. If $x_j = 1$, then $x_i = 1$. Therefore, this property is readily satisfied. Assume otherwise. We know that $d_i^B - d_j^B \leq w_{ij}^{-1}$ by the shortest path property. So, we have $w_{ij}(x_i - x_j) \leq w_{ij}(\rho d_i^B - \rho d_j^B) \leq \rho$.

□

A practical consequence of Theorem 2.1 is that the potential vector x may be computed in $O(m + n \log n)$ time by finding d_i^B for each node, which requires a single source (treating all nodes in F as a single node) all shortest paths computation. Although the above discussion offers a simple, efficient method of finding an optimal solution to (10), this solution is not, in general, unique [2]. It is a matter of debate as to how ultimately problematic non-uniqueness of a solution can be. However, from a computational standpoint, the solution returned for a non-unique functional will depend on the details of the solver. Since dependency on solver details dampens the utility of a global optimum, we will examine the use of regularization to produce a unique solution. In the following section, we will investigate this degeneracy issue further.



(a) Blank (uniform) image

(b) Image from Figure 7

Figure 4. Although the ℓ_∞ algorithm produces a non-unique solution, the segmentation boundary must lie within a certain “ambiguous region” that is simple to compute. In these figures, the red region illustrates this “ambiguous region” in which the segmentation may take either a foreground or background label. Regardless of the regularizer, pixels inside the blue region must take a foreground label and pixels inside the green region must take a background label. Note that the blue and green regions always touch at a point that is halfway along the (weighted) shortest path between the foreground and background seeds. The saturated blue and green pixels indicate the foreground/background seed locations.

2.2.2 Non-uniqueness of the solution

An example of the non-uniqueness of the solution to (10) is given by the triangle graph given in Figure 3. Additionally, the situation in Figure 3 illustrates the degeneracy of the ℓ_1 minimization of (6), as well. In this graph, the value of x_u can take any value between 0 and 1 without affecting the optimum of (10) or (6). Specifically, for any $x_u \in [0, 1]$, $\ell_1(x) = 2$ and $\ell_\infty(x) = 1$. Hence, there exists an infinite number of solutions for these problems and some regularization is required to obtain a unique solution. Regardless of the regularization method employed, a surprising aspect of the solution in the ℓ_∞ case (with regards to the segmentation) is that some pixels are guaranteed to take the foreground label (have solution greater than $\frac{1}{2}$) while others are guaranteed to take the background label (have solution less than $\frac{1}{2}$). However, there remains an “ambiguous region” in which the method of regularization determines the labeling. The next section shows how to calculate these regions for the ℓ_∞ algorithm.

2.2.3 Constraints on the Solution

Although the solution of $\ell_\infty(x)$ minimization problem is not unique, there are some common properties in all solutions. First we will bound the range of possible values for each node.

Theorem 2.2. *Let ρ be the minimum value of $\ell_\infty(x)$. Then, for any potential vector x satisfying the boundary conditions in (10), with $w_{ij}|x_i - x_j| \leq \rho$, we have*

$$1 - \rho d_u^F \leq x_u \leq \rho d_u^B, \forall u \in V. \quad (15)$$

Proof. For each node $u \in V$, let π^F and π^B denote the shortest paths from F and B to u_i respectively. Then

$$\sum_{e_{ij} \in \pi^F} (x_i - x_j) = 1 - x_u, \quad (16)$$

$$\sum_{e_{ij} \in \pi^F} (x_i - x_j) \leq \sum_{e_{ij} \in \pi^F} |x_i - x_j| \leq \rho d_u^F. \quad (17)$$

Combining these inequalities yields

$$\rho d_u^F \geq 1 - x_u \Rightarrow x_u \geq 1 - \rho d_u^F. \quad (18)$$

The same mechanism allows for derivation of the other inequality. \square

These inequalities allow us to find lower and upper bounds on the segmentation areas. In any optimal solution to (10), x_u will always be classified as background if $\rho d_u^B \leq \frac{1}{2}$ and as foreground if $\rho d_u^F \leq \frac{1}{2}$. This is an important property, as it allows us to put bounds on the possible segmentations in an efficient way. Additionally, these bounds demonstrate that the ℓ_∞ algorithm avoids the shrinking problem associated with Graph Cuts.

