Algorithms for Image Segmentation

THESIS

submitted in partial fulfillment of the requirements of
BITS C421T/422T Thesis

by

Yatharth Saraf

ID No. 2001A2A7774

under the supervision of:
Dr. R. R. Mishra
Group Leader, Physics Group
BITS, Pilani

Birla Institute of Technology and Science, Pilani
Rajasthan – 333031

4th May, 2006

ACKNOWLEDGEMENTS

I am grateful to Dr. R. R. Mishra for agreeing to be my thesis supervisor and guiding me during the course of my work. I am thankful to the Research and Consultancy Division for permitting me to do a thesis. Thanks are also due to Mr. V. Ramakrishna and the Practice School Division for enabling my transfer from PS to TS. I am also indebted to Prof. P. K. Thiruvikraman for kindly consenting to attend the seminars and presentations for this thesis.

BITSConnect has been a huge help in every way. Many thanks to the institute and the alumni who made it possible. IPC facilities have been a tremendous help as well. Sincere thanks to the IPC Personnel.

CERTIFICATE

This is to certify that the Thesis entitled, Algorithms for Image Segmentation and submitted by Yatharth Saraf ID No. 2001A2A7774 in partial fulfillment of the requirements of BITS C421T/422T Thesis embodies the work done by him under my supervision.

Dr. R. R. Mishra
Group Leader
Physics Group
BITS, Pilani

Date:

LIST OF SYMBOLS AND ABBREVIATIONS USED

cut	Summation of weights of edges going across a graph partition
$Ncut$	Normalized Cut
$Nassoc$	Normalized Association
V	Set of vertices in a graph
E	Set of edges of a graph
$\cup$	Set Union
$\cap$	Set Intersection
$\oslash$	Null Set
n_r	Number of rows in an image
n_c	Number of columns in an image
w_{ij} or $w(i,j)$	Weight of edge connecting node i to node j
N	Total number of pixels in the image $(n_r \times n_c)$
W	Adjacency matrix storing graph edge weights. $N \times N$ matrix such that $W(i,j) = w_{ij}$.

Abstract

In image analysis, segmentation is the partitioning of a digital image into multiple regions (sets of pixels), according to some homogeneity criterion. The problem of segmentation is a well-studied one in literature and there are a wide variety of approaches that are used. Different approaches are suited to different types of images and the quality of output of a particular algorithm is difficult to measure quantitatively due to the fact that there may be many "correct" segmentations for a single image. Here, a graph-theoretic framework is considered by modeling image segmentation as a graph partitioning and optimization problem using the *normalized cut* criterion. Comparisons with other criteria shows that the results for *normalized cut* are quite good although high computational complexity is a drawback.

Contents

1 Introduction

Depending on the image acquisition model, images can be classified into various types; namely light intensity (visual) images, range or depth images, magnetic resonance images, thermal images and so on. Light intensity images represent the variation of light intensity on the scene and are the most common types of images we encounter in our daily experience. A Range image is a map of depth information at different points on the scene.

Segmentation is the process of partitioning an image into non-intersecting regions such that each region is homogeneous and the union of no two adjacent regions is homogeneous. Formally, it can be defined as follows.

Definition 1 *Let F be the set of all pixels and $P()$ be a uniformity (homogeneity) predicate defined on groups of connected pixels, then segmentation is a partitioning of the set F into a set of connected subsets or regions $(S_1, S_2, \cdots, S_n)$ such that $\cup_{i=1}^{n} S_i = F$ with $S_i \cap S_j = \oslash$ when $i \neq j$. The uniformity predicate $P(S_i)$ is true for all regions S_i and $P(S_i \cup S_j)$ is false when S_i is adjacent to S_j.*

This definition can be applied to all types of images.

The goal of segmentation is typically to locate certain objects of interest which may be depicted in the image. Segmentation could therefore be seen as a computer vision problem. A simple example of segmentation is thresholding a grayscale image with a fixed threshold t: each pixel p is assigned to one of two classes, P_0 or P_1, depending on whether $I(p) < t$ or $I(p) >= t$.

For intensity images (i.e., those represented by point-wise intensity levels), four popular segmentation approaches are: threshold techniques, edge-based methods, region-based techniques, and connectivity-preserving relaxation methods.

Threshold techniques make decisions based on local pixel information and are effective when the intensity levels of the objects fall squarely outside the range of levels in the background. Because spatial information is ignored, however, blurred region boundaries can create havoc.

Edge-based methods center around contour detection: their weakness in connecting together broken contour lines make them, too, prone to failure in the presence of blurring.

A region-based method usually proceeds as follows: the image is partitioned into connected regions by grouping neighboring pixels of similar intensity levels. Adjacent regions are then merged under some criterion involving perhaps homogeneity or sharpness of region boundaries. Overstringent criteria create fragmentation; lenient ones overlook blurred boundaries and overmerge.

A connectivity-preserving relaxation-based segmentation method, usually referred to as the *active contour model*, starts with some initial boundary shape represented in the form of spline curves, and iteratively modifies it by applying various shrink/expansion operations according to some energy function. Although the energy-minimizing model is not new, coupling it with the maintenance of an "elastic" contour model gives it an interesting new twist. As usual with such methods, getting trapped into a local minimum is a risk against which one must guard; this is no easy task.

1.1 Segmentation by Edge Detection

The edge-based methods make use of various edge operators to produce an "edginess" value at each pixel. The values are then thresholded to obtain the edges. The regions within connected edges can be considered as different segments because they lack continuity with adjacent regions. The Sobel operator was studied and implemented to find edges in images. The edges thus found could also be used as aids by other image segmentation algorithms for refinement of segmentation results.

In simple terms, the operator calculates the gradient of the image intensity at each point, giving the direction of the largest possible increase from light to dark and the rate of change in that direction. The result therefore shows how "abruptly" or "smoothly" the image changes at that point, and therefore how likely it is that that part of the image represents an edge, as well as how that edge is likely to be oriented. In practice, the magnitude (likelihood of an edge) calculation is more reliable and easier to interpret than the direction calculation.

In theory at least, the operator consists of a pair of 3×3 convolution masks as shown in Figure 1. One mask is simply the other rotated by 90 degrees. This is very similar to the Roberts Cross operator.

