Spring 2011

# Normalized Cut Approximations

William Stonewall Monroe
*University of Iowa*

# NORMALIZED CUT APPROXIMATIONS

by

William Stonewall Monroe

A thesis submitted in partial fulfillment
of the requirements for the Master of
Science degree in Electrical and Computer
Engineering in the Graduate College of
The University of Iowa

May 2011

Thesis Supervisor: Associate Professor Xiaodong Wu

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

_____

MASTER'S THESIS

_____

This is to certify that the Master's thesis of

William Stonewall Monroe

has been approved by the Examining Committee for the
thesis requirement for the Master of Science degree in
Electrical and Computer Engineering at the May 2011 graduation.

Thesis committee: _____
Xiaodong Wu, Thesis Supervisor

_____
Milan Sonka

_____
Mona Garvin

ACKNOWLEDGEMENTS

A special thanks to Professor Xiaodong Wu for his guidance, support, and insight throughout my research work at the University of Iowa. Also, thanks to Professor Milan Sonka for his teaching and instigating my interest in image processing and algorithm science.

ABSTRACT

Image segmentation is an important task in computer vision and understanding. Graph Cuts have been shown to be useful in image segmentation problems. Using a criterion for segmentation optimality, they can obtain segmentation without relying heavily on *a priori* information regarding the specific type of object. Discussed here are a few approximations to the Normalized Cut criterion, the solving of which has been shown to be an NP-hard problem. Two Normalized Cut algorithms have been previously proposed, and a third is proposed here which accomplishes approximation by a similar method as one of the previous algorithms. It is also more efficient than either of the previously proposed Normalized Cut approximations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

# CHAPTER I: INTRODUCTION

Image segmentation is, simply put, the process of finding objects in pictures. Image segmentation tasks can range from being as straightforward as drawing an outline around a car in a traffic camera photograph, or can be as complex as deriving a three dimensional (3D) model of a liver from a computed tomography (CT) scan. While the human vision process functions in such a way as to make it relatively easy to decipher the existence of objects and information regarding these objects from the field of perception, there is a desire to have computers reach similar levels of visual understanding.

Human segmentation of images is slow and lacks robustness. Even though human vision can understand images quickly, the process of actually delineating the borders of an object by hand is arduous. Computational techniques for computer vision suffer from the opposite problem, however. Given an algorithm for image segmentation, a computer can always reproduce its own segmentation. Such automated, or semi-automated as the case may be, analysis of images is done within the computational framework, and creates robustness by calculating a segmentation based off of measurable properties of a given image. When it comes to object recognition, not simply delineation of a possible object, computational methods do not always achieve the desired results, as identifying objects from measureable features can be difficult. The human identification of images can be seen as being linked to a hierarchical interpretation of the received visual data [1]. This hierarchy leads to an interpretation of minute details within the context of these details as members of a larger whole. For example, a chair behind a desk can be recognized as such by a human easily, knowledge of the context and meaning of visual information enables the object recognition. In seeking to do this, algorithms can be developed in a few ways.

The goal of image segmentation is to bring understanding to bear in the computational context. Some methods seek to give the computer large amounts of *a*

*priori* knowledge regarding its assigned task. Many anatomical image segmentation methods use techniques in this category. For example, determining liver boundaries by taking a pre-existing liver model and manipulating it to fit actual patient data. Such algorithms are generally preferred when available, because using this knowledge for certain types of images, the robustness of computer vision can be combined with the understanding of human vision. This is particularly beneficial in computer vision applications which search for segmentations of well understood objects [2].

Other methods, such as the ones to be discussed here, use very little or no *a priori* knowledge of the given segmentation problem. These algorithms seek to obtain the solution to the segmentation problem with as little human interaction as possible. In the case of the algorithms presented here, the need for previous knowledge of a specific subset of object types within certain imaging applications is attempted to be replaced with computational definitions of an object. Thus the method can seek to find a numerically defined optimal solution for object segmentation. This type of segmentation can be particularly useful when the desired object is ill defined, or varies from image to image. In the case of tumor segmentation in CT data, tumors rarely appear in the same shape in two separate patients. It is vital that a valid segmentation can be obtained for the purposes of treatment planning. A computational technique, which could provide a decent segmentation in such situations, would be desirable.

The algorithms presented here are within a subset of image processing algorithms called "graph-cut" algorithms. Given a network of weighted edges and connecting nodes, a graph cut is seen as being division in the graph between two regions, where the capacity of the cut is given by the sum of edges intersected by the cut. The most widely used graph-cut algorithms are for deriving the "minimum-cut." Minimum-cut (also known as maximum flow) algorithms find the smallest capacity cut that divides the entire graph in two regions. In order to use graph cuts in image segmentation, images must be transformed into graphs. This is done by choosing the pixels, or voxels in 3D, as nodes,

and connecting these nodes with edges. The capacity of the edges is derived from the similarity between nodes. While solving the minimum-cut problem can in many situations produce segmentations of high quality, it will also frequently delineate regions of the input image too small for practical use [3,4]. One proposed modification of the minimum-cut problem to resolve this difficulty is called "Normalized Cut." A normalized cut is one that seeks to create a cut, which divides the input graph into "object-like" regions. Such a cut minimizes the capacity across the cut, while maximizing the capacity within the regions created by the cut.

Within the proposal by Shi and Malik, solving the normalized cut criterion was found to be an NP-hard problem [4]. Therefore, if this criterion is to be used, it is necessary to use approximations to the Normalized Cut rather than Normalized Cut itself. Otherwise, a given solution algorithm would not be efficient enough to merit its use. Thus far, there have been two proposed approximations of the Normalized Cut criterion. One has simply been called Normalized Cut, and the other has been referred to as Normalized Cut'. The Normalized Cut algorithm finds an approximation of the entire Normalized Cut criterion through the use of eigenvalues of a similarity matrix. The Normalized Cut' algorithm finds an approximation of the Normalized Cut by minimizing only one piece of the Normalized Cut equation. Here a third approximation is proposed; it operates in much the same way as the previous Normalized Cut' algorithm, in that it seeks to minimize one term of the Normalized Cut criterion.

In [5], Normalized Cut and Normalized Cut' were compared upon the objective standard of having the closest value to the "optimal" when the results were used to calculate the normalized cut criterion. Even though Normalized Cut approximates the entire normalized cut criterion and Normalized Cut' solves for a variant of the criterion, it was found that Normalized Cut' actually arrived at a better result of the normalized cut objective. The Normalized Cut algorithm has been evaluated as part of a study in creating an image segmentation benchmark. This study sought to use a large database of images

with reasonably clear segmentations as a benchmark of quality in segmentation. This paper seeks to use this segmentation benchmark to evaluate both Normalized Cut and Normalized Cut' over graphs with varying input parameters. In addition, the analysis has been extended to incorporate some precautions taken to insure that the algorithms are run on an even playing field, being given the same input graphs and same platform. The proposal and analysis of the new Normalized Cut' algorithm focuses primarily on run-time. Theoretically, it should provide a similar segmentation as the previous Normalized Cut' algorithm.

Thus far, the first two of these algorithms have been individually evaluated and analyzed comparatively. All of the algorithms here have been found to be of less complexity than the Normalized Cut problem itself. Also, a modified Normalized Cut' algorithm for approximating a Normalized Cut efficiently is proposed here first. An in depth study of these algorithms side by side has not as of yet been accomplished. Such a study would include an analysis of their theoretical advantages and disadvantages as well as their effectiveness and efficiency in computing image segmentations. This thesis has sought to accomplish this study and present the results.

# CHAPTER II: METHODS

For this study, the Normalized Cut and two methods for Normalized Cut' were used.

## Graph Cuts

A Normalized Cut is an extension of using graph partitioning for the purposes of segmentation, or grouping [4]. Graph partitioning can be done by grouping a graph **G(V,E)** into two disjoint sets of vertices. The grouping is based upon the dissimilarity between the two pieces. This dissimilarity is quantified by a graph cut,

(1)

$$cut(S,T) = \sum_{i \in S, j \in T} w(i,j),$$

where *S* and *T* are the two regions of the cut, *i* and *j* are nodes in the graph belonging to *S* and *T* respectively, and *w* is the capacity, or weight, given to an edge. The *cut* measure then, is the sum of capacities between the two regions under consideration. Under only the minimum cut measure, the optimal partitioning of the graph minimizes this cut across the whole graph. Such a cut maximizes the dissimilarity between the partitions. Minimum cuts are used in a variety of applications, and its algorithmic solution and efficient implementation have been found and researched thoroughly. The segmentations based off of this approach can be quite good, especially when used with an additional optimality criterion. However, the results of these cuts are generally highly dependent on the chosen energy functions. The energy function being the measure of similarity, or dissimilarity, between any given pair of vertices (the E in G(V,E)). The other limitation of this minimum cut criterion is that it tends to find small regions [3,4].

The reason for this being that having a strictly minimum cut tends to minimize the number of edges across the cut as well.