The above analysis also provides another property of the solution to (10). This property is that the solution for any nodes along the shortest path from F to B is fixed, regardless of the regularization method employed. Formally, if $d_u^F + d_u^B = \rho^{-1}$, meaning that node u lies on a shortest path between F and B , then $\rho d_u^B = \rho(\rho^{-1} - d_u^F) = 1 - \rho d_u^F$, which implies that x_u is fixed. The ambiguous region is illustrated in Figure 4 for three images: A blank (uniform) image, a weak boundary image and a natural image. Note that an 8-connected lattice was employed.

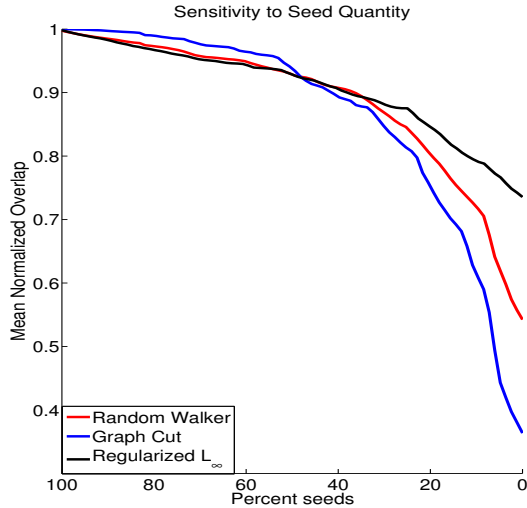


Figure 5. Sensitivity analysis of segmentation to the number of seeds. Because of the “small cut” phenomenon, the Graph Cuts segmentation is the least robust to the quantity of seeds, whereas regularized ℓ_∞ algorithm performs the best.

2.2.4 Regularization

In this section, we will examine various methods of regularizing (10) to yield a unique solution. We will examine the use of the other two norms (ℓ_1 and ℓ_2) as regularizers for solving the degeneracy problem. Employing the other two norms as regularizers offers one avenue for combining the ℓ_∞ algorithm with Graph Cuts or Random Walker.

ℓ_1 Regularization: A natural choice of regularization is to find the solution which, among all solutions with $\ell_\infty(x)$ minimum, also minimizes $\ell_1(x)$. It is possible to use a $\ell_1(x)$ regularizer efficiently, since the problem may be reduced to a minimum cost network flow [1]. Unfortunately, this approach does not solve the degeneracy problem, as illustrated with Figure 3. Here, it can be seen that all solutions minimizing $\ell_1(x)$ have $\ell_\infty(x) = 1$ (the minimum), which means this solution is not unique.

ℓ_2 Regularization: A second approach to solving the degeneracy problem of (10) can be found by using an idea similar to the one presented above. We search for the solution which has the minimum ℓ_∞ value and minimizes ℓ_2 . If we rewrite this as an optimization problem:

$$\begin{aligned} \min \quad & x^T A^T C A x, \\ \text{s.t.} \quad & x_F = 1, \\ & x_B = 0, \\ & w_{ij} |x_i - x_j| \leq \rho, \quad \forall e_{ij} \in E, \end{aligned} \tag{19}$$

which is a quadratic programming problem. Note that since we assume that the graph G is connected, the addition of boundary conditions $x_F = 1$ and $x_B = 0$ cause $A^T C A$ to