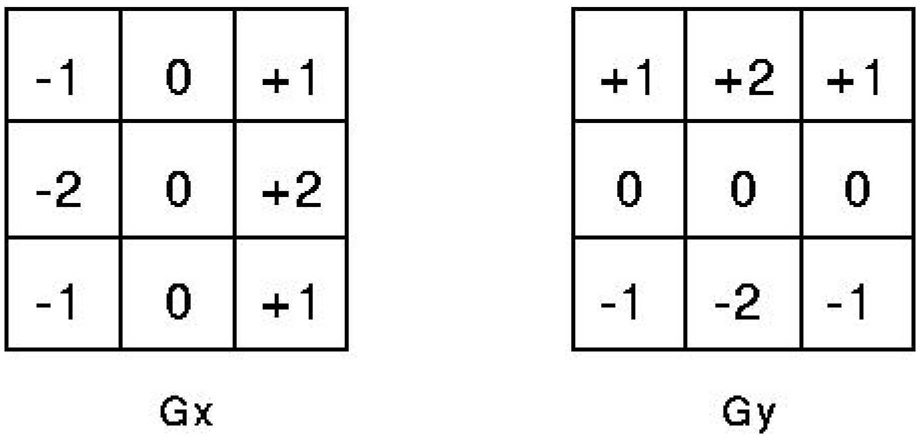
These masks are designed to respond maximally to edges running vertically and horizontally relative to the pixel grid, one mask for each of the two perpendicular orientations. The masks can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (call these G_x and G_y). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The gradient magnitude is given by:

$$|G| = \sqrt{{G_x}^2 + {G_y}^2}$$

Output values from the operator can easily overflow the maximum allowed pixel value for image types that only support smallish integer pixel values (e.g. 8-bit integer images). When this happens the standard practice is to simply set overflowing output pixels to the maximum allowed value. The problem can be avoided by using an image type that supports pixel values with a larger range. Natural edges in images often lead to lines in the output image that are several pixels wide due to the smoothing effect of the Sobel operator. Some thinning may be desirable to counter this. Some results of edges detected by the Sobel operator are shown in Figures 2, 3 and 4.

1.2 Segmentation by Grouping

Image segmentation can be related to perceptual grouping and organization in vision and several key factors, such as similarity, proximity, and good continuation, lead to visual grouping [1]. However, many of the computational issues of perceptual grouping have remained unresolved. In this report, a



-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figure 1: *Sobel convolution masks*

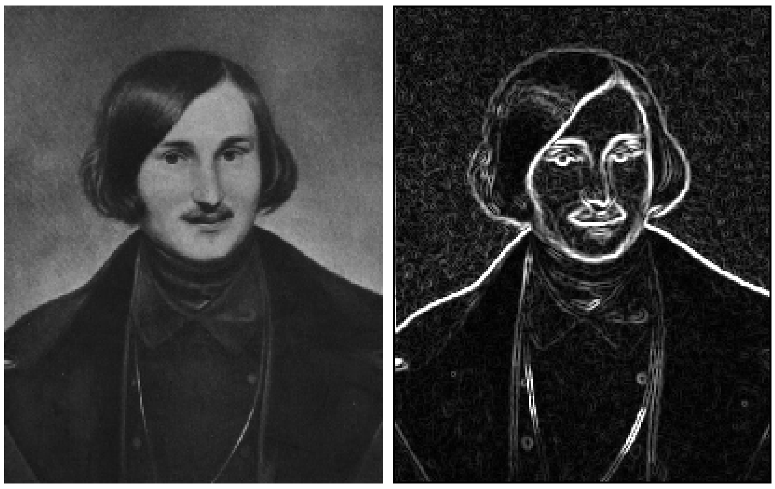


Figure 2: *Edge detection of a man's image with the Sobel operator*

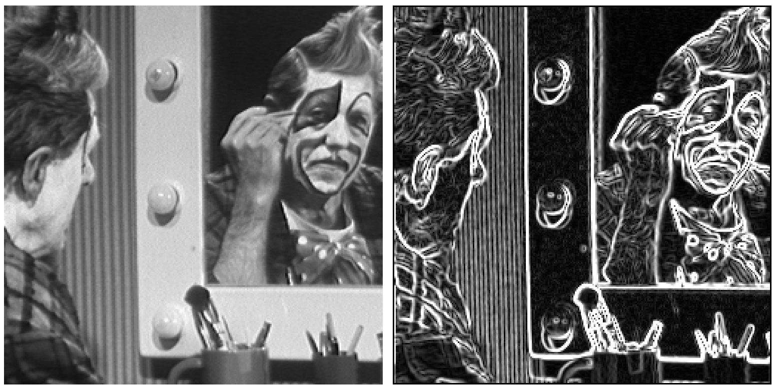


Figure 3: *Edge detection of a clown image with the Sobel operator*

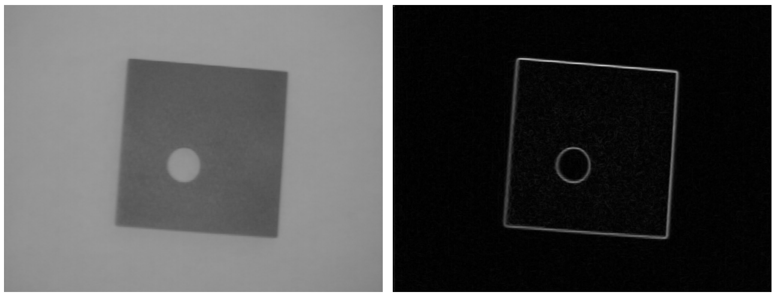


Figure 4: *Edge detection of a wedge image with the Sobel operator*

graph theoretic approach to this problem is adopted, focusing specifically on the case of image segmentation.

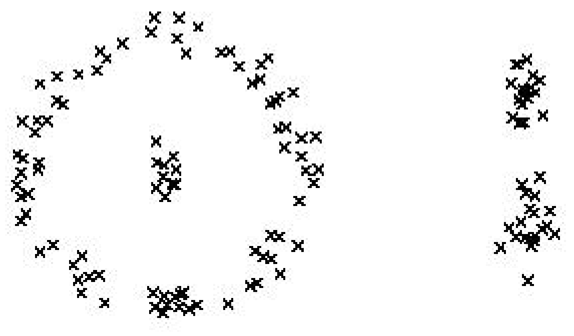


Figure 5: *Points in a plane - what is the "correct" grouping?*

Since there are many possible partitions of an image into subsets, how do we pick the "right" one? This is illustrated in Figure 5 where there are multiple groupings possible. There are two aspects to be considered here. The first is that there may not be a single correct answer. A Bayesian view is appropriate - there are several possible interpretations in the context of prior world knowledge. The difficulty, of course, is in specifying the prior world knowledge. Some of it is in the form of local properties, such as coherence of brightness, color, texture, or motion, but equally important are global properties about symmetries of objects or object models. The second aspect is that the partitioning is inherently hierarchical. Therefore, it is more appropriate to think of returning a tree structure corresponding to a hierarchical partition instead of a single "flat" partition.

