Image Segmentations via Graph Cuts

Rahul Palnitkar

April 29, 2022
Version: Final Draft
Advisor: Jeova Farias Sales Rocha Neto
Abstract

In this paper, we discuss Shi and Malik’s groundbreaking Normalized Cuts approach to image segmentation, and review a number of modified Normalized Cut methods, including Mixed Normalized Cut, Semi-Supervised Normalized Cuts, and the \( gPb \) contour detector. Additionally, we discuss alternate interpretations of the Normalized Cut. After a review of previous approaches to image segmentation, we propose a new method, building off of the Normalized Cuts algorithm by constructing a new image graph which holds pixel color information. We then test our new approach, Color-Nodes Segmentation, comparing its performance and accuracy to the standard \( Ncuts \) approach pioneered by Shi and Malik. Color-Nodes Segmentation offers a more accurate segmentation for noisy images, and a five-to-seven fold improvement in runtime as compared to the classical \( Ncuts \) approach.
Acknowledgements

I would like to thank Professor Farias for being an excellent advisor, Professor Boudourides for being my second reader, and my friends and family for supporting me.
Contents

1 Introduction ................................................. 2
  1.1 Motivation ............................................. 2
  1.2 Background: What is an Image? ......................... 2
  1.3 Mathematical Background ............................... 3
    1.3.1 Graph Theory .................................... 3
    1.3.2 Eigenvectors and Eigenvalues ...................... 4
    1.3.3 Markov Chains ................................... 4
  1.4 The Structure of this Paper ............................ 4

2 Literature Review ........................................ 6
  2.1 Normalized Cuts .......................................... 6
    2.1.1 The Normalized Cut ................................ 6
    2.1.2 The Normalized Cut Algorithm ..................... 8
    2.1.3 Results ........................................... 9
  2.2 Interlude I: Sparse Matrices ............................ 9
  2.3 Interlude II: Segmentation by Random Walks ............ 10
  2.4 MixNcut: A Modified Ncut Algorithm ................... 11
    2.4.1 $G_{\text{grid}}$ ..................................... 11
    2.4.2 $G_{\text{data}}$ ..................................... 12
    2.4.3 Mixed Normalized Cuts ............................ 13
    2.4.4 Results .......................................... 14
  2.5 Semi-Supervised Normalized Cuts ....................... 14
    2.5.1 Must-link and Cannot-link Constraints ............ 15
    2.5.2 Hard Constraints vs Soft Constraints .............. 15
    2.5.3 Semi-Supervision ................................ 16
    2.5.4 Results .......................................... 17
  2.6 Contour Detection ....................................... 17
    2.6.1 Pb, an Existing Contour Detector .................. 18
    2.6.2 $mPb$, a Multiscale Contour Detector .............. 19
    2.6.3 Globalization via Normalized Cuts ................ 20
    2.6.4 Results .......................................... 21
# Color-Node Image Segmentation

## 3.1 Motivation

## 3.2 $G_{color}$

## 3.3 Methods

## 3.4 Results

- 3.4.1 Measuring Segmentation Accuracy
- 3.4.2 Experiments on Synthetic Greyscale Images
- 3.4.3 Experiments on Textured Greyscale Images
- 3.4.4 Experiments on Real Images

## 3.5 Conclusion and Future Directions
1 Introduction

1.1 Motivation

Over the last couple of decades, there has been much work done on Computer Vision and Image Recognition. Broadly defined, the goal of computer vision is to have computers be able to process images like humans—that is, to extract and analyze information from images (Huang 1996). There are a number of approaches to tackling the problem of image recognition, from standard algorithmic techniques to the more recent machine-learning approaches. In this paper, we focus on the former category of recognition approaches. One of the advantages of a non-machine learning approach, of course is the lack of reliance on former training data. While machine-learning approaches may be effective—and they are effective—they require large amounts of training data, and a large amount of computational power. Standard algorithmic techniques, such as the ones discussed in this paper, generally depend only on the input image itself and don’t require prior data. Thus, we focus on a number of effective graph-theoretic based segmentation approaches in this paper.

1.2 Background: What is an Image?

Of course, before we discuss image segmentation techniques, we first must answer an important question: what, to a computer, is an image? Let $I$ be an image of $n \times m$ pixels. One can think of $I$ as an $n \times m$ matrix $A$, where each entry in $A$ corresponds to the color value of a pixel in $I$. Furthermore, one can construct a weighted graph representation $G = (V, E)$ of $I$, where node $v \in V$ if pixel $p \in I$ (Shi...
and Malik 2000). Edges are drawn between every node (making $G$ fully connected, and given weights according to the color similarity and distance between two nodes (Shi and Malik 2000). The image segmentation algorithms we review in this paper then attempt to partition this graph representation of the image in order to segment the image.

### 1.3 Mathematical Background

Before we review image segmentation techniques, we first briefly review some brief mathematical topics, including graphs, eigenvalues and eigenvectors, and Markov Chains, as they show up repeatedly in our exploration of image segmentation approaches.

#### 1.3.1 Graph Theory

Recall that a graph $G = (V, E)$ describes a relation between a set of vertices, or nodes, $V$, and a set of edges $E$, where for an edge $e \in E$, $e = (u, v)$ for $u, v \in V$ (Kleinberg and Tardos 2014). A weighted graph is a graph where every edge has a numerical weight, or cost, associated with it (Kleinberg and Tardos 2014).

**The Minimum Cut**

The cut of a graph $G$ is a partitioning of its vertices $V$ into disjoint subsets $A$ and $B$. Given a partitioned, or cut, graph into subsets $A$ and $B$, we can define $\text{cut}(A, B)$ as the sum of the edge weights exiting $A$ and entering $B$ (Shi and Malik 2000). Formally,

$$\text{cut}(A, B) = \sum_{a \in A, b \in B} w_{ab}. $$

As Shi and Malik note, $\text{cut}(A, B)$ is a useful method of determining dissimilarity between partitions $A$ and $B$ (Shi and Malik 2000).