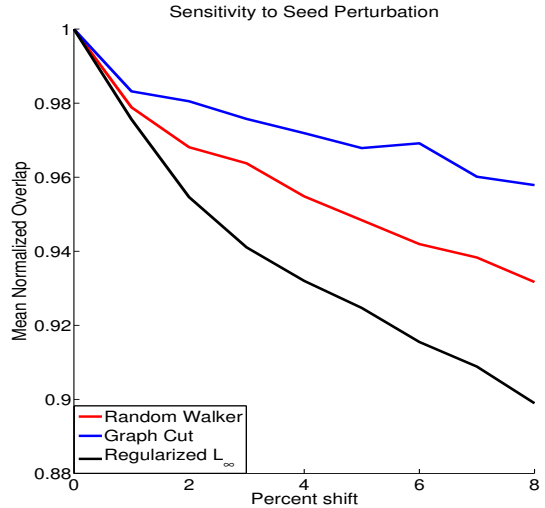


Figure 6. Sensitivity analysis of segmentation to the placement of seeds. In this case, the regularized ℓ_∞ algorithm is the least robust to seed location within an object, while Graph Cuts is the most stable. However, note that the difference in stability in this experiment is much less than the difference in stability exhibited in the seed quantity experiment of Figure 5.

be positive definite [6]. If we assume that there exists two different optimal solutions, x_1 and x_2 , then this implies that every $x = \theta x_1 + (1 - \theta)x_2, \forall \theta \in (0, 1)$ is also an optimal solution. Such a situation would imply that $A^T C A$ is still singular with the foreground and background constraints, which is a contradiction. Therefore the solution obtained by ℓ_2 regularization is indeed unique. This ℓ_2 regularization is used to generate all segmentation results for the ℓ_∞ algorithm in the next section.

In order to solve the quadratic programming problem of (19), we used a conjugate gradient based active set method [11]. All results for ℓ_2 regularized ℓ_∞ minimization problem are obtained using this solver.

3. Results

In this section, we compare the segmentations obtained for all three seeded image segmentation algorithms discussed in this paper: Graph Cuts (ℓ_1 minimization), Random Walker (ℓ_2 minimization), and ℓ_∞ minimization with ℓ_2 regularization. In all images, β is taken as 50^2 for the edge weight computation given in (5). We begin by quantitatively comparing the stability of the three algorithms to seed quantity and seed location. We then compare the segmentations obtained from these algorithms on real images and draw qualitative comparisons.

3.1. Quantitative Validation

An interactive segmentation algorithm should have two properties: 1) The segmentation should not require an excess number of seeds, since the number of seeds is roughly proportional to the amount of work required by a user (or prior knowledge of a preprocessor), 2) The segmentation should not be overly sensitive to where the seeds are placed within an object.

In this section we compare the performance of Algorithm A for various choices of q with respect to seed quantity and placement. For the tests, we placed foreground and background seeds on a set of 28 images such that roughly the same segmentation was obtained for all three choices of q (using the same seeds and graph weights for all three algorithms).

The sensitivity of each algorithm with respect to the number of seeds was tested with the following procedure: Given the initial segmentation, we filled the foreground and background with foreground/background seeds respectively. Both the foreground and background seeds were then progressively eroded. After each erosion, the segmentation was recomputed and compared to the original segmentation. The process terminated when the erosion operation removed all of the foreground or background seeds. This procedure produces seeds that progressively resemble a skeletonization of the object/background.

In order to evaluate the sensitivity of each algorithm to the effects of seed placement within an object, we adopted the following procedure: Given the initial segmentation, the locations of the seeds were shifted as a group in a random direction with random magnitude. The variance of the magnitude ranged from 1% to 8% of the image size. Any sample in which the perturbation caused the foreground (background) seeds to enter the background (foreground) was rejected and then retried until a valid perturbation was obtained. For each magnitude variance, we took 40 samples in which 20 samples shifted the foreground seeds and 20 samples shifted the background seeds.

The similarity between the initial and perturbed segmentations were measured using a normalized overlap

$$\text{Overlap} = \frac{|A \cap B|}{|A \cup B|} \quad (20)$$

where A and B are sets including the pixels within the object for two different segmentations.