2 Graph Theoretic Formulation

Grouping can be formulated as a graph partitioning and optimization problem. The graph theoretic formulation of image segmentation is as follows:

1. The set of points in an arbitrary feature space are represented as a weighted undirected graph $G = (V, E)$, where the nodes of the graph are the points in the feature space

2. An edge is formed between every pair of nodes yielding a dense or complete graph.

3. The weight on each edge, $w(i,j)$ is a function of the similarity between nodes i and j.
4. Partition the set of vertices into disjoint sets $V_1, V_2, \cdots, V_k$ where by some measure the similarity among the vertices in a set V_i is high and, across different sets V_i, V_j is low.

To partition the graph in a meaningful manner, we also need to:

- Pick an appropriate criterion (which can be computed from the graph) to optimize which would result in a good segmentation.
- Finding an efficient way to achieve the optimization.

In the image segmentation and data clustering community, there has been much previous work using variations of the minimal spanning tree or limited neighborhood set approaches. Although those use efficient computational methods, the segmentation criteria used in most of them are based on local properties of the graph. Because perceptual grouping is about extracting the global impressions of a scene, as we saw earlier, this partitioning criterion often falls short of this main goal.

2.1 Weight Function

The weight function that we use to represent similarity between nodes must have generality, must be easy to compute, should be tweakable with parameters, and should take into account pixel gray level difference as well as radial distance between two pixels. One such weight function is:

$$w_{ij} = \exp - \frac{(I(i) - I(j))^2}{\sigma_I^2} * \begin{cases} \exp - \frac{\|X(i)-X(j)\|_2^2}{\sigma_X^2} & \text{if } \|X(i) - X(j)\|_2 < R \\ 0 & \text{otherwise} \end{cases}$$

For brightness images, $I(i)$ represents normalized intensity level of node i and $X(i)$ represents spatial location of node i. σ_I and σ_X are parameters set to 10-20 percent of the range of their related values. R is a parameter that controls the sparsity of the resulting graph by setting edge weights between distant pixels to 0.

This weight measure reflects likelihood of two pixels belonging to the same object. As gray level difference and Euclidean distance decrease (in other words, as pixels become more similar), $w(i,j)$ increases. Figure 6 illustrates this schematically with a graph superimposed on the corresponding image. Similar pixels have thicker edges between them.

Figure 7 shows the weight matrix, W, which represents the graph of the brightness image. W is an adjacency matrix for the graph and is an $N \times N$ matrix such that $W(i, j) = w_{ij}$. Here N is the total number of pixels in the image (number of rows $\times$ number of columns). Figures 7 (c) and (d) show the weights to all the other nodes from two particular pixels, i_1 and i_2 in the form of brightness images. The weights are shown in the form of an image with higher weights being denoted by brighter points in the corresponding images. Thus, pixels similar to the selected pixel are brighter and the ones which are dissimilar are darker in the two figures (c) and (d).

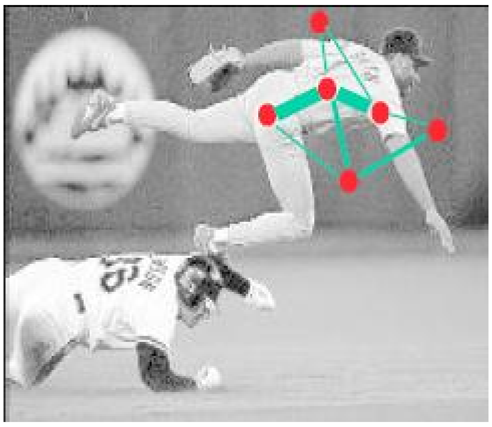


Figure 6: *Schematic diagram showing graph weight edges for a gray scale image. Higher weights are shown as thicker edges.*

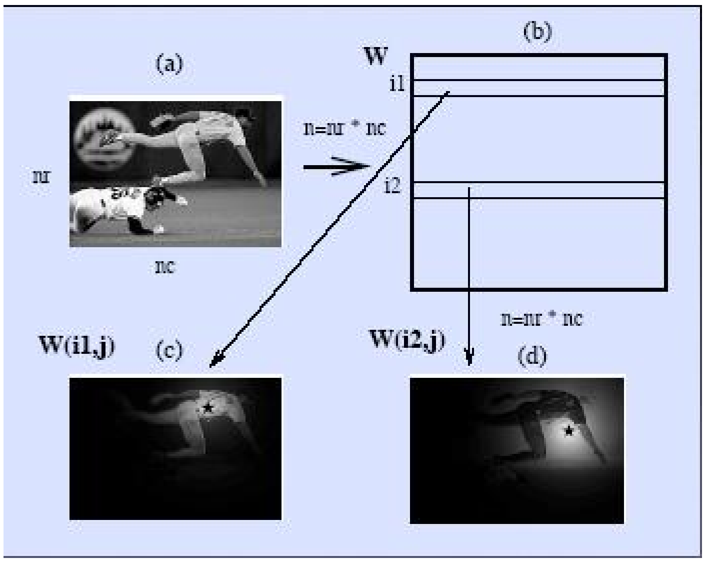


Figure 7: *Edge weights shown in terms of brightness values for two pixels*

2.2 Minimum Cut

A graph can be partitioned into two disjoint sets by simply removing the edges connecting the two parts. The degree of dissimilarity between these two pieces can be computed as total weight of the edges that have been removed. More formally, it is called the *cut*.

$$cut(A, V - A) = \sum_{u \in A, v \in V-A} w(u, v) \tag{1}$$

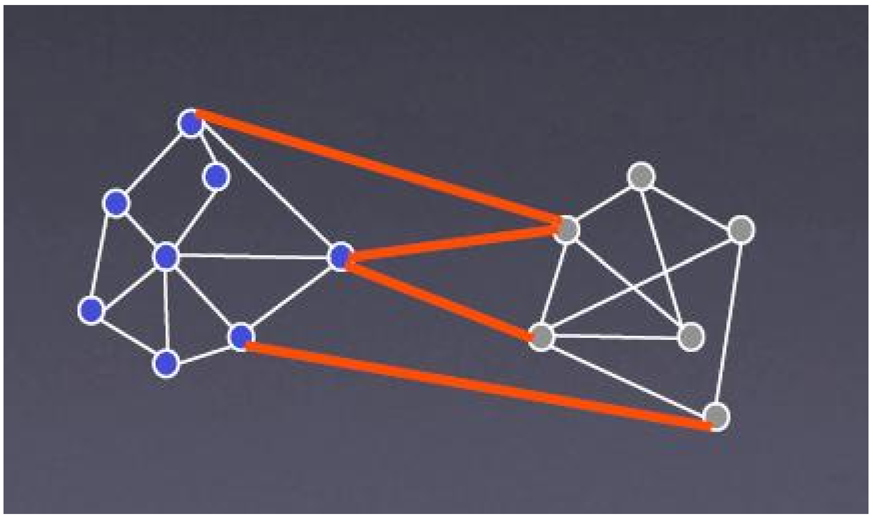


Figure 8: *The sum of the weights of the red edges represents the cut value between the set of nodes on the left and on the right.*

Since the *cut* is a summation of edge weights across the two sets, it represents the amount of similarity between them. In segmentation, the goal is to identify sets which have low similarity and thus we could try to find a partition that minimizes this cut value. Therefore, we could consider the *cut* as our criterion for partitioning and try to solve the optimization problem:

Minimize:

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v)$$

Subject to the constraints: $A \cup B = V$, $A \cap B = \oslash$.