## Normalized Cut

Normalized cut was proposed to deal with this limitation of minimum cut. As shown in equation (1) [4],

(2)

$$Ncut(S,T) = \frac{cut(S,T)}{assoc(S,V)} + \frac{cut(S,T)}{assoc(T,V)}$$

where *Ncut(S,T)* is the measure of normalized cut between the regions *S* and *T,* and *cut(S,T)* is the value of (1) for the two regions *S* and *T*, and *assoc(S,V)* for a subset of nodes *S* and all nodes in the graph *V,* is defined as,

(3)

$$assoc(S,V) = \sum_{i \in S, j \in V} w(i,j).$$

The goal, is then to minimize not simply the *cut(S,T),* but to minimize the *Ncut(S,T)*. Minimizing the *Ncut* measure leads to a cut with similarity across a cut minimized, while in conjunction maximizing the similarity within the regions on either side of the cut, thus encouraging larger cuts [4]. As solving for this directly is an NP-hard problem [4], it is necessary to approximate the criterion if it is to be used reasonably. Shi and Malik derived a method using the eigenvectors of the similarity matrix to approximate the Normalized Cut. For the purposes here, the entire derivation will not be recreated, but some highlights will be noted to provide context.

A generalized eigenvalue system, the second smallest eigenvector of which is then a real-valued solution to the Normalized Cut approximation problem, can be

constructed from the nodes and edges in the graph. This eigenvalue system is represented as,

(4)

$$(\mathbf{D} - \mathbf{W})y = \lambda \mathbf{D}y.$$

$\lambda$ is the eigenvalues of the system. Let *N* equal the total number of vertices in the graph. **W** is the similarity matrix of the graph; it is a symmetrical *N* x *N* matrix where W(i,j) = $w_{ij}$.

(4) also requires the definition of some associated vectors, ***x*** and ***d.*** The ***x*** vector is *N* dimensional, where

$$\begin{cases} x_i = 1, & i \in S \\ x_i = -1, & i \notin S \end{cases}.$$

The ***d*** vector is also an *N* dimensional vector. It is defined such that

$$d_i = \sum_j w_{ij}\,.$$

It is the sum of all the connections between the node *i* and the rest of the graph. **D** is an *N* x *N* diagonal matrix with D(i,i) = $d_i$. That is, ***d*** corresponds to the diagonal of **D**.

$$y = (1 + x) - b(1 - x).$$

Where $b = \frac{k}{1-k'}$ and $k = \frac{\sum_{x_i>0} \boldsymbol{d_i}}{\sum_i d_i}$ [4]. With this generalized eigenvalue system, Shi and Malik [4] derive that the solution to the normalized cut approximation problem is the second smallest eigenvector of this system. The Normalized Cut algorithm then consists of four steps [4].

1. Create a graph G(V,E) from the input image using the similarity information from the image to derive the capacity between adjacent nodes.
2. Solve the system in (4) for its eigenvalues and eigenvectors
3. Use the second smallest eigenvector to obtain a bipartition of the graph.

4. Recursively call steps 1-3 on the partitions if further segmentation is desired.

The run-time of the Normalized Cut algorithm is largely dependent on the computation time of finding the eigenvectors for the similarity matrix. For a general eigenvalue problem, solving for all eigenvectors runs in $O(n^3)$ time, where $n$ is the number of pixels in the image (nodes in the graph). However, in practical image segmentation problems, similarity matrices are fairly sparse. For eigenvalue solvers taking advantage of this sparsity, this run-time can be greatly improved in the application of this algorithm [4].

## Normalized Cut'

The Normalized Cut' variant of the Normalized Cut criterion seeks to minimize a single term Normalized Cut equation. The original Normalized Cut criterion requires the minimization of the sum of two ratios. In the following algorithms, it is noted that the minimization of either term in the equation can lead to an approximation of the entire criterion [5]. A single term being,

$$\frac{cut(S,T)}{assoc(S,V)}.$$

Two algorithms for accomplishing a Normalized Cut' on an image are presented here.

### Hochbaum's Approach

Hochbaum's algorithm for minimizing the Normalized Cut' adapts the parametric maximum flow (minimum cut) problem [7, 8]. It uses a variant proposed by Sharon, *et al*, which when minimized also minimizes a single term of *Ncut* (2) [1]. This variant is the one referred to as Normalized Cut',

(5)

$$Ncut'(S,T) = \frac{cut(S,T)}{assoc(S,S)}.$$

Note that this is not actually the same as either term in *Ncut* (2); however, the proof is as follows,

$$Ncut'(S,T) = \frac{cut(S,T)}{assoc(S,V) - cut(S,T)}$$

$$= \frac{1}{\frac{assoc(S,V)}{cut(S,T)} - 1}.$$

Minimizing *Ncut',* then, requires $\frac{assoc(S,V)}{cut(S,T)}$ to be maximized. Maximizing the ratio will also minimize its reciprocal, $\frac{cut(S,T)}{assoc(S,V)}$, which is a single term of *Ncut.*

Therefore, use of the Normalized Cut' criterion requires that the *Ncut'* value be minimized over the given weighted graph. Such a minimization problem can be rewritten as a series of calls to answer a yes or no $\lambda - question$ [7]. This question for the *Ncut'* problem is,

Does there exist a subset $V' \subset V$, such that $cut(S,T) - \lambda * assoc(S,S) < 0$? Then an algorithm can search for a suitable $\lambda$. A "yes" answer implies that the optimal $\lambda$ value is less than the current $\lambda$. A "no" answer implies that the optimal value is either equal to or greater than the current $\lambda$. This series of calls is then terminated with an optimal solution when the resulting value of the $\lambda - question$ is equal to zero. This could be solved iteratively using a binary search method. The method presented here, however uses a parametric procedure.

A more explicit version of the $\lambda - question$ for *Ncut'* can be written as,

$$\sum_{i,j \in V} w_{ij}\, z_{ij} - \sum_{i,j \in V} \lambda w_{ij}\, y_{ij} < 0,$$

where $z_{ij}$=1 if $i \in S$ and $j \notin S$, and $y_{ij}$=1 if $i \in S$ and $j \in S$. In this formation, the *Ncut'* problem can be solved through the use of a minimum cut on a modified graph.

A general graph *G* can be constructed from an image *I*. First, the image, *I* is modeled as a matrix, in our case of two dimensions, *I(x,y)*; each position in this matrix is taken to be a vertex in the graph. Then, a neighborhood, *N,* must be used during the instantiation of the adjacencies. Generally, a neighborhood is taken as being all of the pixels within a Euclidean distance less than a radius, *r*, to a given center pixel of the neighborhood. All vertices related by the neighborhood are considered to be adjacent in the resulting graph. A directed edge is connected from the center vertex of the neighborhood to all adjacent vertices. Each edge has a capacity related to the similarity between the two vertices. This similarity can be measured using the intensity information given in the image. The costs used were the same as in [4].

Using the gadget in Figure 1, a graph *G'* is constructed based on *G*. *G'* has the nodes, *V',* and edges, *E'*. From a directed graph *G,* additional nodes and edges are added as in the gadget. A single node is added for every adjacency $w_{ij}$. This node, $y_{ij}$, is connected to nodes *i* and *j* by edges of infinite capacity. The additional node is then connected the source with an edge of capacity, $\lambda w'_{ij}$. Source seeds are created by sinking edges with the source through the use of an infinite capacity edge between the source and corresponding added node, $y_{ij}$. Sink seeds are added by sinking desired original graph nodes with the sink with and edge of infinite capacity. A sample transformation from a sample *G* to a *G'* graph is shown in Figure 2.

A bipartition of this graph corresponds to a feasible solution of the $\lambda - question$ for *Ncut'* [7]. This can be proved through a look at the capacity of a cut in such a graph *G'* (Figure 2). Since it is noted that when the edge $w_{ij}$ is included in *T', T'* must also include the node $y_{ij}$. A graph like *G'* leads to a cut with capacity defined by *cut(S',T').*

$$cut(S', T') = \sum_{y_{ij} \in T' \cap V_y} \lambda w'_{ij} + \sum_{i \in V_x \cap S', j \in V_x \cap T'} w_{ij},$$

which can be altered by observing that the complement of the first term is the summation of edges in *S'* connecting the source to the graph.

$$cut(S', T') = \sum_{v \in V_y} \lambda w'_{v} - \sum_{y_{ij} \in S' \cap V_y} \lambda w'_{ij} + \sum_{i \in V_x \cap S', j \in V_x \cap T'} w_{ij}$$

This is the same as a constant, $\lambda W'$, where *W'* is equivalent to the sum of edges connecting the source to the graph and the value of a feasible solution to the $\lambda - question$,

$$cut(S', T') = \lambda W' + [\sum_{i \in V_x \cap S', j \in V_x \cap T'} w_{ij} - \sum_{y_{ij} \in S' \cap V_y} \lambda w'_{ij}].$$

In terms of the original graph *G,* this is the same as

$$cut(S', T') = \lambda W' + [cut(S, T) - \lambda * assoc(S, S)].$$

As the parameter, $\lambda$, increases, the capacity of source-adjacent arcs increase monotonically. The change of this parameter causes breakpoints in the graph to occur, where a breakpoint is the value at which the source set is changed by even one node due to this input capacity change. Thus, each breakpoint corresponds to a possible solution, and these are nested source sets. Letting the breakpoint values equal $\lambda_1 < \lambda_2 < \cdots < \lambda_l$ and the corresponding source sets, $S_1 \subset S_2 \subset \cdots \subset S_l$. These $\lambda$ and source set values are then used to compute the series of calls to the $\lambda - question$ and also minimize (5).

The values are searched for the first $\lambda$ value, which provides a *yes* answer to the $\lambda - question$, the corresponding source set and complement of that source set are returned as the bipartition of the graph.

