

# ENEE731 Project

## Normalized Cuts and Image Segmentation

Naotoshi Seo, sonots@umd.edu

November 8, 2006

### 1 Introduction

Shi and Malik (1997) [1] proposed the Normalized Cuts for image segmentation problem, which is based on Graph Theory. This algorithm treats an image pixel as a node of graph, and considers segmentation as a graph partitioning problem. The Normalized Cuts algorithm measures both the total dissimilarity between the different groups as well as the total similarity within the groups. Amazingly, the optimal solution of splitting points is easily computed by solving a generalized eigenvalue problem.

### 2 Fundamental Idea

A graph  $\mathbf{G} = (\mathbf{V}, \mathbf{E})$  can be partitioned into two disjoint sets,  $A$ ,  $B$ . The degree of dissimilarity between these two pieces can be computed as

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v). \quad (1)$$

where  $w(u, v)$  is the similarity between node  $u$  and  $v$ . The optimal bipartitioning of a graph is the one that minimizes this cut value. Finding the *minimum cut* is a well-studied problem and there exist efficient algorithms for solving it. However, the minimum cut criteria favors cutting small sets of isolated nodes in the graph, and gives bad partition in some cases such as Fig. 1.

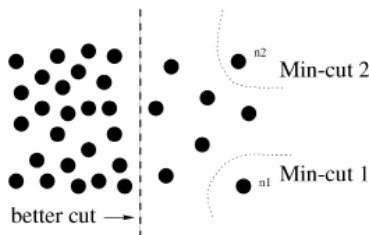


Figure 1: A case where minimum cut gives a bad partition.

Shi and Malik proposed a new measure of disassociation, the *normalized cut* (Ncut):

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

where  $assoc(A, V) = \sum_{u \in A, t \in V} w(u, t)$  is the total connection from nodes in A to all nodes in the graph and  $assoc(B, V)$  is similarly defined.

Let  $\mathbf{d}(i) = \sum_j w(i, j)$  be the total connection from the node  $i$  to all other nodes. Let  $\mathbf{D}$  be an  $N \times N$  diagonal matrix with  $\mathbf{d}$  on its diagonal,  $\mathbf{W}$  be an  $N \times N$  symmetric matrix with  $\mathbf{W}(i, j) = w(i, j)$ . Then it turns out that we can minimize  $Ncut(A, B)$  by

$$\min_{A, B} Ncut(A, B) = \min_{\mathbf{y}} \frac{\mathbf{y}^T (\mathbf{D} - \mathbf{W}) \mathbf{y}}{\mathbf{y}^T \mathbf{D} \mathbf{y}}. \quad (3)$$

If  $\mathbf{y}$  is relaxed to take real values, the above equation can be minimized by solving the generalized eigen value system,

$$(\mathbf{D} - \mathbf{W}) \mathbf{y} = \lambda \mathbf{D} \mathbf{y}. \quad (4)$$

Amazingly, the second smallest eigenvector  $\mathbf{y}$  gives the solution of the normalized cut problem.

### 3 Algorithm

Given an image sequence  $\mathbf{I}$ . Construct a weighted graph  $\mathbf{G} = (\mathbf{V}, \mathbf{E})$  whose each node is each pixel of the image  $\mathbf{I}$ . Let  $N$  be the number of nodes (pixels), i.e.,  $|\mathbf{V}|$ .

#### Step 1

Construct an  $N \times N$  symmetric similarity matrix  $\mathbf{W}$  as:

$$w_{ij} = \exp \frac{-\|\mathbf{F}(i) - \mathbf{F}(j)\|_2^2}{\sigma_I^2} * \begin{cases} \exp \frac{-\|\mathbf{X}(i) - \mathbf{X}(j)\|_2^2}{\sigma_X^2} & \text{if } \|\mathbf{X}(i) - \mathbf{X}(j)\|_2 < r \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

where  $\mathbf{X}(i)$  is the spatial location of node  $i$ , i.e., the coordinates in the original image  $\mathbf{I}$ , and  $\mathbf{F}(i)$  is a feature vector defined as:

- $\mathbf{F}(i) = 1$  for segmenting point sets,
- $\mathbf{F}(i) = \mathbf{I}(i)$ , the intensity value, for segmenting brightness (gray scale) images,
- $\mathbf{F}(i) = [v, u \cdot s \cdot \sin(h), v \cdot s \cdot \cos(h)](i)$ , where  $h, s, v$  are the HSV values, for color segmentation,
- $\mathbf{F}(i) = [| \mathbf{I} * f_1 |, \dots, | \mathbf{I} * f_n |](i)$ , where the  $f_i$  are DOOG filters at various scales and orientations, for texture segmentation.