Although there are an exponential number of such partitions, the minimum cut of a graph can be found efficiently [3]. For getting k segments, we can partition the graph into k subgraphs such that the maximum cut across the subgroups is minimized. This problem can be efficiently solved by recursively finding the minimum cuts that bisect the existing segments.

However, the minimum cut criterion favors cutting small sets of isolated nodes in the graph. This is not surprising since the cut defined in Equation 1 increases with the number of edges going across the two partitioned parts. Figure 9 illustrates one such case.

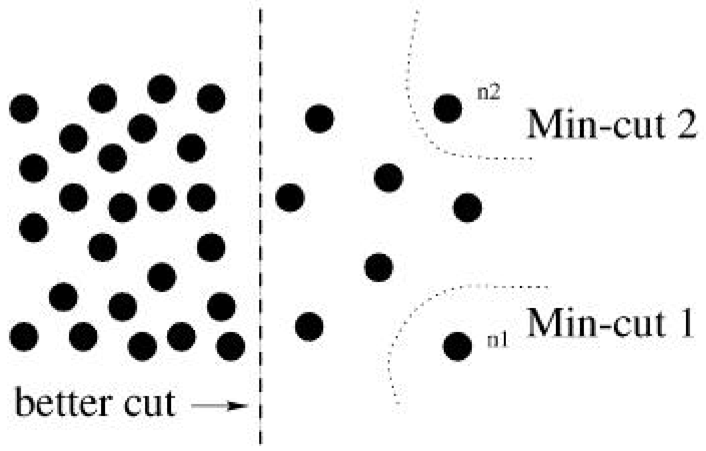


Figure 9: *Minimum Cut favors cutting off singular, isolated nodes in order to reduce the cut value*

Assuming the edge weights are inversely proportional to the distance between the two nodes, we see the cut that partitions out node n_1 or n_2 will have a very small value. In fact, any cut that partitions out individual nodes on the right half will have smaller cut value than the cut that partitions the nodes into the left and right halves because the number of edges going across the cut reduces as the number of nodes in a set is reduced.

2.3 Normalized Cut

To avoid this unnatural bias for partitioning out small sets of points, we must look at a different measure of disassociation. The problem with the *cut* criterion is that it does not consider association within a cluster. In order to circumvent this problem, we can look at the cut cost as a fraction of the total edge connections to all the nodes in the graph. Thus, we can define the *normalized cut* as:

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \tag{2}$$

where $assoc(A,V) = \sum_{u \in A, v \in V} w(u,v)$ is the total connection from nodes in A to all nodes in the graph. With this definition of the disassociation between the groups, the cut that partitions out small isolated points will no longer have small $Ncut$ value, since the cut value will almost certainly be a large percentage of the total connection from that small set to all other nodes. In the case illustrated in Fig. 9, we see that the cut_1 value across node n_1 will be 100 percent of the total connection from that node. To minimize dissimilarity across groups, we can try to minimize $Ncut$ subject to the same constraints as those for minimum cut.

Alternatively, we can try to look for clusters which have high within-group similarity. Thus, we can define the *normalized association* as:

$$Nassoc(A,B) = \frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)} \tag{3}$$

where $assoc(A,A)$ and $assoc(B,B)$ are total weights of edges connecting nodes within A and B, respectively. This measure reflects how tightly nodes within the group are connected to each other as a fraction of the total connection from the group to all other nodes.

A very important property that can be derived here is that $Ncut$ and $Nassoc$ are naturally related.

$$\begin{aligned} Ncut(A,B) &= \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)} \\ &= \frac{assoc(A,V) - assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,V) - assoc(B,B)}{assoc(B,V)} \\ &= 1 - \frac{assoc(A,A)}{assoc(A,V)} + 1 - \frac{assoc(B,B)}{assoc(B,V)} \\ &= 2 - \left(\frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)} \right) \\ &= 2 - Nassoc(A,B) \end{aligned}$$

Hence, the two partition criteria that we seek in our grouping algorithm, minimizing the disassociation between the groups and maximizing the association within the groups are in fact, identical and can be satisfied simultaneously. Thus, the $Ncut$ criterion takes care of both inter-set dissimilarity (by the minimization of $Ncut$) as well as intra-set similarity (by the simultaneous maximization of $Nassoc$).

But there is a major stumbling block. An exact solution to minimizing normalized cut is an NP-complete problem [4]. However, approximate discrete solutions can be found efficiently because the normalized cut criterion can be computed by solving a generalized eigenvalue problem.

3 Matrix Formulation

The criteria given in the preceding sections can be cast into matrix form which in turn can be used for converting the optimization problem into a generalized eigenvalue problem.

Let $\mathbf{x}$ be an $N = |V|$ dimensional indicator vector such that

$$\begin{aligned} x_i &= 1 \quad \text{if } i \text{ belongs to set } A \\ &= 0 \quad \text{otherwise} \end{aligned}$$

In Figure 10, the entries in $\mathbf{x}$ shaded yellow signify nodes that are in set A. Multiplying $\mathbf{x}^T W \mathbf{x}$ will give the sum of the weights within the red rectangle which is nothing but twice the sum of the weights connecting nodes in A or $2 * assoc(A, A)$. Similarly, in Figure 11, the sum of the weights within the blue rectangle represents $2 * assoc(A, V)$. This sum cannot be found by multiplying the indicator vector $\mathbf{x}$ with the weight matrix. Instead, we use the diagonal matrix D such that the diagonal entries of D contain the sums of the corresponding row in W. So that $D(i,i) = \sum_j W(i,j)$. Now, the sum of the weights within the blue rectangle can be written $\mathbf{x}^T D \mathbf{x}$.