A parametric maximum flow algorithm is used to find all of the breakpoints in a graph and thus possible solutions. Hochbaum used the pseudoflow algorithm to solve the parametric maximum flow problem [14]. In order to further test the algorithm, the parametric maximum flow formulation was used here [8,13]. The algorithm results in a number of possible segmentations corresponding to the breakpoints in the input graph. These possible solutions are used as the possible bipartitions and the breakpoints as values for $\lambda$. Hochbaum's Normalized Cut' algorithm consists of 4 steps.

1. Create a graph G(V,E) from the input image using similarity information.
2. Create the modified graph G'(V',E') by identifying source and sink seeds, and adding the edges and nodes as specified by the gadget in Figure 1.
3. Run a parametric maximum flow/minimum cut solver on the graph *G'*.
4. Search the breakpoints for the first $\lambda$ value that gives a yes answer to the $\lambda - question$ for *Ncut'* and return the corresponding minimal source set as the bipartition of *G*, transforming *G* back into an image for further use.

The run-time of this algorithm is highly dependent on the parametric maximal flow problem, which, depending on the maximal flow algorithm used, has a $O(nmlog(n^2/m))$ bound, where *n* is the number of nodes in the graph, and *m* is the number of adjacencies in the graph. With the addition of the gadget in Figure 1, it can be seen that the number of nodes in the *G'* used for *Ncut'* is $m'=n+m$, or $O(m)$. Since the number

of adjacencies is only multiplied by a constant factor, it remains *O(m)*. Therefore, the runtime on a general graph is $O(m^2 log(m))$.

## The Proposed Approach

A novel algorithm proposed here approximates the Normalized Cut criterion via the Normalized Cut' variant. This algorithm is a modification of Hochbaum's Normalized Cut' algorithm, eliminating the need for additional nodes and edges required by the *G'* graph and gadget. However, knowledge of the source to graph weights from Normalized Cut' is used to assign source adjacent edges intelligently. This alters the objective function, though when it is minimized, is equivalent to the Normalized Cut'. The objective function for this formulation is

(6)
$$\frac{cut(S,T)}{assoc(S,S)+\frac{1}{2}cut(S,T)}.$$

It can be shown that minimizing this is accomplishes the same task as minimizing *Ncut'*; that is, when minimized, it also minimizes a single term of *Ncut* (2).

$$= \frac{cut(S,T)}{assoc(S,V)-cut(S,T)+\frac{1}{2}cut(S,T)}$$

$$= \frac{cut(S,T)}{assoc(S,V)-\frac{1}{2}cut(S,T)}$$

$$= \frac{1}{\frac{assoc(S,V)}{cut(S,T)}-\frac{1}{2}}.$$

This demonstrates that a minimization of this new formulation also leads to the maximization of a reciprocal of a single term of *Ncut*, $\frac{assoc(S,V)}{cut(S,T)}$, which therefore also

minimizes a single term of *Ncut*. Again, the minimization of a ratio can be written to a series of calls to a $\lambda - question.$

Does there exist a subset $V'' \subset V,$ such that

$$cut(S,T) - \lambda * [assoc(S,S) + \frac{1}{2}cut(S,T)] < 0?$$

This can be more explicitly be written with the edge weights as

$$\sum_{i\in S, j\in T} w_{ij} - \lambda[\sum_{i,j\in S} w_{ij} + \sum_{i\in S, j\in T} \frac{1}{2}w_{ij}] < 0.$$

The reasoning behind the modification is that for a graph *G'* as in Figure 2, created from a graph *G* with the gadget (Figure 1) proposed in Hochbaum's algorithm adds a large number of nodes and edges to compute the desired criterion. This of course increases run-time and complexity of the parametric maximal flow problem in need of being solved.

Again starting with an arbitrary graph *G* constructed in the same method as discussed in the Hochbaum approach, a graph *G''* with nodes, *V'',* and edges, *E''*, can be constructed. The set of nodes is not altered in this process, however, edges from the source to each node, *i,* are added, letting each edge $w'_i$ in *G''* be equivalent to $\sum_{j\in N} \frac{1}{2} w'_{ij}$ where *N* is the neighborhood of *i.* A sample *G''* constructed from a sample *G* is shown in Figure 4. This can also be seen as a reduction of a graph like *G'* (Figure 2) to be *G''* (Figure 4), when the gadget in Figure 1 is altered to be Figure 3.

It is evident that the parametric maximal flow algorithm would run more efficiently on *G''*, it is not necessarily intuitive that this reduction still leads to the desired minimization of the single term of *Ncut,* as it is also clear that certain observations which lead to the $\lambda - question$ for Hochbaum's approach being computed do not hold in *G''*.

Specifically that in *G'*, if an edge $w_{ij}$ is included in the cut, its corresponding edge $w'_{ij}$ is not included, with the graph *G''* there is no such guarantee. In fact, using the definition of $w'_i$ used here, that information is assuredly part of the cut value. However, it can be shown that by using this modification of *Ncut',* the criterion is minimized and therefore accomplishes the same task as Hochbaum's Normalized Cut' algorithm with a more efficient run-time. A graph like *G''* leads to a cut with capacity defined by *cut(S'',T'').*

$$cut(S'', T'') = \sum_{i \in T''} \lambda w'_i + \sum_{i \in S'', j \in T''} w_{ij}.$$

Using the definition of $w'_i$ this can be rewritten as,

$$cut(S'', T'') = \sum_{i,j \in T''} \lambda w'_{ij} + \sum_{i \in S'', j \in T''} \tfrac{\lambda}{2} w'_{ij} + \sum_{i \in S'', j \in T''} w_{ij}$$

Then through some similar observations as in the previous proof, and allowing $w'_{ij} = w_{ij}$

$$cut(S'', T'') = \sum_{i,j \in T''} \lambda w_{ij} + \sum_{i \in S'', j \in T''} \frac{\lambda}{2} w_{ij} + \sum_{i \in S'', j \in T''} w_{ij}$$

$$= \sum_{i,j \in V''} \lambda w_{ij} - \sum_{i,j \in S''} \lambda w_{ij} - \sum_{i \in S'', j \in T''} \lambda w_{ij} + \sum_{i \in S'', j \in T''} \frac{\lambda}{2} w_{ij} + \sum_{i \in S'', j \in T''} w_{ij}$$

$$= \lambda W' + \sum_{i \in S'', j \in T''} w_{ij} - \lambda [\sum_{i,j \in S''} w_{ij} + \sum_{i \in S'', j \in T''} \frac{1}{2} w_{ij}]$$

Which is equal to the same constant $\lambda W'$ as in the Normalized Cut' algorithm added to the value of a feasible answer to the $\lambda - question$ for this novel approach. In terms of the original graph,

$$cut(S'', T'') = \lambda W' + [cut(S, T) - \lambda * (assoc(S, S) + \frac{1}{2} cut(S, T)].$$

Through the same breakpoint searching procedure as in the Normalized Cut' algorithm, the bipartition from the new algorithm is found. The algorithm consists of 4 steps.

1. Create a graph G(V,E) from the input image using similarity information.
2. Create the modified graph G''(V'',E'') by identifying source and sink seeds, and intelligently assigning the source adjacent arcs as in Figure 3.
3. Run a parametric maximum flow/minimum cut solver on the graph *G''*.
4. Search the breakpoints for the first $\lambda$ value that gives a yes answer to the $\lambda - question$ and return the corresponding minimal source set as the bipartition of *G*, transforming *G* back into an image for further use.

With the elimination of the additional node required by the exact *Ncut'* formulation of the problem, the run-time is exactly that of a parametric maximal flow problem. Thus, the run-time is $O(mn\log(n^2/m))$, where *n* is the number of nodes in the graph, and *m* is the number of adjacencies in the graph.

# CHAPTER III: EXPERIMENTATION

Algorithms were implemented and experiments were run on a Macintosh Macbook with a 2.4 GHz Intel Core 2 Duo processor and 4 GB 1067 MHz DDR3 memory.

## Data

Most of the data for the experiments performed is the same as in [6]. This is a database of 1023 images selected from "real natural images from [the] internet, digital photos, and some well known image databases such as Corel" [6]. Each image has the ground truth defined for comparison to segmentations. The images were chosen for the clarity of foreground/background relationship, and the ground truth was collaboratively defined by a team of computer-science students. The subjectivity of human segmentation of images for the ground truth is then limited by the need for agreement among multiple segmenters, and with such a large volume of test data, trends in segmentation quality can still be observed, even in the presence of a "human variance" of ground truth quality. All images are fairly small, between the sizes of 80 x 80 and 200 x 200, to facilitate reasonable computation time of segmentations of the entire database. Some additional test images of a larger size were fabricated to illustrate the run-time differences amongst the algorithms. Images created had clear foreground and homogeneous background.

## Implementations

### Normalized Cut

The implementation of the Normalized Cut algorithm used is the version one written by Shi and Malik [4]. It is based in the MATLAB platform for easy scripting whereas a bulk of the computational work is handled by C implementations through MATLABs mexFunction interface for using C programs.

## Normalized Cut'

### Hochbaum's Algorithm

Hochbaum's Normalized Cut' algorithm was implemented for the purposes of this comparative study. Instead of using the pseudoflow algorithm as the basis for the implementation, as in [7], the parametric maximal flow algorithm used here was the one implemented and used in [8]. This C code was modified to be used in MATLAB using the mexFunction interface, as in the Normalized Cut code, so that the algorithms would be working on the same platform.