The min-cut, or min-cut, problem is the problem of partitioning $G$ into subsets $A$ and $B$ such that $\text{cut}(A, B)$ is minimized. There are a number of algorithms for
solving this problem; Karger provides an edge-contraction method for finding the minimum cut of both weighted and unweighted graphs in (Karger 1993).

1.3.2 Eigenvectors and Eigenvalues

Our definitions are adapted from (Axler 1997).

**Definition 1.** (Axler 1997) Let \( A \) be a matrix. A real number \( \lambda \) is called an **eigenvalue** of \( A \) if there exists a non-zero vector \( v \) such that \( Av = \lambda v \).

**Definition 2.** (Axler 1997) Let \( A \) be a matrix and \( \lambda \) be an eigenvalue of \( A \). Then an **eigenvector** of \( A \) is a non-zero vector \( v \) such that \( Av = \lambda v \).

1.3.3 Markov Chains

A couple of the papers discussed below use the concept of Markov Chains, and as such, we define this term below, adapting a definition from (Weisstein n.d.):

**Definition 3.** (Weisstein n.d.) A **Markov Chain** is a sequence of random variables \( \{X_t\} \) where if \( \{i_t\} \) is a set of states,

\[
P(X_t = j | X_0 = i_0, X_1 = i_1, \cdots, X_{t-1} = i_{t-1}) = P(X_t = j | X_{t-1} = i_{t-1})
\]

1.4 The Structure of this Paper

In the next section, we review a number of image segmentation papers, starting with Shi and Malik’s *Normalized Cuts and Image Segmentation* (Shi and Malik 2000). In section 2.2, we discuss the use of sparse matrices in \( Ncut \)-based image segmentation. In section 2.3, we discuss a stochastic interpretation of the Normalized Cut, an interpretation used in the section 2.4, where we review Rocha Neto and Felzenszwalb’s *MixNcut* approach in (Rocha Neto and Felzenszwalb 2020). In
section 2.5, we discuss Chew and Cahill’s semi-supervised approach to Normalized Cuts in (Chew and Cahill 2015), and in 2.6, we discuss the application of the $Ncut$ approach to contour detection in (Arbeláez et al. 2011).

We then propose a new segmentation method, Color-Nodes Segmentation, building off of the Normalized Cuts approach, where we aim to build a graph that holds color information, allowing for long-distance pixels to be contained in the same region. In section 3.2, we discuss the theoretical construction of such a graph, and in section 3.3, we test the proposed segmentation on a number of images, comparing it to existing methods. Finally, we conclude with a discussion of the proposed method and potential future directions.
2

Literature Review

2.1 Normalized Cuts

In their seminal paper *Normalized Cuts and Image Segmentation*, Shi and Malik introduce the concept of a normalized graph cut, and further describe an effective image segmentation algorithm using the normalized cut to separate pixels into clusters (Shi and Malik 2000).

2.1.1 The Normalized Cut

Recall that an image $I$ of $n \times m$ pixels can be represented as a weighted, fully connected graph $G(V, E)$, where $|V| = nm$. In (Shi and Malik 2000), Shi and Malik define the weight $w_{ij}$ between two nodes $i$ and $j$ as

$$w_{ij} = e^{-\frac{||F(i) - F(j)||^2}{\sigma_I^2}} \times \begin{cases} e^{-\frac{||X(i) - X(j)||^2}{\sigma_X^2}} & ||X(i) - X(j)|| < r, \\ 0 & \text{otherwise} \end{cases}, \quad (2.1)$$

where $F(i)$ is the color value of pixel $i$, and $X(i)$ is the Euclidean position of pixel $i$ in image $I$ (Shi and Malik 2000). Now, in order to properly segment the image into two (and possibly more) parts, $G$ is partitioned, or cut, into sets $A$ and $B$, where $A \cup B = V$.

Since nodes that have similar color values and/or are close together (in terms of Euclidean distance apart) have higher edge weights, it makes sense to want to find a cut of $G$ into $A$ and $B$ that minimizes $\text{cut}(A, B)$—that is, we would want to find the min-cut of $G$. However, as Shi and Malik note, trying to minimize $\text{cut}(A, B)$ usually
results in an unbalanced partition, where, for example, \( A \) contains most nodes and \( B \) a very small number, or even an individual node, resulting in an incorrect or incomplete segmentation of the image (Shi and Malik 2000).

In order to remedy this, Shi and Malik propose type of cut, called the normalized cut or \( Ncut \): (Shi and Malik 2000)

\[
Ncut(A, B) = \frac{\text{cut}(A, B)}{\text{Vol}(A, V)} + \frac{\text{cut}(A, B)}{\text{Vol}(B, V)}.
\]

The volume of \( S \subset V \) and \( V \), also known as the association, is defined in (Shi and Malik 2000) as:

\[
\text{Vol}(S, V) = \sum_{s \in A, v \in V} w_{sv}.
\]

In (Shi and Malik 2000), Shi and Malik use the terms association and \( \text{assoc}(A, V) \); here we use the term volume.