We can see that

$$\begin{aligned} 2 * assoc(A, A) &= \mathbf{x}^T W \mathbf{x} \\ 2 * assoc(A, V) &= \mathbf{x}^T D \mathbf{x} \\ 2 * cut(A, V - A) &= 2 * assoc(A, V) - 2 * assoc(A, A) \\ &= \mathbf{x}^T (D - W) \mathbf{x} \end{aligned}$$

Thus, the sum of the weights within the red rectangle in Figure 11 represents $2 * cut(A, V - A)$.

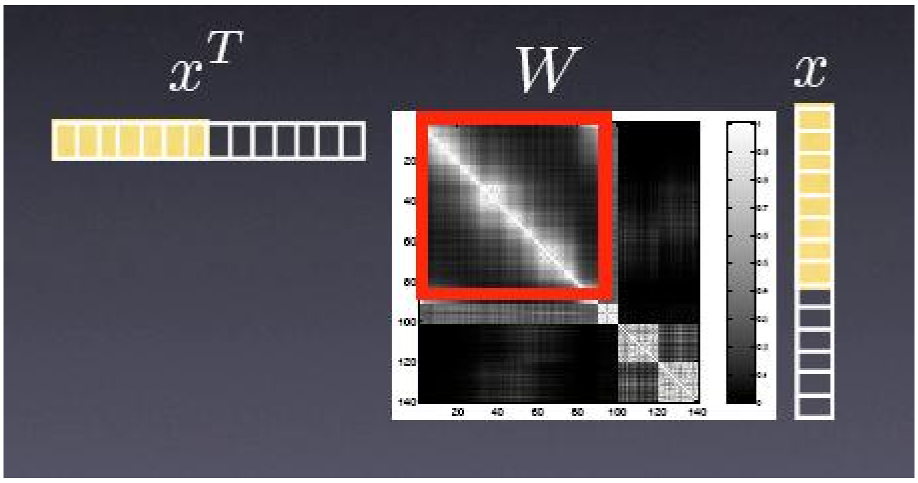


Figure 10: *The points shaded yellow within the indicator vector* $\mathbf{x}$ *denote nodes belonging to set A (they all correspond to 1's). Sum of weights within the red rectangle represents 2*assoc(A,A).*

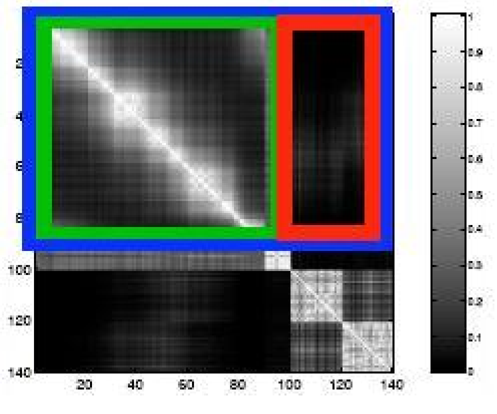


Figure 11: *Sum of weights within (i) blue rectangle represents 2*assoc(A,V) (ii) red rectangle represents 2*cut(A,V-A).*

4 Algorithm for Brightness Images

Segmentation of brightness images based on minimizing normalized cut can be achieved as follows:

1. Construct the weighted graph representing the image. Summarize the information into matrices, W and D. Edge weight is an exponential function of feature similarity as well as distance measure as given in Section 2.1. The matrices must have appropriate representations so that their sparse structures are leveraged to reduce storage space.

2. Solve $(D - W)\mathbf{x} = \lambda D\mathbf{x}$ for eigenvectors with the smallest eigenvalues. This can be transformed to a standard eigenvalue problem [4]:

$$D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}\mathbf{x} = \lambda\mathbf{x} \tag{4}$$

3. Partition the graph into two pieces using the second smallest eigenvector. Signs of the eigenvector tell us exactly how to partition the graph.

4. Recursively run the algorithm on the two partitioned parts. Recursion stops once the $Ncut$ value exceeds a certain limit. This maximum allowed $Ncut$ value controls the number of groups segmented.

Solving Equation 4 for all eigenvectors takes $O(n^3)$ operations, where n is the number of nodes in the graph. This becomes impractical for image segmentation applications where n is the number of pixels in an image. The eigenvalue solution was implemented in C using the QR-reduction method. QR-reduction attempts to find all the eigenvectors and ran too slowly to be of any use even for very small images. For the problem at hand, the graphs are often only very locally connected, only the top few eigenvectors are needed for graph partitioning, and the precision requirement for the eigenvectors is low, only the right sign bit is required. These features might be better exploited by an eigensolver based on the Lanczos' method.

5 A Physical Interpretation

We can draw an analogy between a spring-mass system and the weighted graph by taking graph nodes as physical masses and graph edges as springs connecting each pair of nodes. Furthermore, we will define the graph edge weight as the spring stiffness and the total edge weights connecting to a node as its mass.

Imagine what would happen if we were to give a hard shake to this spring-mass system, forcing the nodes to oscillate in the direction perpendicular to the image plane. Nodes that have stronger spring connections among them will likely oscillate together. As the shaking becomes more violent, weaker springs connecting to this group of node will be overstretched. Eventually, the group will "pop" off from the image plane. The overall steady state behavior of the nodes can be described by its fundamental mode of oscillation.

Let k_{ij} be the spring stiffness connecting nodes i and j. Define K to be the $N \times N$ stiffness matrix. Define the diagonal $N \times N$ mass matrix M as $M(i,i) = \sum_j k_{ij}$. Let $x(t)$ be the $N \times 1$ vector describing the motion of each node. This spring-mass dynamic system can be described by:

$$Kx(t) = -M\ddot{x}(t)$$

Assuming the steady state solutions of this spring-mass system take the form $x(t) = v_k \cos(\omega_k t + \theta)$, the eigenvectors v_k give the steady state displacement of the oscillation in each mode and the eigenvalues ω_k^2 give the energy required to sustain each oscillation. Therefore, finding graph partitions that have small normalized cut values is, in effect, the same as finding a way to "pop" off image regions with minimal effort.