### The Proposed Approach

The implementation of the novel approach is the same as in Hochbaum's Normalized Cut', but with the graph modification applied. The parametric maximal flow was used with the mexFunction interface.

## Graph Construction

In order to ensure a fair comparison, both algorithms need to be run on the same input graphs. For this purpose, graph creation was done separately from the computation done by the algorithms. For 2D image analysis, the graph creation used by Shi and Malik's code was mostly adequate. Their code creates an algorithm based on a given radius and sampling rate. This allowed for graphs using eight-neighborhood, and neighborhoods of arbitrary radius greater than two. The sampling rate helps to cut run-time, as Shi and Malik noted that the result of the algorithm was largely the same, even when a random sampling of edges were deleted from the graph.

## Tests

A series of tests using the segmentation benchmark data were constructed. These tests can be seen in (Table 1). The primary parameters changed in the tests were the graph related parameters that directly affect both algorithms. Graph neighborhood size

was changed and run for both algorithms in the range from 8-Neighborhood to a radius of 3, with the introduction of sampling as the radius grew larger. Normalized Cut was run alone on some larger graph sizes to see more clearly how graph density affected its segmentation quality. Also, even though Normalized Cut can be recursively called to find multiple segmentations, we only call the algorithm once, to find a bi-partition of the images. This is because since Normalized Cut' finds a foreground and background, in order for the comparison to be a fair one, both algorithms need to be searching for the same type of segmentation.

## Metrics

The metric used in the tests run to determine quality of segmentation is known as the Jaccard Coefficient [7]. This measure gives no preference to size of output segmentation, and simply computes the coinciding of segmentation with the ground truth. This is defined as

(7)

$$J(R;A) = \frac{|R \cap A|}{|R \cup A|} = \frac{|R \cap A|}{|R| + |A| - |R \cap A|}$$

(7) computes the area of the intersection between the segmentation region and the ground truth divided by the union of the segmentation region and the ground truth. The Jaccard Coefficient is also equivalent to the Dice Coefficient [12], another measure of segmentation quality. The Dice Coefficient is defined as

(8)

$$D(R;A) = \frac{2|R \cap A|}{|R| + |A|}.$$

The Dice Coefficient can be found from the Jaccard Coefficient as follows,

(9)

$$D = \frac{2J}{1 + J}$$

These coefficients consist of values between zero and one, with a value of one being perfect matching between the two regions being compared, and a value of zero carrying a meaning that the two regions are disjoint.

In addition to the ground truth measure using the Jaccard Coefficient, we have used the Normalized Cut criterion itself as a measure of quality as well as run-time.

# CHAPTER IV: RESULTS AND DISCUSSION

It is noted that the segmentation quality of the novel approach to Normalized Cut' is theoretically very similar to that of Hochbaum's algorithm, so except for where specifically mentioned, its segmentation quality is not discussed here. It can be assumed that otherwise, observations applying to Hochbaum's Normalized Cut' algorithm apply to the novel approach as well.

## Previous Result

In Figure 5, the cumulative distribution of segmentation qualities over the segmentation benchmark database is shown. This is the result from [6], for running the Normalized Cut algorithm on the image database while only searching for a bi-partition of the image. The average quality found of segmentation using the Jaccard Coefficient was 0.3837.

## Validation of Test

A test was run to recreate the graph from [6]. Figure 6 is the cumulative distribution of this test. The average from this Jaccard Coefficient value from this test was 0.3750. This being similar to the result found in [6], and the plot in Figure 6 displaying similar properties to those Figure 5, it was determined that the results from these two tests show that the implementation of at least Normalized Cut is the same, or similar enough and outputs of similar tests were feasible to use in investigating the qualities of the two algorithms.

## Quality

Table 2 shows the average Jaccard Coefficient values from the tests run on the segmentation benchmark database. From the data in Table 2, Normalized Cut' and Normalized Cut give similar quality when run on the same graph. It would appear that relatively, Normalized Cut' tends to give slightly better segmentations when small

neighborhood sizes are used. Also, the observation from [4] that a graph constructed from a random sampling of neighborhood edges as opposed to using the full graph gives similar results appears to hold not only for Normalized Cut but also Normalized Cut'. In fact, the results here indicate that Normalized Cut' segmentation suffers less from this information loss than its counterpart. This result is visualized in Figure 7. As is noted in [6], however, merely looking at average quality does not give the full picture. A look at the cumulative distribution of qualities from the entire dataset can give deeper insight into the segmentations provided by these algorithms. In Figures 8 and 9 are the distributions from Normalized Cut and Normalized Cut' respectively for a graph with a neighborhood of size radius 3. For much of the distribution, it appears as if Normalized Cut is performing better than Normalized Cut', however, the most difference can be seen at the right hand side of both distributions. For the final 10% or so of images, Normalized Cut' has a much higher quality. So even if the average qualities of each algorithm's segmentations are similar, this indicates that the segmentations themselves are rather different. Due to the nature of the Normalized Cut algorithm, it tends to partition the image into equal halves, and so objects, which are actually small, are often not segmented correctly. On the other hand, Normalized Cut', being based on more of the minimum cut procedure directly can in some cases still tend to find small regions or suffer from bleeding if the cost function used does not create a large enough difference between the foreground and background. The overall Jaccard coefficients of segmentation for this data set are fairly low, however, this is most likely due to the general nature of these segmentations, and the naïve approach to picking source and sink seeds used.

Another criterion used to measure the quality of the respective approximations was the Normalized Cut criterion itself from (2). How well the segmentations minimize this criterion can give another, objective measure of the segmentation quality. Table 3 shows the Normalized Cut value, calculated with (2), for both Normalized Cut and

Normalized Cut' for a selected set of twenty images. The results in this table confirm the results found in [5], where it was found that Normalized Cut' algorithm actually approximated the Normalized Cut criterion better than the Normalized Cut algorithm.

## User Interaction

The primary difference in amount of user interaction is the semi-automatic nature of Normalized Cut'. The segmentation requires some user-inputted seeds for the foreground and background. This can be useful when it comes to attempting to influence the cut and can for some situations, particularly when the desired object isn't large, lead to a better segmentation by Normalized Cut'. For the purposes here a set of points at a corner of the image were taken for the source seeds and a naïve average of the points defining the ground truth was used for the sink seeds.

## Run-time

The run time of the algorithms presented here is of particular interest since the Normalized Cut problem itself is known to be NP-complete. As stated in the Methods chapter, the run time for the Normalized Cut algorithm is $O(n^3)$, the run time for Normalized Cut' using Hochbaum's algorithm is $O(m^2 log(m))$, the run time for the proposed algorithm is $O(mnlog(n^2/m))$, with $m$ equaling the number of adjacencies in the graph, and $n$ equaling the number of nodes in the graph. This means that for large input images, Hochbaum's Normalized Cut' will run faster than Normalized Cut, and the proposed approach will always run faster than Hochbaum's approach. In a practical sense, since Normalized Cut uses an Eigen solver that takes advantage of graph sparseness, it runs more efficiently in general than $O(n^3)$. Figure 10 shows the actual run time in seconds of the three algorithms as image size increases. This chart demonstrates that the actual run time seen by these methods does reflect this relationship. For relatively small images, this relationship is less clear, and in fact, with higher graph density Normalized Cut was seen to run more efficiently. Figure 11 shows this

relationship for Normalized Cut and Hochbaum's Normalized Cut'. Intuitively, one can look at the theoretical run times provided and note that Normalized Cut' relies more heavily on the neighborhood size, and in Figure 11, its run time is more heavily affected by neighborhood than in Normalized Cut. This relationship between neighborhood size and the relative efficiency of the algorithms' run times can be seen in the theory as well. The intersection of run times can be found where the theoretical run times are equivalent, where

$$n^3 = m^2 * \log(m).$$

Noting that the number of adjacencies *m* can be approximated as the number of nodes multiplied by the number of adjacencies for a single node in the middle of the graph, that is, the neighborhood size, *N*. Letting, $m \cong n * N$, we have

$$n^3 = n^2 * N^2 \log(n * N),$$

$$n = N^2 \log(nN).$$

$$e^n = e^{N^2} nN.$$

This gives

(10)

$$\frac{e^n}{n} = Ne^{N^2}.$$

Figure 12 visualizes the relationship between these terms and their contribution to run-time when *N* is held constant. This shows that Normalized Cut will run more efficiently for relatively small graph sizes compared to the neighborhood size. Figure 13 shows that the proposed method runs more efficiently than Hochbaum's method for Normalized

Cut' no matter the neighborhood size, but both are still more heavily impacted by graph density than Normalized Cut.

## Sample Results

Figures 14, 15, 16, and 17 show sample results from the use of the algorithms presented. These can be used to gain intuition about the tendencies of these algorithms. In Figure 15 a), an original image is shown, and even though it is the desired foreground, Normalized Cut, Figure 15 b), tends to bi-partition an image in two geometrically equal pieces, so since there is not a large variation in grayscale values, it is seen that the segmentation basically splits the image in half, though it does follow along the shadow at the bottom of the image, and roughly follows one side of the foreground object. Normalized Cut'; however, appears to give a more desirable segmentation. This is perhaps because even though it has similar tendencies to Normalized Cut, these are not as strong, and tempered by the use of source and sink seeds. In Figure 16, this cut balancing tendency is displayed by both segmentations. It is interesting to note that in these results, even in images with some large variance in grayscale values within the desired region, a segmentation enclosing the desired region can be achieved, the Normalizing of a cut giving way to some additional robustness in segmentation. For further robustness of segmentation, it could be beneficial to complement Normalized Cut procedures with some fuzzy connectivity information, by using synergistic arc-weight estimation [15].