In order to minimize the \( Ncut \), two matrices \( D \) and \( W \) are constructed, where \( W \) is the adjacency matrix of \( G \), and \( D \) is the diagonal matrix where the \( i \)th diagonal entry \( d_i \) is given by

\[
d_i = \sum_{j \in V} w_{ij}.
\]

Then, the Laplacian matrix \( L \) of \( G \) is defined as the difference between \( D \) and \( W \):

\[
L = D - W.
\]

Using the Laplacian matrix \( L \), Shi and Malik show that minimizing the \( Ncut \) is equivalent to solving the generalized eigensystem

\[
Ly = \lambda Dy,
\]

where \( y \) is a real-valued vector \(^1\) (Shi and Malik 2000). Now, we can turn the above generalized eigensystem into a normal eigensystem by using the fact that \( D \) is

\(^1\)Ideally \( y \) is comprised only of \( 1 \)s and \( -1 \)s, allowing us to partition \( G \) easily.
symmetric and positive semidefinite, and as such, \( D^{\frac{1}{2}} \) exists. The above generalized eigensystem becomes the normal eigensystem

\[
D^{-\frac{1}{2}}LD^{-\frac{1}{2}}z = \lambda z,
\]

(2.7)

where \( z = D^{\frac{1}{2}}y \) (Shi and Malik 2000). Shi and Malik show that the second-smallest eigenvector of \( Ly = \lambda Dy \) yields the proper normalized cut (Shi and Malik 2000).

To illustrate how \( y \) might partition a graph \( G \), suppose we have a \( 2 \times 2 \) image \( I \). Then, suppose, following the above calculations, we found the second-smallest eigenvector of the generalized eigensystem to be

\[
y = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}.
\]

Then, we would send nodes 1 and 3 into one subgraph of \( G \), and nodes 2 and 4 into another subgraph. In this way, we have normalized-cut \( G \).

### 2.1.2 The Normalized Cut Algorithm

After defining the normalized cut of a graph, and determining methods of finding the normalized cut, Shi and Malik then describe an algorithm to effectively segment images using the normalized cut method. Generally, given an image, the algorithm creates a graph, uses the normalized-cut method to partition the graph, and then repeats the partitioning processes as needed (Shi and Malik 2000). More formally, the algorithm is given below:

**Algorithm: Normalized Cuts (Ncut) (Shi and Malik 2000)**

1. Given an \( m \times n \) image \( I \), create a graph \( G = (V, E) \) where \( |V| = mn \) and \( |E| = (mn)^2 \).

2. Calculate diagonal matrix \( D \) and adjacency matrix \( W \) of \( G \)

3. Solve the eigensystem \( Ly = \lambda Dy \)
4. Using the eigenvector with the second smallest eigenvalue, partition $G$.

5. Repeat the process by calling this algorithm on the subgraphs of $G$ as necessary.

2.1.3 Results

As Shi and Malik show in (Shi and Malik 2000), the above algorithm is effective at segmenting greyscale images. In Figure 2.1, the $Ncut$ algorithm is able to segment the image of a person, differentiating the head from torso, and background from foreground. (Shi and Malik 2000) Additionally, as Shi and Malik note, the $Ncut$ algorithm has a time complexity of $O(nm)$, where $n$ is the number of pixels in $I$ (equivalently, $n = |V|$), and $m$ represents the convergence time of the Lanczos algorithm for calculating eigenvalues/eigenvectors (Shi and Malik 2000).

In the next few sections, we explore a number of proposed methods to speed up the $Ncut$ algorithm and make the $Ncut$ algorithm more effective, as well as alternative methods of cutting $G$.

2.2 Interlude I: Sparse Matrices

One of many ways to speed up the $Ncut$ algorithm is to have the matrix $L$ be sparse. Informally, a sparse matrix is a matrix where most entries are zero (Gilbert et al.
For the purposes of this paper, we define a sparse matrix to be any matrix where the number of non-zero entries are linear with the number of rows/columns. Due to the small number of non-zero elements, sparse matrices are easier to store (taking up less space) and are faster to run calculations on (Gilbert et al. 1992). Thus, we see that a number of $Ncut$ methods use the properties of sparse matrices to help speed up the algorithm.

### 2.3 Interlude II: Segmentation by Random Walks

In their paper *Learning Segmentation by Random Walks*, Melia and Shi provide a probabilistic interpretation of the Normalized Cut, allowing for a more intuitive notion of the $Ncut$ method (Meila and Shi 2001).

Let $W$ be the adjacency matrix of the graph $G = (V, E)$ of an image $I$ and $D$ be the diagonal matrix of edge-weight sums, as described in (Shi and Malik 2000). Then, Melia and Shi "normalize" $W$, creating a stochastic matrix $P$:

$$P = D^{-1}W$$  \hfill (2.8)

Importantly, each row of $P$ sums to 1, and the element $P_{ij}$ is interpreted as the probability that a "walker" moves from node $i$ to node $j$ (Meila and Shi 2001). Melia and Shi then propose that if $\lambda$ is an eigenvalue, and $x$ is an eigenvector, of the system

$$Px = \lambda x,$$  \hfill (2.9)

then $(1 - \lambda)$ is an eigenvalue, and $x$ is an eigenvector of the system $Lx = \lambda Dx$ (Meila and Shi 2001).

The more interesting consequence of $P$, however, is the aforementioned probabilistic interpretation of Normalized cuts. Melia and Shi define the vector

$$\pi^\infty = [\pi^\infty_v]_{v \in V}$$  \hfill (2.10)

where $\pi^\infty_v = \frac{d_v}{\text{vol}(V)}$ and the elements $d_v$ are the diagonal elements of $D$ (Meila and Shi 2001). Note that as $P^T \pi^\infty = \pi^\infty$, $\pi^\infty$ is considered to be a stationary distribution of a Markov chain (Meila and Shi 2001). Then, with this stationary distribution
defined, Melia and Shi define the probability, given two subsets \( A \) and \( B \) of \( V \), of moving from one set to the other, \( P_{AB} \), as

\[
P_{AB} = \frac{\sum_{i \in A, j \in B} \pi_i^{\infty} P_{ij}}{\pi^{\infty}(A)} = \frac{\sum_{i \in A, j \in B} W_{ij}}{vol(A,V)}.
\]  

(2.11)

(Meila and Shi 2001) But then, it is clear to see that the above is just equal to \( \frac{cut(A,B)}{vol(A,V)} \). Thus, Melia and Shi show that

\[
Ncut(A,B) = P_{AB} + P_{BA},
\]  

(2.12)

that is, the value of \( Ncut(A,B) \) is the probability of walking to \( B \) given a starting position in \( A \) plus the probability of walking to \( A \) starting in \( B \) (Meila and Shi 2001). Intuitively, then, it makes sense to want to minimize \( Ncut \)—in doing so, you minimize the chance of a pixel belonging to one segment accidentally being included in the other set.