6 Comparisons with other criteria

There are also other formulations for image segmentation as a graph partitioning problem using criteria different from Normalized Cut. The *Average Cut* criterion is:

$$min_{A \in V} \frac{cut(A, V-A)}{|A|} + \frac{cut(A, V-A)}{|V-A|}$$

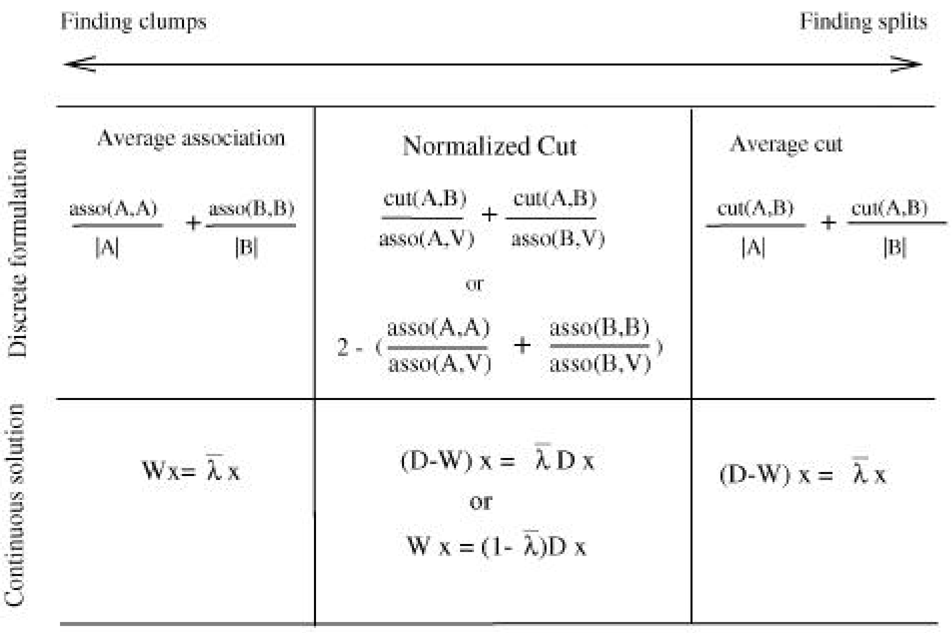
Analogously, we can define *Average Association* as:

$$max_{A \in V} \frac{assoc(A, A)}{|A|} + \frac{assoc(V-A, V-A)}{|V-A|}$$

All three of these criteria can be reduced to solving certain eigenvalue systems. How are they related to each other?

The relationship between these three criteria is summarized in Figure 12. On one hand, both the *normalized cut* and the *average cut* algorithm are trying to find a "balanced partition" of a weighted graph, while, on the other hand, the *normalized association* and the *average association* are trying to find "tight" clusters in the graph. As explained in Section 2.3, *Ncut* and *Nassoc* are related to each other such that minimizing one is equivalent to maximizing the other. Thus, the normalized cut formulation seeks a balance between the goal of clustering and partitioning.

Unlike in the case of *normalized cut* and *normalized association*, *average cut* and *average association* do not have a simple relationship between them. Consequently, one cannot simultaneously minimize the disassociation across the partitions while maximizing the association within the groups.



Finding clumps $\longleftrightarrow$ Finding splits

	Average association	Normalized Cut	Average cut
Discrete formulation	$\frac{asso(A,A)}{\lvert A\rvert} + \frac{asso(B,B)}{\lvert B\rvert}$	$\frac{cut(A,B)}{asso(A,V)} + \frac{cut(A,B)}{asso(B,V)}$ or $2 - (\frac{asso(A,A)}{asso(A,V)} + \frac{asso(B,B)}{asso(B,V)})$	$\frac{cut(A,B)}{\lvert A\rvert} + \frac{cut(A,B)}{\lvert B\rvert}$
Continuous solution	$Wx = \bar{\lambda} x$	$(D-W)x = \bar{\lambda} D x$ or $W x = (1-\bar{\lambda}) D x$	$(D-W)x = \bar{\lambda} x$

Figure 12: *Compared to the average cut and average association formulation, normalized cut seeks a balance between the goal of finding clumps and finding splits*

For a performance comparison among the different formulations, consider random 1-D data points as shown in Figure 13. Each data point is a node in the graph and the weighted graph edge connecting two points is defined to be inversely proportional to the distance between two nodes. We will consider two different monotonically decreasing weight functions, $w(i,j) = f(d(i,j))$, defined on the distance function, $d(i,j)$, with different rates of fall-off.

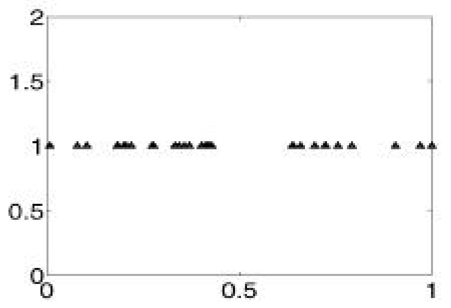


Figure 13: *Random 1-D data points*

The first weight function that we take is one with a very high rate of fall-off in weight with distance. The weight function is plotted in Figure 14 and is given by:

$$w(x) = e^{-(\frac{d(x)}{0.1})^2}$$

Due to the exponential rate of decay in weight with distance in this weighting function, only close-by points are connected. The cluster on the right has less within-group similarity compared with the cluster on the left. The results for the three algorithms are shown in Figure 15. Because of the weight function used, there are huge dissimilarities between the distant left and right clumps and thus, *Ncut* and *average cut* have no problems in partitioning the data. *Average association* on the other hand does not look at dissimilarity at all and is thus misled due to lack of tight connectivity in the clump on the right. Thus, it focuses on finding small clusters in each of the two main subgroups and fails to find the right partition.

The second weight function that we look at is one that decays very slowly with distance. It is plotted in Figure 16 and is given by:

$$w(x) = 1 - d(x)$$

With this weighting function, most points have some nontrivial connections to the rest. To find a cut of the graph, a number of edges with heavy