# CHAPTER V: CONCLUSION

The Normalized Cut criterion is one that can measure the optimality of a cut based on a definition of what makes a good object. If this criterion is appropriately minimized, valuable image segmentations can be achieved. However, with the solution of the problem exactly being NP-hard, it is necessary to approximate its solution. The Normalized Cut approximation introduced by Shi and Malik is generally seen as a bit slow and a newer approximation, known as Normalized Cut', can be solved reasonably faster and provides for a similar quality of segmentation, and in some cases a better quality. A novel modification of the Normalized Cut' algorithm proposed here finds a similar segmentation with a much improved run time. It has been demonstrated here that a good approximation of the Normalized Cut criterion can be made in polynomial time, and with the use of the approach proposed here, this approximation is accomplished in exactly the run time of an ordinary parametric maximal flow problem.

Table 1: Tests Run

| Test Number | Radius | sampling | sampling rate | Graph | dataset |
|---|---|---|---|---|---|
| **1** | 8N | yes | 0.9 | Ncut | Segmentation Benchmark |
| **2** | 8N | no | | Ncut | Segmentation Benchmark |
| **3** | 2 | no | | Ncut | Segmentation Benchmark |
| **4** | 2 | yes | 0.5 | Ncut | Segmentation Benchmark |
| **5** | 3 | no | | Ncut | Segmentation Benchmark |
| **6** | 3 | yes | 0.5 | Ncut | Segmentation Benchmark |
| **7** | 5 | yes | 0.5 | Ncut | Segmentation Benchmark |
| **8** | 10 | yes | 0.5 | Ncut | Segmentation Benchmark |

Table 2: Tests Results

| Test Number | Radius | sampling | sampling rate | Ncut Avg Quality | Ncut' Avg Quality |
|---|---|---|---|---|---|
| **1** | 8N | no | | 0.3064 | 0.3143 |
| **2** | 8N | yes | 0.9 | 0.2956 | 0.3285 |
| **3** | 2 | no | | 0.3216 | 0.3002 |
| **4** | 2 | yes | 0.5 | 0.3041 | 0.3288 |
| **5** | 3 | no | | 0.3298 | 0.3197 |
| **6** | 3 | yes | 0.5 | 0.3176 | 0.3216 |
| **7** | 5 | yes | 0.5 | 0.3395 | |
| **8** | 10 | yes | 0.5 | 0.3750 | |

Table 3: Normalized Cut Criterion Results for Selected Images.

| Image | Ncut | Ncut' |
|---|---|---|
| **100040.pgm** | 0.0351 | 4.50E-04 |
| **90.pgm** | 0.0224 | 0.0022 |
| **Test_100_100.tif** | 3.96E-06 | 3.96E-06 |
| **2.pgm** | 1.28E-07 | 4.07E-09 |
| **1.pgm** | 0.0268 | 4.17E-04 |
| **4.pgm** | 0.0185 | 9.65E-04 |
| **5.pgm** | 0.0018 | 1.05E-04 |
| **6.pgm** | 0.0157 | 0.0010 |
| **7.pgm** | 0.0204 | 0.0039 |
| **8.pgm** | 0.0074 | 3.17E-05 |
| **9.pgm** | 0.0719 | 0.0011 |
| **10.pgm** | 0.0372 | 3.44E-04 |
| **11.pgm** | 0.0345 | 2.26E-04 |
| **12.pgm** | 0.0080 | 3.63E-04 |
| **13.pgm** | 0.0061 | 0.0028 |
| **14.pgm** | 0.0183 | 5.26E-04 |
| **15.pgm** | 0.0399 | 0.0016 |
| **16.pgm** | 0.0292 | 9.21E-04 |
| **17.pgm** | 0.0258 | 6.42E-04 |

Figure 1: Gadget for Constructing *G'* [7]

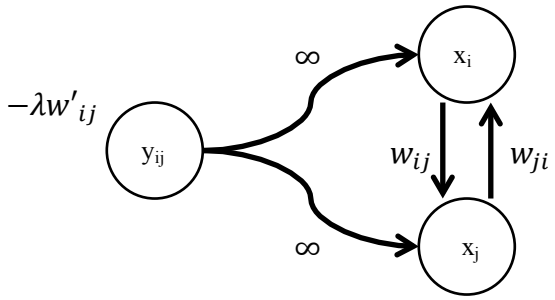



Figure 2: Sample *G'* from sample *G* [7]