Let  $\mathbf{d}_i = \sum_j w_{ij}$  be the total connection from node  $i$  to all other nodes.

Construct an  $N \times N$  diagonal matrix  $\mathbf{D}$  with  $\mathbf{d}$  on its diagonal.

#### Step 2

Solve a generalized eigensystem,

$$(\mathbf{D} - \mathbf{W}) \mathbf{y} = \lambda \mathbf{D} \mathbf{y}, \quad (6)$$

and get an eigenvector with the second smallest eigenvalue. Fortunately, matlab has a function, eigs, to solve generalized eigensystems.

### Step 3

Use the eigenvector to bipartition the graph. In the ideal case, the eigenvector should only take on two discrete values, and the signs tell us exactly how to partition the graph ( $\mathbf{A} = \{\mathbf{V}_i | \mathbf{y}_i > 0\}$ ,  $\mathbf{B} = \{\mathbf{V}_i | \mathbf{y}_i \leq 0\}$ ).

However,  $\mathbf{y}$  is relaxed to take real values, therefore, we need to choose a splitting point. There are several ways such as

- Take 0
- Take median
- Search a splitting point which results in that  $Ncut(\mathbf{A}, \mathbf{B})$  is minimized.

The splitting point which minimizes  $Ncut$  value also minimizes

$$\frac{\mathbf{y}^T(\mathbf{D} - \mathbf{W})\mathbf{y}}{\mathbf{y}^T\mathbf{D}\mathbf{y}} \quad (7)$$

where  $\mathbf{y} = (\mathbf{1} + \mathbf{x}) - b(\mathbf{1} - \mathbf{x})$  where  $b = k/(1 - k)$  where

$$k = \frac{\sum_{x_i > 0} \mathbf{d}_i}{\sum_i \mathbf{d}_i}$$

where  $\mathbf{x}$  is an  $N$  dimensional indicator vector,  $x_i = 1$  if node  $i$  is in  $\mathbf{A}$  and -1, otherwise.

To find the minimal  $Ncut$ , we need to try different values of splitting points. The optimal splitting point is generally around the mean value of the obtained eigenvector. Fortunately, matlab has a function, `fminsearch`, which is suitable for this purpose.

### Step 4

Repeat bipartition recursively. Stop if  $Ncut$  value is larger than a pre-specified threshold value (Large  $Ncut$  value means that there is no clear partition point any more). Furthermore, stop if the total number of nodes in the partition ( $Area$ ) is smaller than a pre-specified threshold value (this is another criteria added newly to the paper's algorithm.)

## 4 Experiments and Results

### Gray scale Image

Fig. 2 shows the result of segmentation of a gray scale image. The image was obtained from [4].