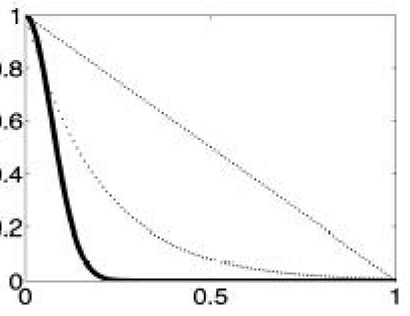


Figure 14: *Weight function that falls rapidly with distance*

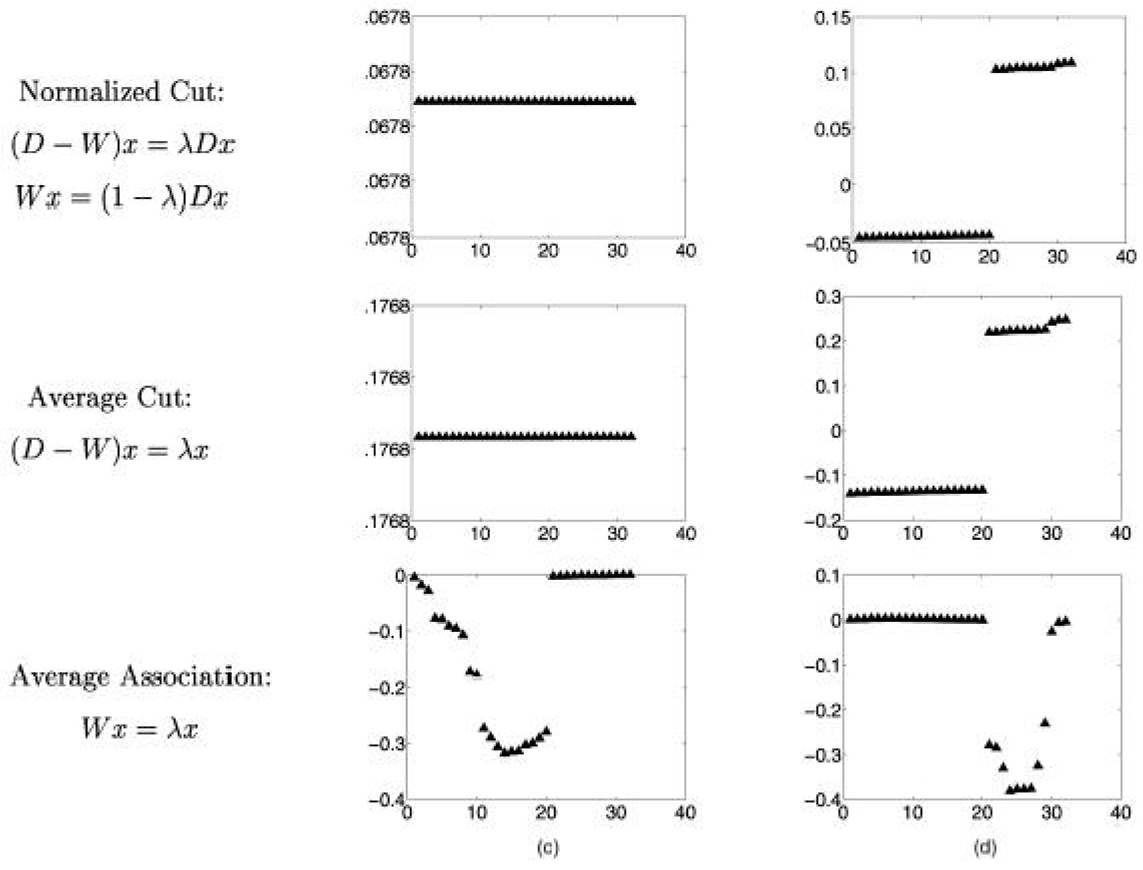


Figure 15: *Results for the three criteria with rapidly falling weight function*

weights have to be removed. The results for the three algorithms are shown in Figure 17. The left and right clumps have tight within-group similarity and so *Ncut* and *average association* separate out these two groups quite nicely. Because of the heavy weighing function there isn't too much dissimilarity between the two clumps and thus *average cut* gets confused and has trouble deciding on where to cut.

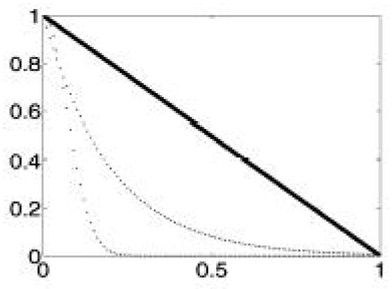


Figure 16: *Weight function that falls slowly with distance*

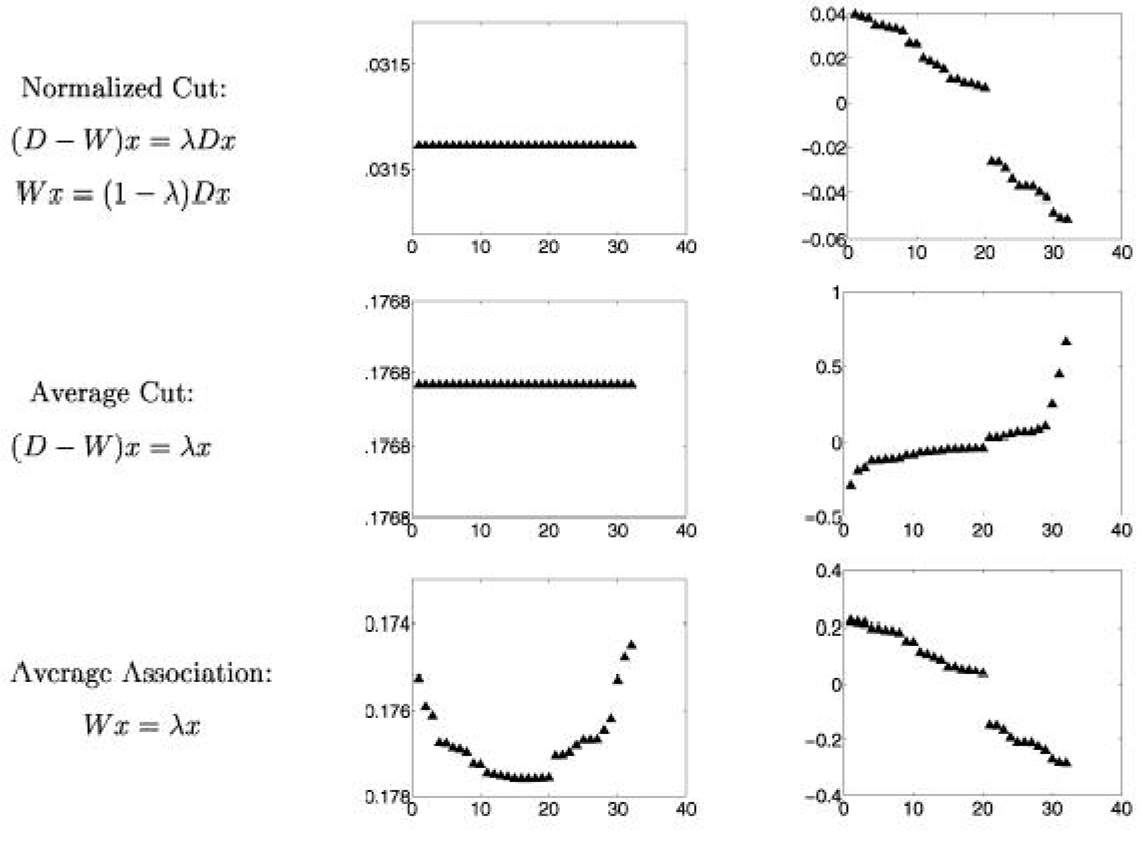


Figure 17: *Results for the three criteria with slowly varying weight function*

7 Results

The results of running the *normalized cut* algorithm are shown. The original image is on the left and the segmented output is shown on the right.