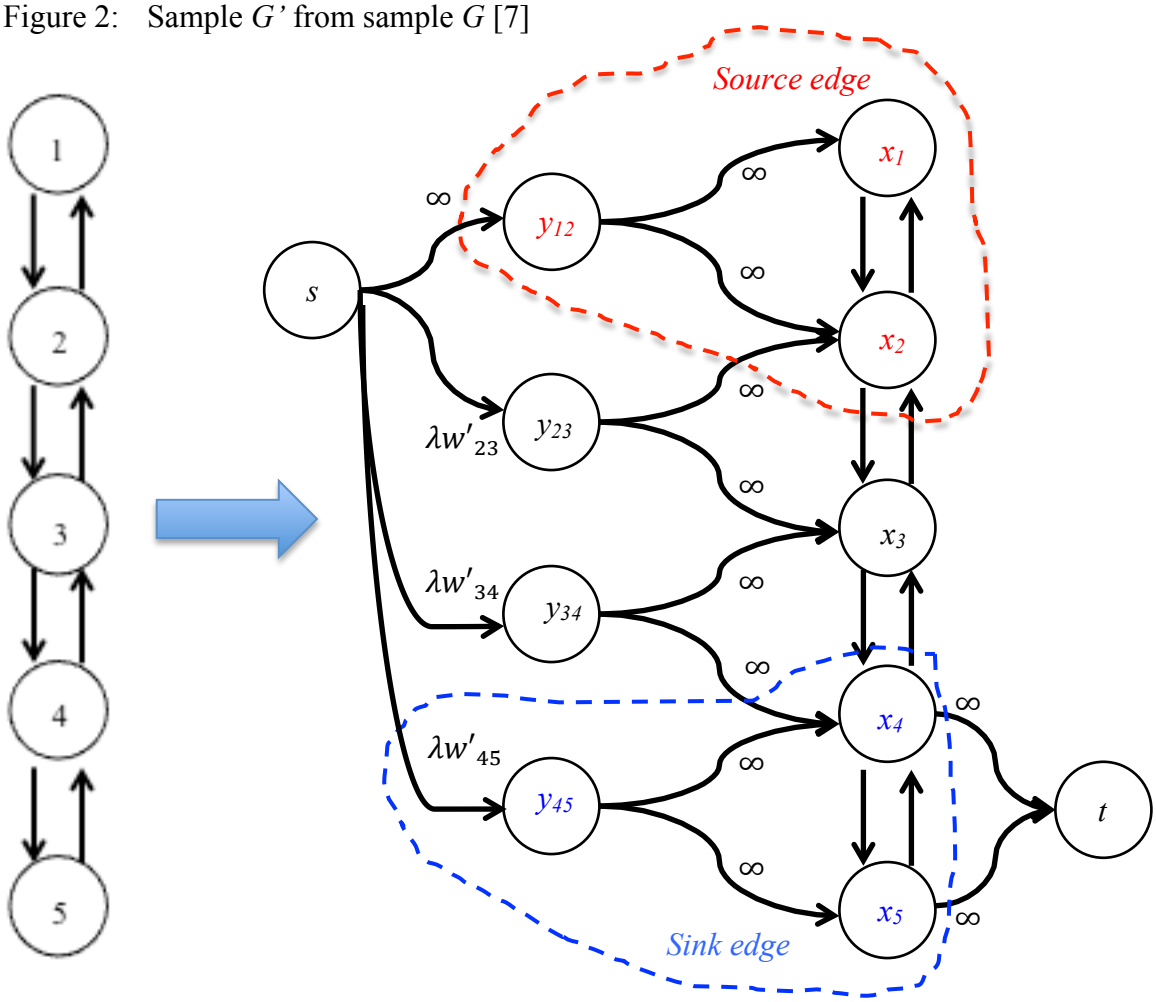


Figure 3: Reduction of Gadget from Figure 1

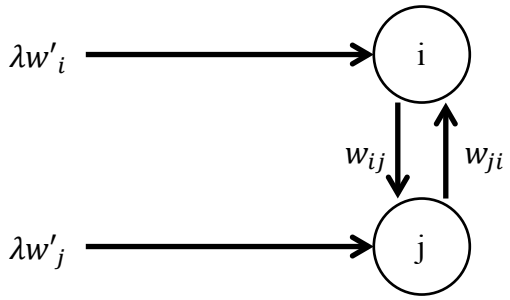



Figure 4: Sample *G''* from *G*

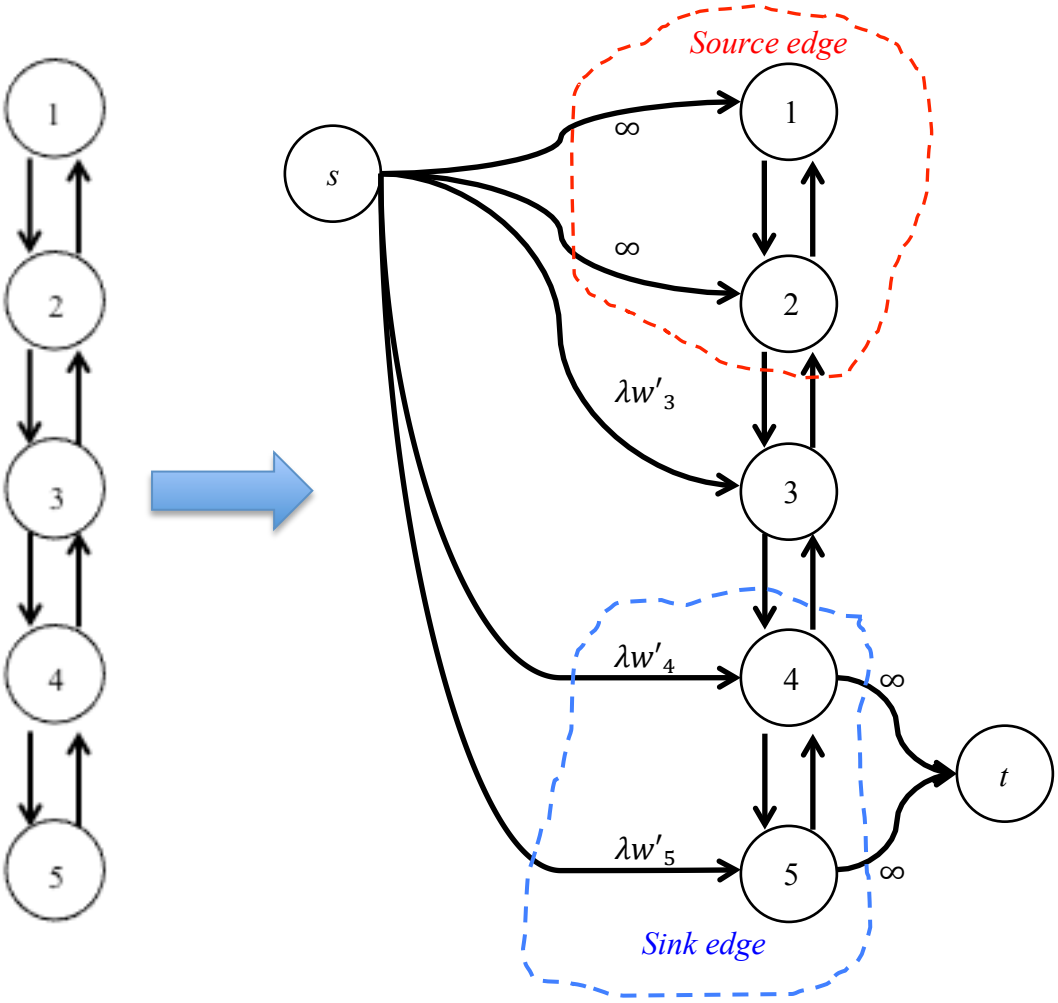


Figure 5: Normalized Cut Result from [6]