A number of these ideas, particularly the stochastic matrix \( P \), inform the work done by Rocha Neto and Felzenszwalb discussed in the next section.

2.4 \textbf{MixNcut: A Modified Ncut Algorithm}

In their paper \textit{Spectral Image Segmentation with Global Appearance Modeling}, Rocha Neto and Felzenszwalb aim to enhance the existing Normalized Cut method by accounting for relationships between pixels that have otherwise similar features, but are separated a great distance (Rocha Neto and Felzenszwalb 2020). To accomplish this, they introduce two new graphs, that detail the relationship between pixels, and then combine the information obtained by the two graphs by calculating a minimum modified Normalized Cut of the two (Rocha Neto and Felzenszwalb 2020).

2.4.1 \( G_{\text{grid}} \)

The first of the two graphs constructed, \( G_{\text{grid}} \), emphasizes the relationship between neighboring pixels by connecting each pixel \( (x,y) \) to its neighboring pixels. For each
pixel, edges of weight 1 are drawn between the pixel and its four neighbors, and no other edges are created (Rocha Neto and Felzenszwalb 2020). Thus, we note that $G_{\text{grid}}$ is not fully connected, unlike the standard graph $G$ in (Shi and Malik 2000).

Now, Rocha Neto and Felzenszwalb observe two key facts about graph cuts and $G_{\text{grid}}$. Firstly, the cut of a $G_{\text{grid}}$ into subsets $A$ and $B$ is approximately the length of the boundary between the two regions. Mathematically,

$$\text{cut}(A, B|G_{\text{grid}}) \approx \text{Len}(\Gamma),$$

where $\Gamma$ is the boundary between $A$ and $B$. (Rocha Neto and Felzenszwalb 2020)

Then, they also note that given a subgraph $S$ of $G_{\text{grid}}$, $\text{Vol}(S|G_{\text{grid}}) \approx 4|S|$ (Rocha Neto and Felzenszwalb 2020).

Then, using the above observations, they derive an expression for approximating the value of $N\text{cut}(G_{\text{grid}})$ as follows:

$$N\text{cut}(A, B|G_{\text{grid}}) \approx \frac{|V|}{4} \frac{\text{Len}(\Gamma)}{|A||B|},$$

(2.14)

where $|V|$ is the number of vertices of $G_{\text{grid}}$ (Rocha Neto and Felzenszwalb 2020). It is clear to see here that in order to minimize $N\text{cut}(A, B|G_{\text{grid}})$, one should minimize $\text{Len}(\Gamma)$, and maximize $|A||B|$ (and as such, $A$ and $B$ should be of similar size).

### 2.4.2 $G_{\text{data}}$

The second of the two constructed graphs, $G_{\text{data}}$, is more similar to the graph $G$ described in (Shi and Malik 2000), but location information is not considered. Then, $G_{\text{data}}$ is a fully connected graph where the edge weight $w_{ij}$ between nodes $i$ and $j$ is defined as

$$w_{ij} = e^{-\frac{|I(i) - I(j)|^2}{2\sigma^2}},$$

(2.15)

where $I$ is a function that returns the appearance value of an input pixel, and $\sigma$ is a constant (Rocha Neto and Felzenszwalb 2020).

Then, Rocha Neto and Felzenszwalb introduce the notion of using kernel density estimates of subgraphs to calculate $\text{cut}$ and $N\text{cut}$ values for $G_{\text{data}}$. 
In particular, let $V$ denote the set of pixels of $G_{\text{data}}$, let $S \subset V$, and let $g_S$ be the kernel density estimate of pixels in $S$ such that

$$g_S(c) = \frac{1}{|S|} \sum_{i \in S} K(I(i) - c),$$

(2.16)

where $K$ is a Gaussian kernel and $c$ is a fixed point.

Rocha Neto and Felzenszwalb demonstrate that

$$\text{cut}(A, B|G_{\text{data}}) = (2\pi\sigma^2)^{\frac{d}{2}} |A||B| \langle g_A, g_B \rangle,$$

(2.17)

where $\langle g_A, g_B \rangle$ denotes the inner product of $g_A$ and $g_B$, and $d$ is the number of appearance features of the image. (Rocha Neto and Felzenszwalb 2020)

Then, with $\text{cut}$ defined in terms of Kernel density estimates, $N\text{cut}(A, B|G_{\text{data}})$ is defined such that

$$N\text{cut}(A, B|G_{\text{data}}) = \langle g_V, g_V \rangle \frac{\langle g_A, g_B \rangle}{\langle g_A, g_V \rangle \langle g_B, g_V \rangle},$$

(2.18)

where $V$ is the set of all pixels of the image (Rocha Neto and Felzenszwalb 2020).

### 2.4.3 Mixed Normalized Cuts

With $N\text{cut}$ described for both $G_{\text{grid}}$ and $G_{\text{data}}$, a combined Normalized Cut, $MixN\text{cut}$, is computed, where

$$MixN\text{cut}(A, B) = (1 - \lambda)NCut(A, B|G_{\text{data}}) + \lambda NCut(A, B|G_{\text{grid}}),$$

(2.19)

where $\lambda \in [0, 1]$ is a constant (Rocha Neto and Felzenszwalb 2020). Importantly, by increasing $\lambda$, more importance is given to $G_{\text{grid}}$, while decreasing $\lambda$ gives more importance to $G_{\text{data}}$. Thus, choosing $\lambda$ allows for relative weighting of the two graphs (Rocha Neto and Felzenszwalb 2020).