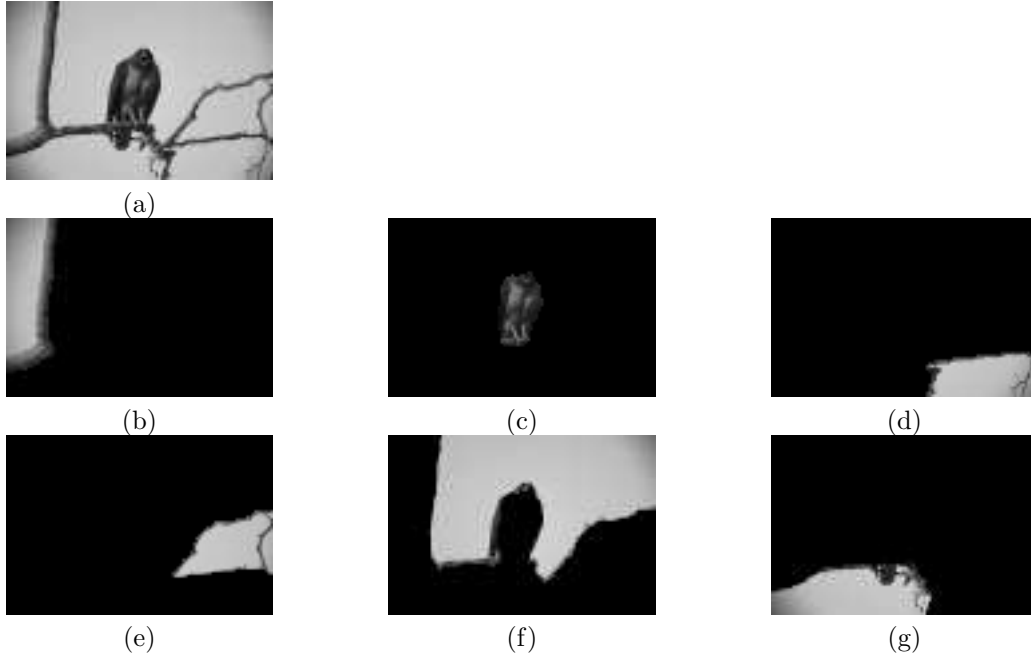


Figure 2: (a) shows the gray scale original image of size 100x67. Image intensity is normalized to lie within 0 and 255. Subplots (b)-(g) show the components of the partition with  $Ncut$  value less than 0.14,  $Area$  size more than 220. Parameter setting:  $\sigma_I = 5.0, \sigma_X = 4.0, r = 1.5$ .

### Color Image

Fig. 3 shows the result of segmentation of a color image. The image was obtained from [4].

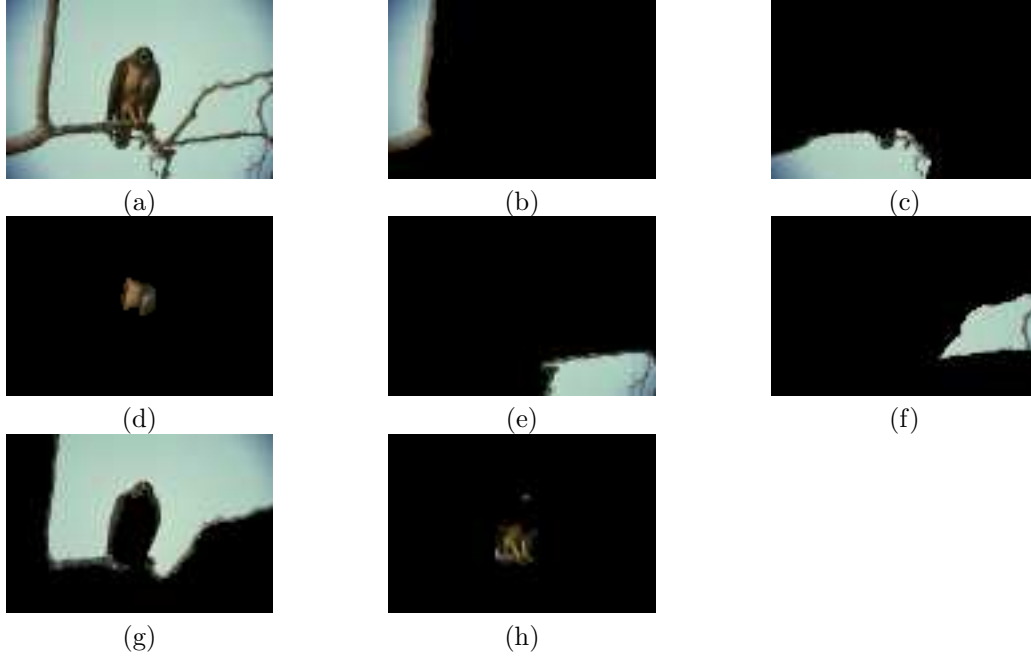


Figure 3: (a) shows the color original image of size 100x67. Each RGB intensity is normalized to lie within 0 and 255. Subplots (b)-(h) show the components of the partition with  $Ncut$  value less than 0.21,  $Area$  size more than 120. Parameter setting:  $\sigma_I = 5.0, \sigma_X = 6.0, r = 1.5$ .

### Comparison

Fig. 4 shows a comparison with a result of the program provided by Dr. Shi [3]. The input image was also obtained from [3]. The Dr. Shi's program took 77.6793 seconds for computation, but my program took 22.6640 seconds.