The results for segmentation sensitivity to seed quantity are given in Figure 5. It is not surprising that Graph Cuts exhibits the least robustness to seed quantity, since a small number of seeds will result in a smallest cut that encompasses only the seeds. Unfortunately, in practice, the number of seeds placed by a user is likely to be closer to 10%, which still corresponds to seeding one in every ten pixels in the object. In contrast, the ℓ_∞ algorithm exhibits the

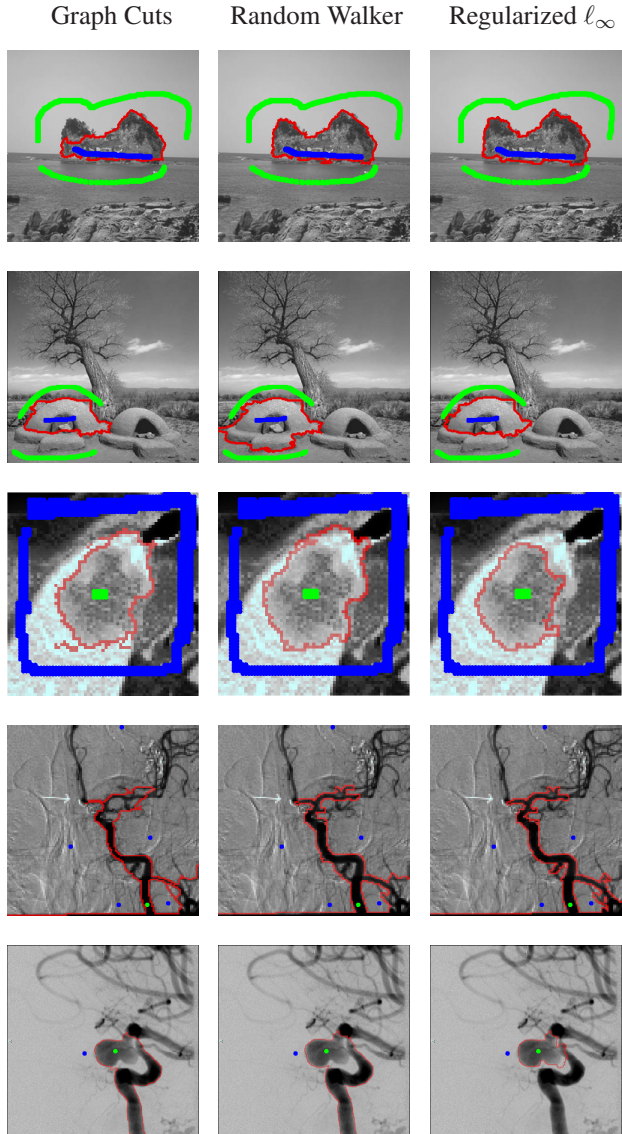


Figure 7. Segmentation results on real images using an 8-connected lattice. The segmentation boundary is outlined in red. Edge weights were equal for all algorithms in these experiments. Common problems associated with Graph Cuts are the tendency to “shortcut” the boundary (as in the top image) and a tendency to be overly committed to high contrast boundaries (as in the bottom image). A common problem associated with the Random Walker algorithm is the tendency to produce segmentations that are overly likely to provide balance between the foreground and background, which may cause the foreground segment to be too large. In contrast, the ℓ_∞ solution does not suffer from the “small cuts” (shortcut) problem, nor is it overly committed to balanced partitions.

greatest robustness to the number of seeds, while the Random Walker algorithm falls somewhere in the middle. One might conclude from these results that Algorithm A exhibits greater robustness to the number of seeds as q increases.

The results for segmentation sensitivity to seed location

within the object are given in Figure 6. In this test, Graph Cuts demonstrates the greatest robustness to seed placement within an object. This result is not surprising, since one might expect that the cost of the object boundary is generally lower than internal boundaries and therefore the location of the source/sink within the object should not have any effect. In contrast, the ℓ_∞ algorithm exhibits the least robustness to seed placement within an object. This effect may be explained by the inherent dependence of the ℓ_∞ solution on the distance from the given seeds. As before, the Random Walker algorithm exhibits a level of robustness to seed placement in between the other two algorithms, leading to the conjecture that Algorithm A exhibits less robustness to seed placement within an object as q increases. We note that the overall sensitivity of all three algorithms to seed placement (roughly 10%) is much less than the overall sensitivity of all three algorithms to seed quantity (roughly 60%).