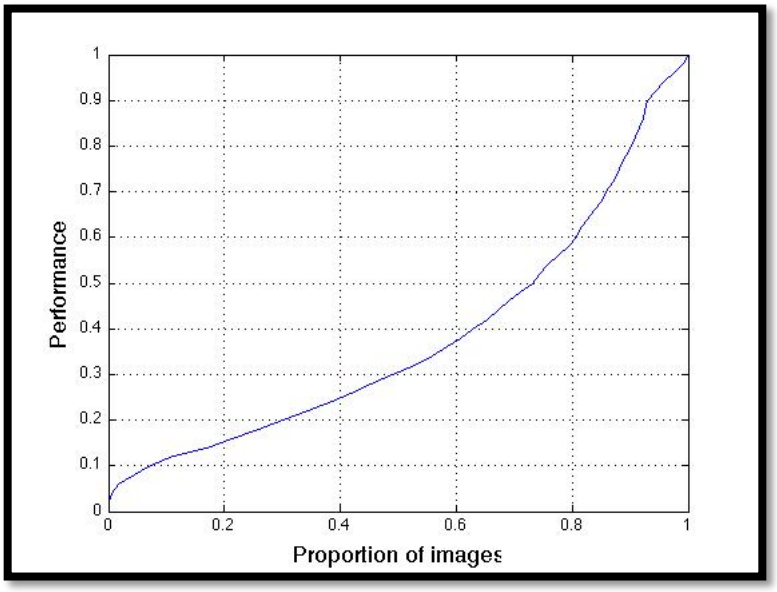



Figure 6: Normalized Cut Result for Default Settings

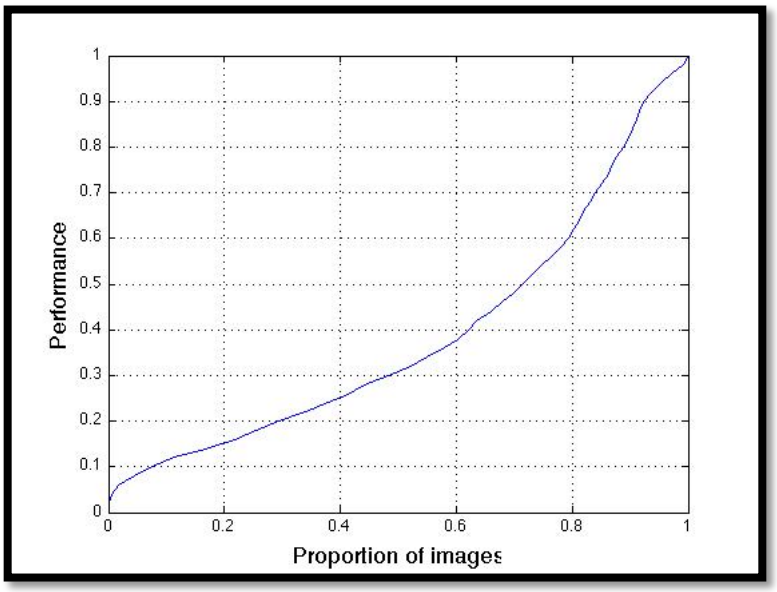


Figure 7: Neighborhood Size vs. Jaccard Value

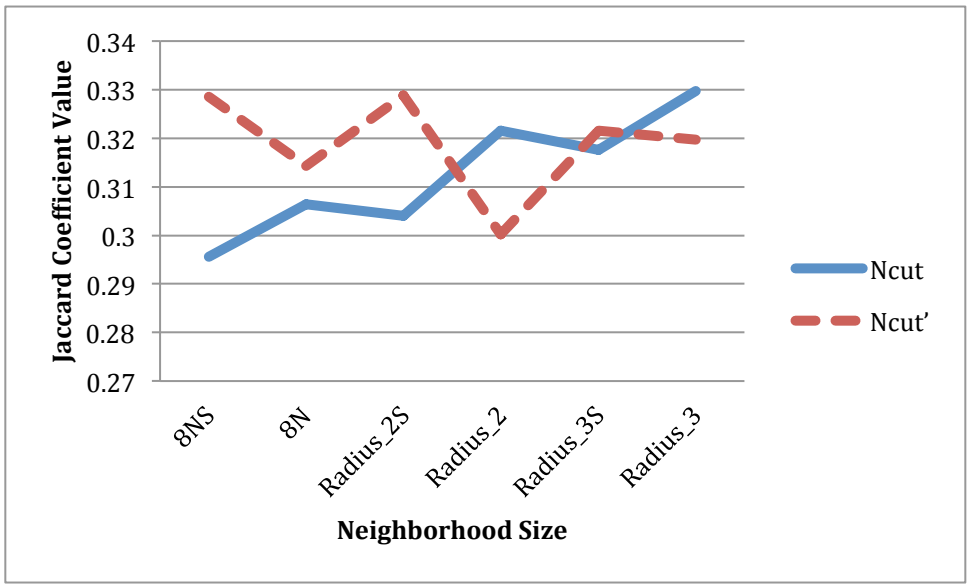



Figure 8: Normalized Cut Result for Radius 3 with Sampling

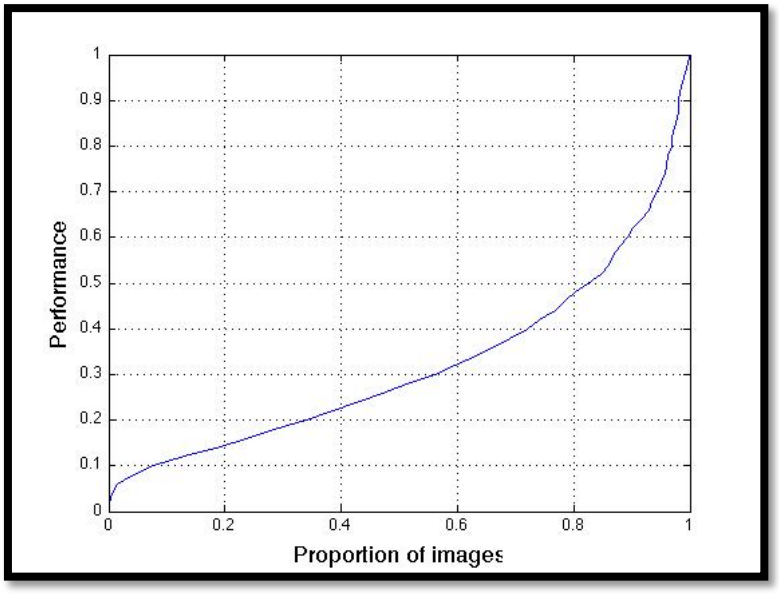


Figure 9: Normalized Cut Result for Radius 3 with Sampling

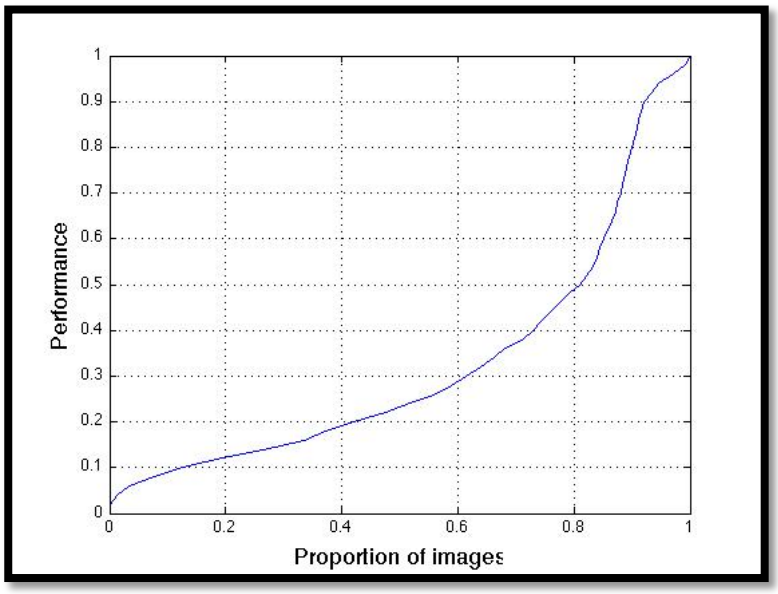



Figure 10: Number of Pixels in an Image vs. Run-Time in seconds

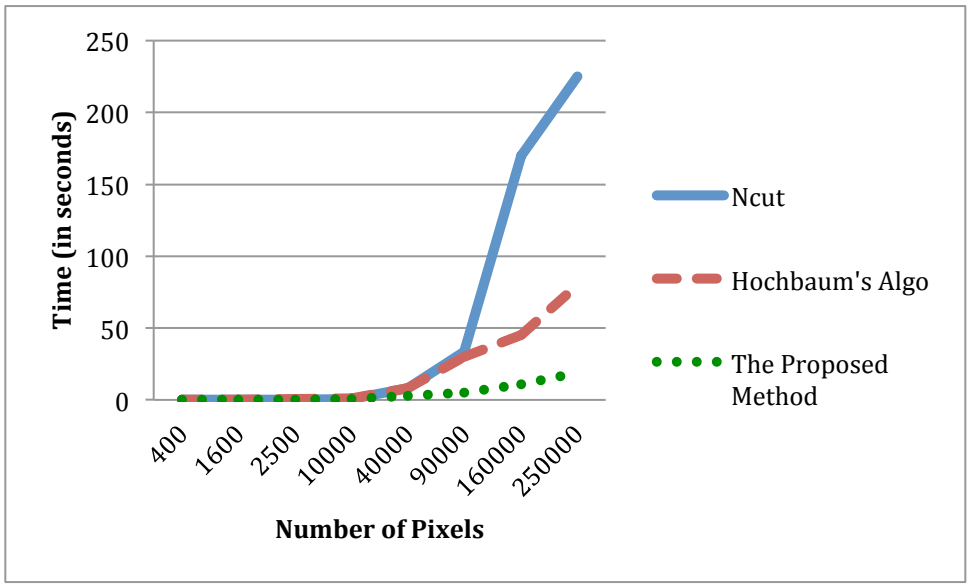


Figure 11: Neighborhood Size vs. Run-Time for Ncut and Ncut'

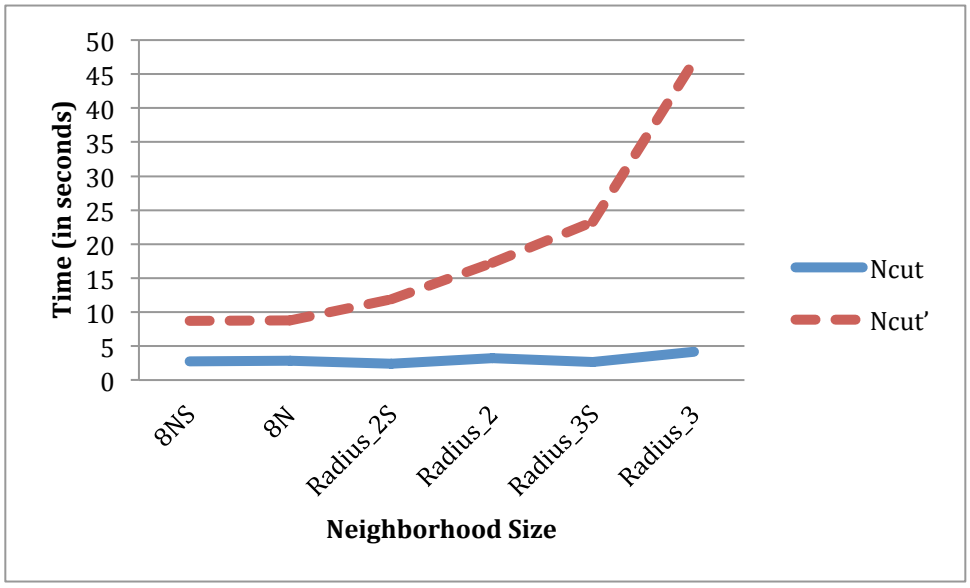



Figure 12: Graph Size vs. Run-Time Contribution due to the Terms in Equation 10 Where *N* is Held Constant.

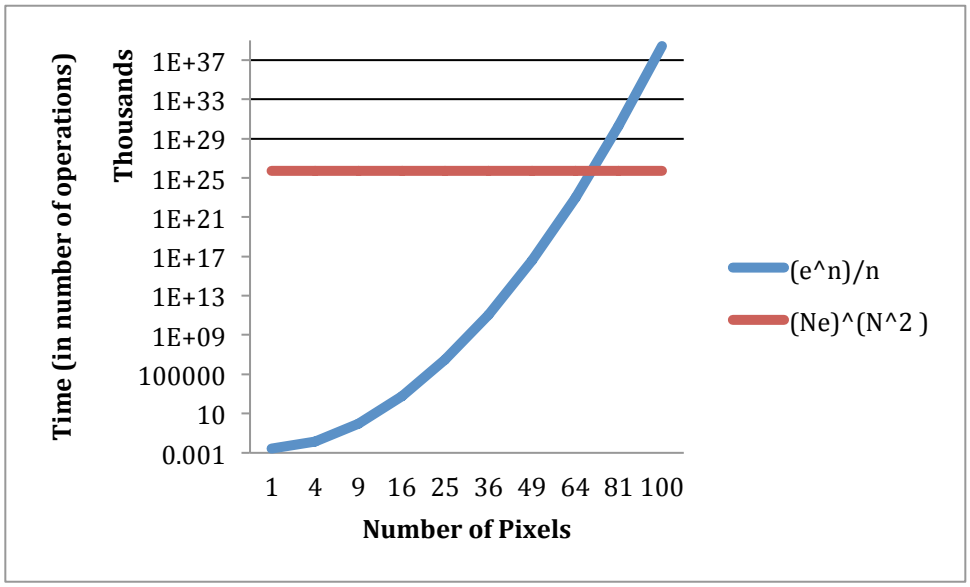


Figure 13: Neighborhood Size vs. Run-Time for Ncut' using both approaches

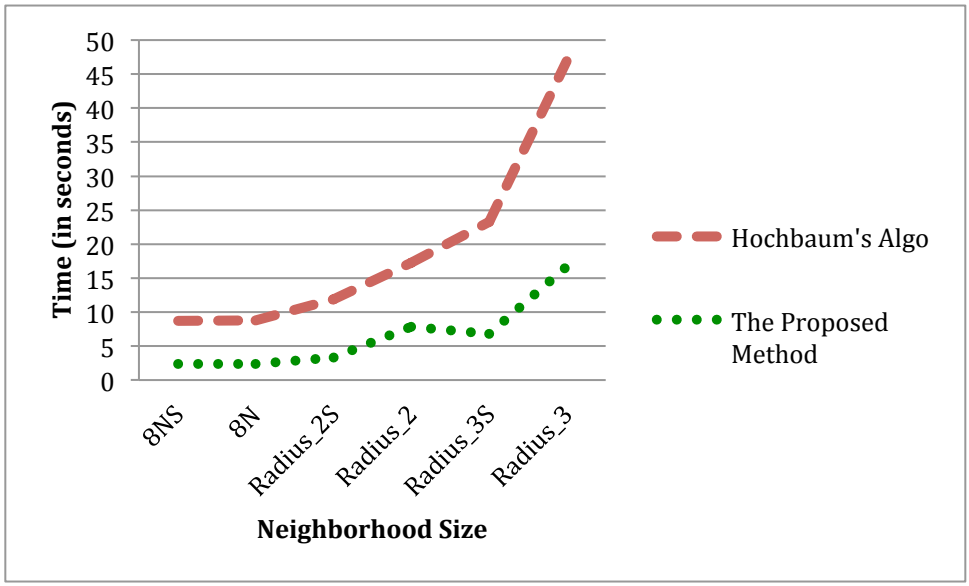



Figure 14: Sample result, a) Original image, b) Normalized Cut c) Normalized Cut'