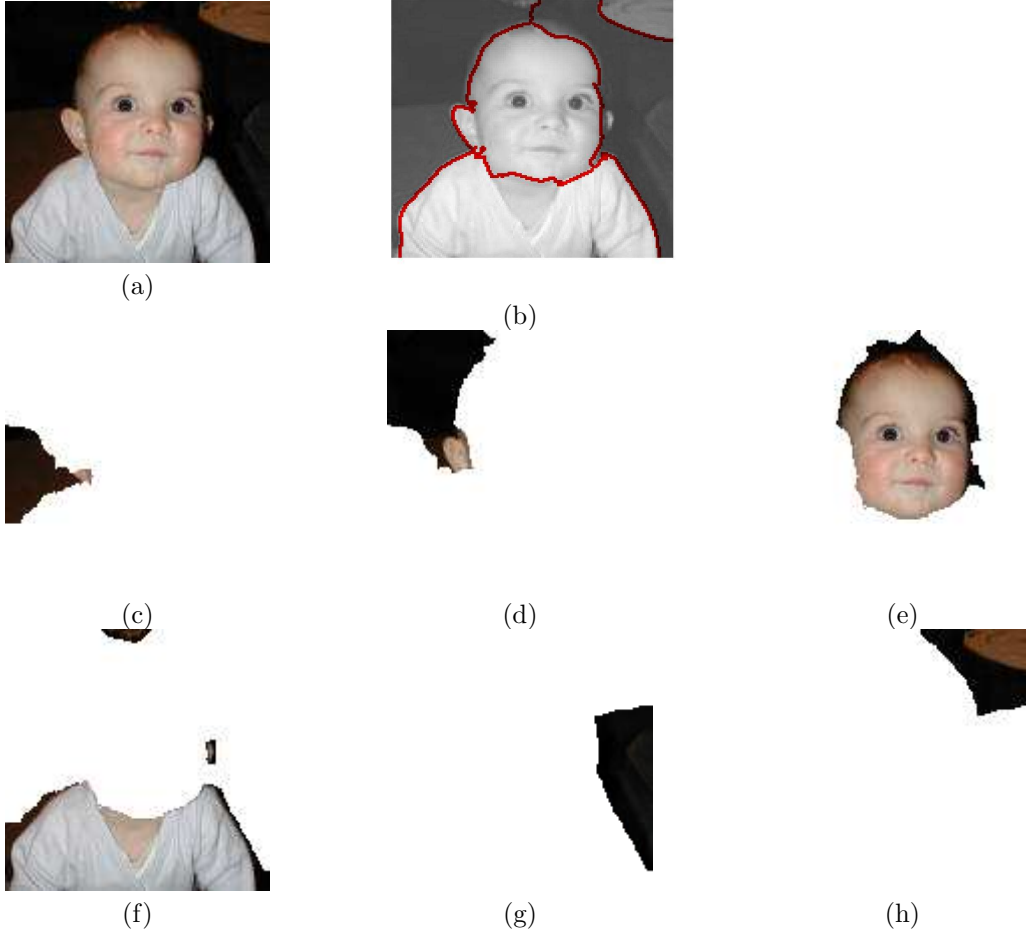


Figure 4: (a) shows the color original image of size 160x160. (b) shows the result by Dr. Shi's program. Subplots (c)-(h) show the result by my program with  $Ncut$  value less than 0.04,  $Area$  size more than 1000. Parameter setting:  $\sigma_I = 5.0, \sigma_X = 6.0, r = 1.5$ .

## 5 Discussion

My program worked reasonably fast for small images (e.g., 100 x 64), but it worked impractically for large images (e.g., 481 x 321). In particular, the calculation of eigen vectors and weight matrix required a lot of time, therefore, more efficient algorithm to compute them, and efficient tricks for memory management is required (they requires much of memory). Of course, we should consider implementing in low level languages such as C and using matlab mex, or running on powerful machines to shorten computation time, too.

My program worked faster than the program provided by Dr. Shi although his program is implemented by C and using matlab mex. It is probably because the Dr. Shi's program is more complicated or my matlab trick for fast computation worked well.

In order to obtain desired results, I had to carefully choose parameters. This is another optimization issue which must be solved in the future.

The electric version of this document and source codes is available at <http://note.sonots.com/SciSoftware/NcutIm>

## Bibliography

- [1] Jianbo Shi and Jitendra Malik, "Normalized Cuts and Image Segmentation," *IEEE Transactions on PAMI*, Vol. 22, No. 8, Aug. 2000. <http://www.cs.berkeley.edu/~malik/papers/SM-ncut.pdf>
- [2] Jianbo Shi, "Graph Based Image Segmentation Tutorial", *CVPR*, June. 2004. <http://www.cis.upenn.edu/~jshi/GraphTutorial/>
- [3] Jianbo Shi, "MATLAB Normalized Cuts Segmentation Code", 2006 (present). <http://www.cis.upenn.edu/~jshi/software/>
- [4] D. Martin and C. Fowlkes and D. Tal and J. Malik, "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics", *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, pp. 416-423, July 2001. <http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/>