3.2. Qualitative Validation

In this section, we employ all three algorithms to segment real images. The results of these segmentations are displayed in Figure 7. Qualitatively, the new ℓ_∞ algorithm produces “tighter” segmentations than the Random Walker algorithm. Additionally, the ℓ_∞ algorithm does not “short-cut” objects in the manner exhibited by Graph Cuts, as seen in the top image in Figure 7.

4. Conclusion

In this work, we presented a general seeded image segmentation algorithm called “Algorithm A”, which is based on the minimization of ℓ_q norms. We showed that two popular seeded image segmentation algorithms, Graph Cuts and Random Walker, correspond to the parameter choices of $q = 1$ and $q = 2$ in Algorithm A. We then examined the segmentation algorithm defined by choosing $q = \infty$. Although the solution of this problem is degenerate, we showed that it is possible to produce a minimal and maximal segmentation, depending on the choice of regularization. In order to solve the associated degeneracy problem, we proposed regularization with an ℓ_2 energy.

We performed two quantitative comparisons of the segmentations found by the ℓ_∞ algorithm to the Graph Cuts and Random Walker segmentation algorithms. These comparisons measured the relative robustness of the segmentation results to seed quantity and seed location within the object. The new ℓ_∞ algorithm exhibited the greatest robustness to seed quantity, but the least robustness to seed placement. However, the three algorithms had more similar levels of seed placement robustness (within 10% at the highest level) than seed quantity robustness (within 40% at the highest level). Based on these results, one could spec-

ulate that the robustness of Algorithm A to seed quantity increases with increased value of q , while the robustness of Algorithm A to seed placement decreases with increased q .

A qualitative comparison of the image segmentation results was also performed to contrast the relative segmentations of the various algorithms. The new algorithm was shown to find “tighter” segmentations than the Random Walker algorithm while not suffering from the problem of “shortcutting” the object that is associated with the minimum-cut criterion of the Graph Cuts algorithm.

Future work will examine specialty solvers for the ℓ_2 regularized ℓ_∞ minimization and look at the segmentations obtained from jointly minimizing combinations of the Graph Cuts, Random Walker and ℓ_∞ energy terms from the “Algorithm A” framework.

References

- [1] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993. 6
- [2] G. Aronsson, M. G. Crandall, and P. Juutinen. A tour of the theory of absolutely minimizing functions. *Bulletin of the American Mathematical Society*, 41(4):439–505, 2004. 4
- [3] Y. Boykov and G. Funka-Lea. Graph cuts and efficient n-d image segmentation. *International Journal of Computer Vision*, 70(2):109–131, 2006. 1, 2, 3
- [4] 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*, 60:233–260, 1998. 1
- [5] L. Grady. Multilabel random walker image segmentation using prior models. In *Proc. CVPR 2005*, volume 1, pages 763–770, San Diego, June 2005. IEEE. 3
- [6] L. Grady. Random walks for image segmentation. *IEEE Trans. on PAMI*, 28(11):1768–1783, Nov. 2006. 1, 2, 3, 4, 6
- [7] L. Grady and E. L. Schwartz. Isoperimetric graph partitioning for image segmentation. *IEEE Trans. on PAMI*, 28(3):469–475, March 2006. 3
- [8] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1987. 1
- [9] C. Mattiussi. The finite volume, finite element and finite difference methods as numerical methods for physical field problems. In *Advances in Imaging and Electron Physics*, pages 1–146. Academic Press Inc., April 2000. 3
- [10] E. Mortensen and W. Barrett. Interactive segmentation with intelligent scissors. *Graphical Models in Image Processing*, 60(5):349–384, 1998. 1
- [11] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, New York, NY, 1999. 6
- [12] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover, 1998. 3
- [13] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999. 1
- [14] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on PAMI*, 22(8):888–905, Aug. 2000. 3