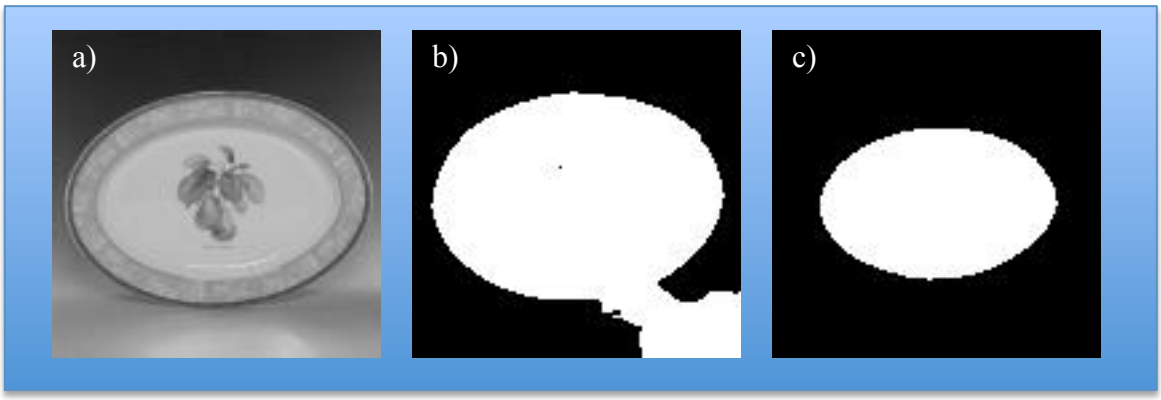


Figure 15: Sample result, a) Original image, b) Normalized Cut c) Normalized Cut'

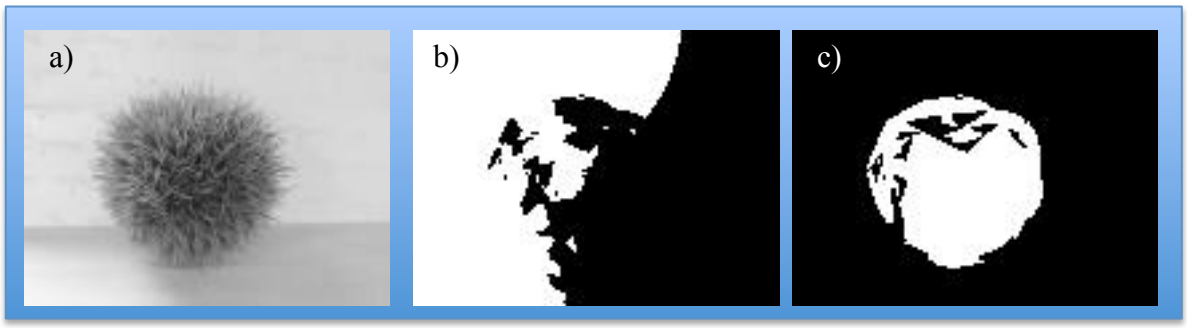



Figure 16: Sample result, a) Original image, b) Normalized Cut c) Normalized Cut'

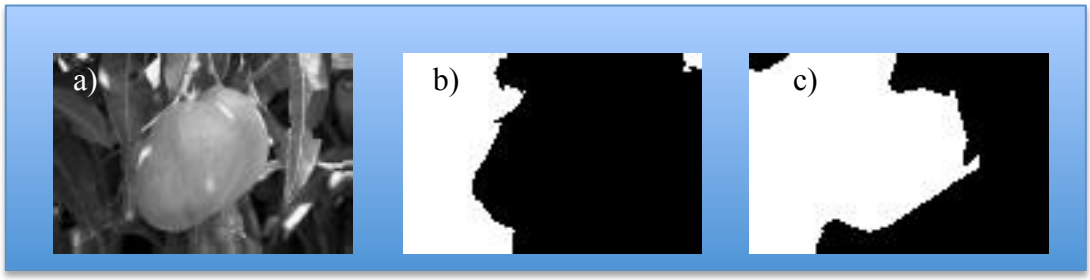
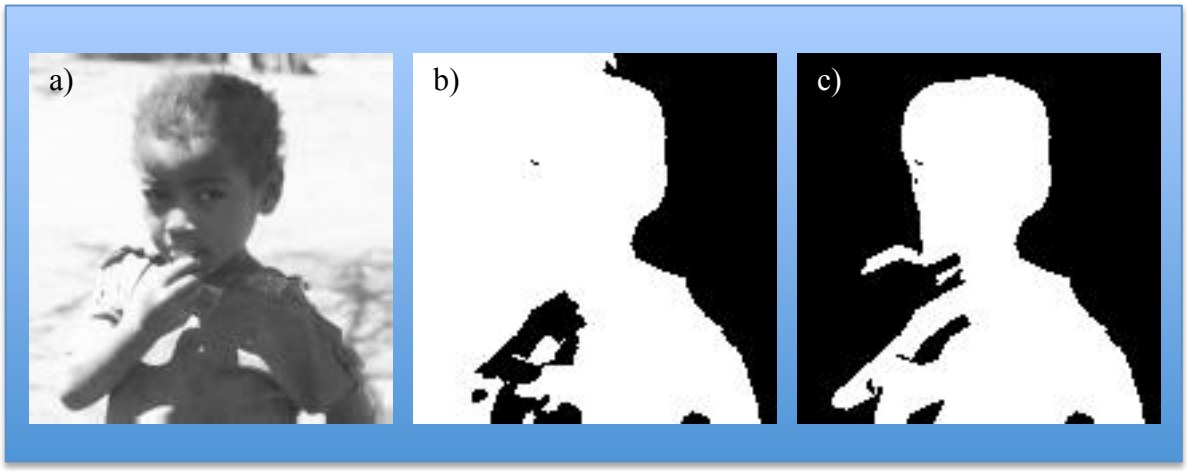



Figure 17: Sample result, a) Original image, b) Normalized Cut c) Normalized Cut'



# REFERENCES


[1] E. Sharon, M. Galun, D. Sharon, R. Basri, and A. Brandt, "Hierarchy and Adaptivity in Segmenting Visual Scenes" *Nature*, volume 442, 2006

[2] C. Florin, N. Paragros, G. Funka-Lea, and J. Williams, "Liver Segmentation Using Sparse 3D Prior Models with Optimal Data Support" *Information Processing in Medical Imaging*, volume 4584, 2007

[3] Z. Wu and R. Leahy, "An Optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 15(11), 1993

[4] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation" *IEEE Transactions on Pattern Analysis and Machine Intelligence,* volume 22(8), 2000

[5] D. Hochbaum, "Polynomial Time Algorithms for Ratio Regions and a Variant of Normalized Cut" *IEEE Transactions on Pattern Analysis and Machine Intelligence,* volume 32, 2010

[6] F. Ge, S. Wang, and T. Liu, "A New Benchmark for Image-Segmentation Evaluation" *Journal of Electronic Imaging,* volume 16(3), 2007

[7] D. Hochbaum, "Polynomial Time Algorithms for Bi-criteria, Multi-objective and Ratio Problems in Clustering and Imaging Part I: Normalized Cut and Ratio Regions" *IEEE Transactions on Pattern Analysis and Machine Intelligence,* volume 32, 2010

[8] M. Babenko and A. Goldberg, "Experimental Evaluation of a Parametric Flow Algorithm" *Microsoft,* Technical Report: MSR-TR-2006-77, 2006

[9] M. Garey, D. Johnson, and L. Stockmeyer, "Some Simplified NP-Complete Problems" *STOC Proceedings of the sixth annual ACM symposium on Theory of computing*, 1974

[10] D. Felleman, D. Van Essen, "Distributed Hierarchical Processing in the Primate Cerebral Cortex" *Cerebral Cortex,* volume 1(1), 1991

[11] G. Miller and D. Tolliver, "Graph Partitioning by Spectral Rounding Applications in Image Segmentation and Clustering" *Computer Vision and Pattern Recognition*, volume 1, 2006

[12] L. Dice, "Measures of the Amount of Ecologic Association Between Species" *Ecology*, volume 26(3), 1945

[13] G. Gallo, M. Grigoriadis, and R. Tarjan, "A New Approach to the Maximum Flow Problem" *SIAM Journal of Computing,* volume 18(1), 1989

[14] D. Hochbaum, "The Pseudoflow Algorithm: A New Algorithm for the Maximum Flow Problem," *Operations Research*, volume 56(4), 2008

[15] P. de Miranda, A. Falcao, J. Udupa, “Synergistic Arc-weight Estimation for Interactive Image Segmentation using Graphs,” *Computer Vision and Image Understanding*, volume 114, 2010