Then, using the Markov Chain interpretation of Normalized Cuts discussed in (Meila and Shi 2001), matrices $P_1 = D_1^{-1} W_1$ and $P_2 = D_2^{-1} W_2$ are constructed, where $W_1$
2.4.4 Results

When compared to the standard \textit{NCuts} approach, the \textit{MixNcut} method appears to be more effective at segmenting images. In Figure 2.2, it is apparent that the segmentations found using \textit{MixNcut} are less noisy than segmentations found using \textit{Ncut}, and appear to focus on more pertinent regions of the image. Additionally, Rocha Neto and Felzenszwalb show that the \textit{MixNcut} approach is faster than the standard \textit{Ncut} process (Rocha Neto and Felzenszwalb 2020).

2.5 Semi-Supervised Normalized Cuts

While the \textit{Ncut} algorithm is quite effective at segmenting certain images, it relies on both color of and distance between pixels in order to segment images. If two close-by pixels are of the same color, they are more likely to be placed in the same cluster, and if their colors differ a great deal, they are more likely to be placed in
different clusters. However, what happens if close-by pixels are of the same color, but are known to be different parts of the image?

For example, consider an image of a white-furred arctic fox on a snowy background. Intuitively, we know that the fox and snow are two separate things, and so the image should be segmented as such.

In their paper *Semi-Supervised Normalized Cuts for Image Segmentation*, Chew and Cahill propose an algorithm that makes use of prior constraints on pixel-linking/separation and the \( \text{Ncut} \) algorithm to better segment images (Chew and Cahill 2015).

### 2.5.1 Must-link and Cannot-link Constraints

Chew and Cahill first begin with a discussion of two types of constraints that can be introduced to limit how pixels are clustered: must-link and cannot-link constraints. If two pixels are determined to be must-link pixels, they should not be separated into different subgraphs when the cutting algorithm is run (Chew and Cahill 2015). On the other hand, if two pixels are determined to be cannot-link pixels, they should not be together in the same subgraph after the cutting algorithm is run (Chew and Cahill 2015). In this way, one is able to prevent similarly-colored objects from incorrectly being put together in the same segment.

### 2.5.2 Hard Constraints vs Soft Constraints

The problem with hard (that is to say, required) constraints, however, is that it may be impossible to satisfy all constraints (Chew and Cahill 2015). Thus, Chew and Cahill introduce the notion of *soft constraints*, adding cost functions to the \( \text{Ncut} \) definition. In particular, for must-link constraints, the new Normalized Cut formula, denoted \( \text{Ncut}_{\text{ML}} \), is defined as:

\[
\text{Ncut}_{\text{ML}}(A, B) = \frac{\text{cut}_{\text{ML}}(A, B)}{\text{Vol}(A, V)} + \frac{\text{cut}_{\text{ML}}(A, B)}{\text{Vol}(B, V)}, \tag{2.21}
\]
where
\[
\text{cut}_{ML}(A, B) = \text{cut}(A, B) + \frac{1}{2} \sum_{k=1}^{m} \gamma_k \cdot \theta(v_{i_k}, v_{j_k}), \tag{2.22}
\]
\[
\{(v_{i_k}, v_{j_k}) | k = 1, \ldots, m\}
\]
is the set of pairs of must-link pixels, \(\gamma_k\) is the weight of constraint \(k\), and where \(\theta(v_i, v_j) = 0\) if \(v_i\) and \(v_j\) are in the same graph, or 1 if \(v_i\) and \(v_j\) are in different graphs (Chew and Cahill 2015).

We see then, that when including must-link constraints, our goal is not only to minimize \(Ncut\), but also minimize the above cost function.

Chew and Cahill define the modified Normalized Cut formula for cannot-link constraints analogously. In particular, the cannot-link cut formula is given as
\[
\text{cut}_{CL}(A, B) = \text{cut}(A, B) + \frac{1}{2} \sum_{k=1}^{m} \tilde{\gamma}_k \cdot \tilde{\theta}(v_{\tilde{i}_k}, v_{\tilde{j}_k}), \tag{2.23}
\]
where \(\tilde{\theta}(v_i, v_j) = 1 - \theta(v_i, v_j)\) (Chew and Cahill 2015).

Combining the modified Normalized Cut formulas for must-link and cannot-link constraints, Chew and Cahill introduce their Semi-Supervised Normalized Cut formula, \(Ncut_{SS}\):
\[
Ncut_{SS}(A, B) = \frac{\text{cut}_{SS}(A, B)}{\text{Vol}(A, V)} + \frac{\text{cut}_{SS}(A, B)}{\text{Vol}(B, V)}, \tag{2.24}
\]
where \(\text{cut}_{SS}(A, B) = \text{cut}_{ML}(A, B) + \text{cut}_{CL}(A, B) - \text{cut}(A, B)\) (Chew and Cahill 2015).

2.5.3 Semi-Supervision

But what is the Semi-Supervised part of “Semi-Supervised Cuts?” In this case, it refers to marking some subset of pixels in one group, and another subset of pixels in another. Then, within the groups, the pixels are must-link, and pixels in the first group are "cannot-link" with pixels in the second group (Chew and Cahill 2015). In other words, only some of the pixels are labeled, as compared to an unsupervised
approach (like the standard $Ncut$ algorithm) where pixels are entirely unlabeled, or a supervised approach where every pixel would have a label indicating the region it belongs to.

### 2.5.4 Results

Through their experiments, Chew and Cahill show that this approach to image segmentation is quite effective, given, of course, some prior knowledge about pixel relations (Chew and Cahill 2015). In Figure 2.3, it is clear from the resulting eigenvectors that a semi-supervised approach, where some data is labeled and where relations between pixels can be defined, can offer better performance than a fully unsupervised approach. At the same time, when compared to a hard-constrained $Ncut$ approach, the soft-constraints approach introduced by Chew and Cahill appears to reduce noise and improve clarity of the eigenvector image (Chew and Cahill 2015).