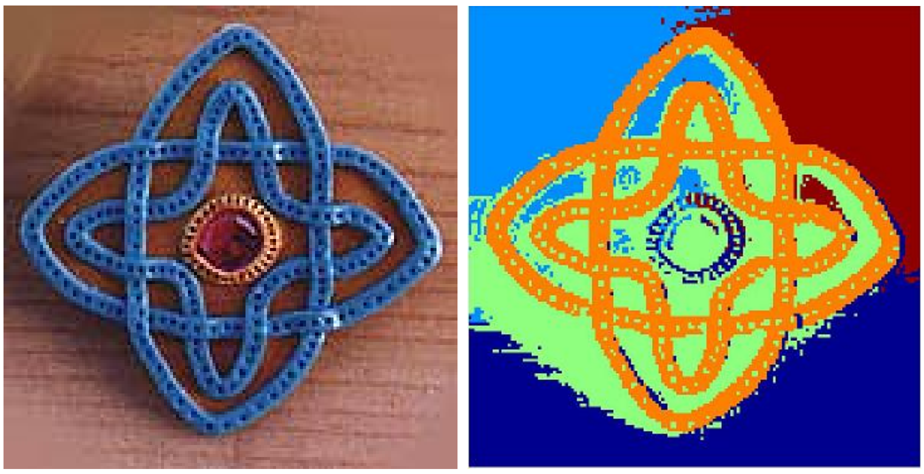


Figure 18: An image of a button and the segmented output

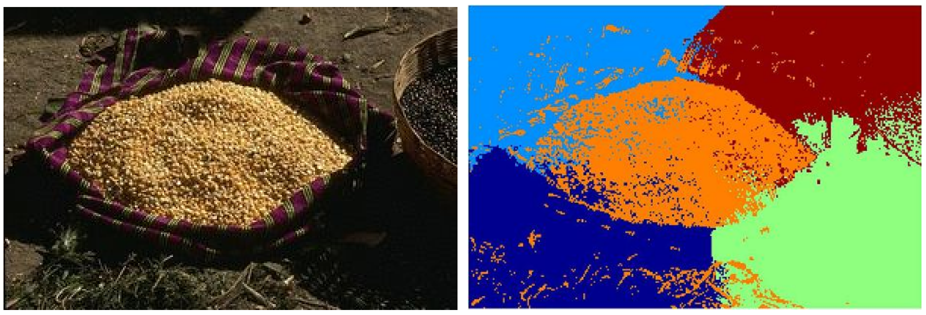


Figure 19: An image showing food grains and the segmented output

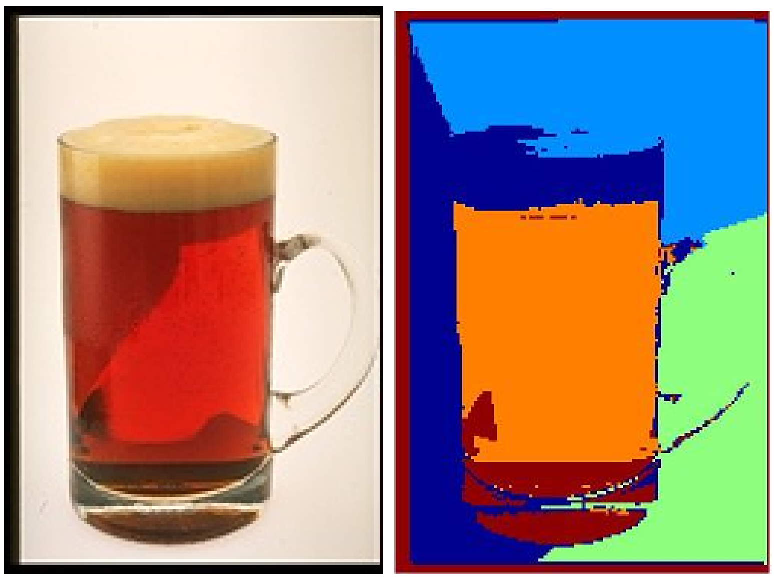


Figure 20: A glass image and the segmented output

8 Conclusion

The *normalized cut* formulation can be seen to give quite good results for image segmentation. Segmentation is an inherently subjective problem and quantitatively measuring performance of different segmentation algorithms is extremely tricky since there is no real "correct" answer to be compared with. Thus, the user should be able to parametrically control the segmentation that is achieved and this is provided for in the parameters of the weight function in all the graph theoretic formulations. Additionally, by setting a threshold on the *normalized cut* value, the number of groups segmented can also be controlled.

When comparing the other formulations like *average association* and *average cut* with *Ncut*, it was seen, both theoretically and experimentally that *Ncut* gives better results. *Average association* has a bias for finding tight clusters and it runs the risk of finding small, tight clusters in the data even though dissimilarities with other clumps should actually result in less number of clusters. On the other hand, *average cut* does not look at within-group similarity and this causes problems when the dissimilarity between groups is not very pronounced. Thus, *normalized cut* produces better results in practice because it takes into account both similarity within groups as well as dissimilarity across groups.

The problems, illustrated in Figures 15 and 17, are quite typical in segmenting real natural images. It is very difficult to predetermine the right weighting function on each image region. Therefore, it is important to design a grouping algorithm that is more tolerant to a wide range of weighting functions. The advantage of using *normalized cut* becomes more evident in this case.

However, *normalized cut* poses serious computational issues. The bottleneck in computation is in the generalized eigenvalue solution step. Even an implementation that takes advantage of all the optimizations possible (sparse matrix representations, approximate solutions, solving for only the smallest eigenvalues, etc) still runs quite slowly and becomes impractical for large images.

References

[1] M. Wertheimer, “Laws of Organization in Perceptual Forms”, A Sourcebook of Gestalt Psychology, W.B. Ellis, ed., pp. 71-88, Harcourt, Brace, 1938.

[2] S. K. Pal and N. R. Pal, “A Review on Image Segmentation Techniques”, Pattern Recognition, Vol. 26, No. 9, pp. 1277-1294, 1993.

[3] Z. Wu and R. Leahy, “An Optimal Graph Theoretic Approach to Data Clustering: Theory and Its Application to Image Segmentation”, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 15, No. 11, pp. 1101-1113, Nov. 1993.

[4] J. Shi and J. Malik, “Normalized Cuts and Image Segmentation,” IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 22, No. 8, pp. 888-905, Aug. 2000.

[5] R. C. Gonzalez and R. E. Woods. Digital Image Processing, 2nd ed., Pearson Education, 2000.

[6] S. D. Conte and C. de Boor. Elementary Numerical Analysis - An Algorithmic Approach, 3rd ed., TMH, 1986.