### 2.6 Contour Detection

In addition to image segmentation problems, the Normalized Cuts method also finds use in the closely related field contour detection problems, and indeed, Arbeláez et al. use a Normalized Cuts approach to accurately differentiate the contours of a given image (Arbeláez et al. 2011).
Before discussing that paper, however, we first briefly define what an image contour is. Broadly speaking, we can think of the contours of an image as the boundaries of the various sections of an image. Given an image of a ball on a blank background, for example, the outline of the ball would be a (and this case, the only) contour of the image. Furthermore, we can think of a closed contour as a contour that has no breaks in the line—in other words, one can start and end at the same point on the contour by following its path. One can easily appreciate the close relationship between the problems of finding contours and segmenting images, as perfect, closed contours mark the boundary between different segments of an image.

Indeed, in their paper *Contour Detection and Hierarchical Image Segmentation*, Arbeláez et al. demonstrate how the problem of accurate image segmentation can be reduced to a contour-finding problem (Arbeláez et al. 2011). In this review, we primarily focus on Arbeláez et al.’s proposed contour detection method, \( g\Phi \) (Arbeláez et al. 2011).

### 2.6.1 \( Pb \), an Existing Contour Detector

Arbeláez et al. start with an existing contour detection function, \( Pb(x, y) \), which determines the probability that any given pixel \((x, y)\) is part of a contour (Arbeláez et al. 2011).

Central to calculating \( Pb \) is the function \( G(x, y, \theta) \), which computes the gradient signal at a point \((x, y)\) with an orientation of \( \theta \). More specifically, a disk of specified radius is centered on \((x, y)\), and then divided into two semi-circles, with the dividing line being a line with angle \( \theta \) (Arbeláez et al. 2011). Histograms \( g \) and \( h \) of the intensity values of the pixels covered by the semicircle are then created for each semicircle. Then

\[
G(x, y, \theta) = \chi^2(g, h) = \frac{1}{2} \sum_i (g(i) - h(i))^2 \frac{g(i)}{g(i) + h(i)},
\]

(2.25)

where \( i \) indexes the bins of histograms \( g \) and \( h \) (Arbeláez et al. 2011). After calculating \( G(x, y, \theta) \), additional processing and filtering is performed (Arbeláez et al. 2011).
When running the \( Pb \) contour detector, images are divided into four channels—brightness, color a, color b, and texture—and then \( G(x, y, \theta) \) is calculated for each pixel in each channel, and finally, the results are recombined (Arbeláez et al. 2011).

### 2.6.2 \( mPb \), a Multiscale Contour Detector

Arbeláez et al. then introduce a modification to the existing \( Pb \) detector, observing the gradient signal for the brightness and color channels at multiple scales \([\frac{\sigma}{2}, \sigma, 2\sigma]\) (Arbeláez et al. 2011). Note that \( \sigma \) denotes the radius of the circle centered on the pixel of interest, and \( \sigma \) can differ between channels. In (Arbeláez et al. 2011), for example, \( \sigma = 5 \) pixels is used for the brightness channels, whereas \( \sigma = 10 \) pixels is used for the color channels.

Then, taking this into account, Arbeláez et al. describe the multiscale \( Pb \) detector, where

\[
mPb(x, y, \theta) = \sum_{s} \sum_{i} \alpha_{i,s} G_{i,\sigma(i,s)}(x, y, \theta),
\]

(2.26)

and where \( s \in \{\frac{\sigma}{2}, \sigma, 2\sigma\} \), and \( i \) indexes the feature channels described above (Arbeláez et al. 2011). \( G_{i,\sigma(i,s)}(x, y, \theta) \), somewhat like in \( Pb \), computes the gradient signal at a point \((x, y)\) by forming a circle with radius \( \sigma(i,s) \) and dividing it into two half disks (with the dividing line angled at \( \theta \)) and measuring the difference in intensity histograms between the two halves, and \( \alpha_{i,s} \) is a learned weight (Arbeláez et al. 2011).

Then, once \( mPb(x, y, \theta) \) has been computed for a pixel \((x, y)\), the angle \( 0 \leq \theta \leq \pi \) is taken such that \( mPb(x, y, \theta) \) is maximized. Thus

\[
mPb(x, y) = \max_{\theta} \{mPb(x, y, \theta)\}
\]

(2.27)

is computed for every pixel \((x, y)\) in the image (Arbeláez et al. 2011). The results of \( mPb \) for the various channels can be seen in Figure 2.4. Note that different channels emphasize different contours detected by \( mPb \).
2.6.3 Globalization via Normalized Cuts

While $mPb$ is on its own an effective contour detector, it only operates locally—that is, $mPb(x, y)$ determines the probability that any given pixel is a boundary/contour point, but only given information about the pixel. In other words, it does not consider how the various pixels relate to each other. To remedy this, Arbeláez et al. introduce a spectral approach to contour detection, allowing for global contour consideration (Arbeláez et al. 2011). In particular, the Normalized Cuts approach, with a few modifications, is used to cluster pixels. We detail this procedure now:

- First, an adjacency matrix $W$ is constructed, where
  \[ W_{ij} = e^{-\max_{p \in 

\text{ij}} \{mPb(p)\}/\rho} \]
  and where $p$ is a pixel in the line segment of fixed radius $\overline{ij}$ which connects pixels $i$ and $j$, and $\rho$ is a constant. Note that $W$ is symmetric, and perhaps more importantly, $W$ is sparse (Arbeláez et al. 2011).

- Next, the diagonal matrix $D$ is constructed such that $D_{ii} = \sum_j W_{ij}$

- Finally, the Normalized Cuts algorithm from (Shi and Malik 2000) is used to find the first $n$ generalized eigenvectors corresponding to the $n$ smallest eigenvalues of the eigensystem $Lv = \lambda Dv$, where the Laplacian matrix $L = D - W$.

Once the eigenvectors $v_0, \cdots, v_n$ of the above system are found, Arbeláez et al. observe that the eigenvectors themselves carry information about contours (Arbeláez et al. 2011). In particular, each eigenvector $v_k$ is then treated as an image, and then the gradient $\nabla v_k(x, y)$ of each eigenvector is taken for each pixel $(x, y)$ at multiple
angles \( \theta \). Finally, Arbeláez et al. define a new contour detector, \( sPb \), that uses the eigenvector gradients described above:

\[
sPb(x, y, \theta) = \sum_{k=0}^{n} \frac{1}{\sqrt{\lambda_k}} \cdot \nabla_{\theta} v_k(x, y),
\]

(2.29)

where \( \lambda_k \) is the eigenvalue that corresponds to eigenvector \( v_k \) (Arbeláez et al. 2011). As with \( mPb \), \( \theta \) is taken such that \( sPb(x, y, \theta) \) is maximized (Arbeláez et al. 2011).

It is easy to see that \( sPb \) too is a useful contour detector. In Figure 2.5, originally taken from Arbeláez et al.’s results in (Arbeláez et al. 2011), we see that the eigenvectors of the generalized eigensystem \( L v = \lambda D v \) rather accurately highlight contours, and each eigenvector focuses on a different subset of the set of image contours.

Finally, with \( mPb \) and \( sPb \) defined, Arbeláez et al. define \( gPb \), a contour detector that combines the local contour detection of \( mPb \) with the globalized detection of \( sPb \) (Arbeláez et al. 2011). In particular,

\[
gPb(x, y, \theta) = \sum_s \sum_i \beta_{i,s} G_{i,s}(x, y, \theta) + \gamma \cdot sPb(x, y, \theta),
\]

(2.30)

where \( \beta_{i,s} \) and \( \gamma \) are learned weights (Arbeláez et al. 2011).

### 2.6.4 Results

As seen in Figure 2.6, taken from the results in (Arbeláez et al. 2011), \( gPb \) is an effective contour detector that accurately maps contours in a less noisy manner than...
Fig. 2.6: From Top to Bottom: The original picture, $Pb$-detected contours, and $gPb$ detected contours. Note the reduced noise in the $gPb$ processed contour images (Arbeláez et al. 2011).

Fig. 2.7: Precision-Recall Curves for $Pb$, $mPb$, $sPb$, and $gPb$. Note that the $gPb$ curve is closest to that of human vision, and each refinement of the $Pb$ detector yields a better curve (Arbeláez et al. 2011).
Furthermore, the plot of ROC curves given in Figure 2.7 shows that \( gPb \) is the best-performing contour detector of the four detectors, outperforming \( Pb \), \( mPb \), and \( sPb \).

After the contours of an image are detected, Arbeláez et al. use an edge-detection approach to close any open contours and fully segment the image. We leave that discussion to (Arbeláez et al. 2011).

One of the major downsides of the \( sPb \) and \( gPb \) is that calculating the weight matrix \( W \) is computationally heavy, because for each pair of pixels \((i, j)\) one has to compute \(-\max_{p \in \mathcal{I}} \{mPb(p)\}\). However, notice that the weighting described in (Arbeláez et al. 2011) is just one such method for constructing \( W \), and constructing a cheaper adjacency matrix may still provide useful contour information. So, one could use the other adjacency matrices described above and possibly still get a useful result.
3

Color-Node Image Segmentation

3.1 Motivation

We are interested in improving the existing \textit{Ncut} method in order to account for objects that are far away but should belong to the same segment (patterns, for example). Oftentimes, these objects are linked by color/pixel values.

With that in mind, for both image segmentation and , we propose constructing a new graph $G_{\text{color}}$ which contains additional color nodes, where each pixel is connected to color nodes with edge weights determined by color similarity, and to its neighboring pixels. In Figure 3.1, a simplified diagram of graph $G_{\text{color}}$ of a $3 \times 3$ red-and-blue image is depicted. After constructing these graphs, we will run \textit{Ncut} experiments using this new method to see whether efficacy is improved.

![Fig. 3.1: Simplified $G_{\text{color}}$ of a $3 \times 3$ red-and-blue image.](image-url)
3.2 $G_{\text{color}}$

Given an image $I$ of size $r \times s$, with $n = rs$ pixels, we construct a graph $G_x$ where each node (representing a pixel), is ordered on the plane according to their relative positions in the image. Edges with weight $\lambda$ are drawn between each node and its immediate horizontal and vertical neighbors, if such a neighbor exists. Additionally, each node is connected to itself with weight 1. For example, the node representing the pixel at $(r, s) = (0, 0)$ is connected to the pixel nodes at $(0, 1)$ and $(1, 0)$, itself, and no others.

The resulting adjacency matrix of $G_x$, $W_x$, resembles the following matrix:

$$
\begin{bmatrix}
1 & \lambda & 0 & 0 & \cdots & 0 \\
\lambda & 1 & \lambda & 0 & \cdots & 0 \\
0 & \lambda & 1 & \lambda & \cdots & 0 \\
0 & 0 & \lambda & 1 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & \cdots & \lambda & 1
\end{bmatrix}
$$

A graphical representation of $W_x$ for a $5 \times 5$ image ($n = 25$) appears in Figure 3.2.

Fig. 3.2: $W_x$ for a $5 \times 5$ image. Black squares represent entries $(i, j)$ in $W_x$ where $W_x(i, j) = 0$, and white squares represent nonzero entries of $W_x$.

The adjacency matrix $W_x$ is sparse (that is, the number of non-zero entries is linear with the number of rows/columns of $W_x$. As discussed in Section 2.2, sparse matrices are desirable, as they require less memory and speed up matrix calculations (Gilbert et al. 1992).
While $G_x$ takes into account the relationships between neighboring pixels, it does not weight pixel colors at all, and thus, we must augment $G_x$ to account for color-based pixel relationships. To do so, $c$ additional nodes, representing the unique color values that appear in the image, are added to the graph, and edges are drawn between each pixel and each color node with a weight $w_{ij} = e^{-\frac{\|F(i) - F(j)\|}{2\sigma^2 C}}$, where $i$ denotes the $i^{th}$ pixel, $j$ denotes the $j^{th}$ color node, and $F(x)$ is the color value of the $x^{th}$ node.

In order to construct the adjacency matrix of $G_{color}$ with $n$ pixels and $c$ unique color values, we construct the $n \times c$ Color-Relation Matrix, $C$, where $C_{ij} = w_{ij}$ is the weight of the edge between the $i^{th}$ pixel and the $j^{th}$ color node.

Then, with $C$ defined, we can construct the Adjacency Matrix $W_{color}$ of $G_{color}$:

$$W_{color} = \begin{bmatrix} W_x & C \\ C^T & I_c \end{bmatrix}, \quad (3.1)$$

where $I_c$ is the $c \times c$ identity matrix.

It should be immediately clear that $W_{color}$ is symmetric, as $W_x$ and $I$ are symmetric matrices. Furthermore, assuming $c \ll n$, the matrix $C$ is sparse, and since $W_x$ and $I_c$ are sparse as well, we see that $W_{color}$ is sparse, as defined in Section 2.2, as well.

Then, with $G_{color}$ and $W_{color}$ defined, we use the Normalized Cuts algorithm as specified in Shi and Malik 2000, substituting $W_{color}$ for the $W$ used by Shi and Malik. We thus solve the generalized eigensystem:

$$\left(D_{color} - W_{color}\right)y = \lambda D_{color}y, \quad (3.2)$$

finding the eigenvector $y$ associated with the second smallest eigenvalue of the above system. Then, we perform K-Means clustering on the first $n$ entries of $y$. 


3.3 Methods

All algorithms below were implemented using Python, run on a 2021 MacBook Pro® with an 8-core Apple M1 Pro® and 16 GB of unified RAM.

For the first set of experiments (noisy concentric squares), images were constructed by generating a ground-truth segmentation using matrices¹, and then adding Gaussian noise to the ground-truth matrix. The resulting image matrix was then used as the basis for generating the adjacency and diagonal matrices of the image graph for \( N_{cuts} \). For Color-Nodes Segmentation, \( W_x \) was created by first generate the lattice graph \( G_x \), and then by multiplying the resulting adjacency matrix by \( \lambda \). \( C \), the color-relation matrix was then constructed³. Adjacency matrices for graphs of the second and third sets of experiments (textures and colors) were generated in a similar way.

In order to speed up calculations in the Color-Nodes Segmentation Method, all matrices were implemented as scipy sparse matrices. Due to the adjacency matrix for \( N_{cuts} \), \( W \), not being sparse, \( W \) was implemented as a normal numpy 2D array, as converting \( W \) to a scipy sparse matrix added computational time.

3.4 Results

3.4.1 Measuring Segmentation Accuracy

In order to measure how accurate a predicted segmentation is, we use the Jaccard Distance to gauge the dissimilarity between the ground-truth segmentation and the predicted segmentation. The Jaccard Distance of two sets \( A \) and \( B \), \( \neg J(A, B) \) is defined as \( 1 - J(A, B) \), where \( J(A, B) \) is the Jaccard Index of \( A \) and \( B \) (Jaccard 1912).

Then, let \( S \) represent the ground-truth segmentation, and \( A \) represent the segmented image. The Jaccard dissimilarity of the two images is given by:

\[\neg J(S, A) \]

¹In Python, using numpy 2D arrays
²Using the nx_grid_2d_graph() method to
³using the scipy.spatial.distance.cdist method
3.4.2 Experiments on Synthetic Greyscale Images

Figure 3.3 compares an initial $60 \times 60$ noisy image, its ground-truth segmentation, the respective segmentations obtained by $Ncuts$ and Color-Nodes Segmentation, as well as the eigenvector corresponding to the second smallest eigenvalue of generalized eigensystem (3.2). For the $Ncuts$ segmentation, $\sigma_I = 10$, and $\sigma_X = 60$. For the Color-Nodes Segmentation, $\sigma_C = 4.4$ and $\lambda = 20$.

In Figure 3.4, we compare the accuracy of $Ncuts$ and Color-Nodes Segmentation of five noisy test images for a variety of values of $\sigma_I$, $\sigma_X$, $\sigma_C$, and $\lambda$. In particular, starting from a ground-truth segmentation, we add various levels of Gaussian noise (centered around 0.001, 0.5, 1, 1.5, and 2, respectively) to obtain our test images.

We note that the Color-Nodes Segmentation method results in a more accurate segmentation for these particular images, particularly when the noise levels are high. When Noise = 2, for example, $Ncuts$ is highly inaccurate, and even the human eye can struggle to separate foreground from background. The Color-Nodes Segmentation approach, on the other hand, is still able to segment the noisy image with a very high degree of accuracy. Thus, Color-Nodes Segmentation is able to filter out image noise quite well, unlike $Ncuts$.

Additionally, Color-Nodes Segmentation runs five to seven times faster than the standard $Ncuts$ approach, likely owing to the sparse nature of $W_{color}$, as compared
Fig. 3.4: A comparison of \textit{Ncuts} and Color-Nodes Segmentation for various noisy images. From left to right: the test image, a heatmap of \textit{Ncuts} accuracy for varying values of $\sigma_X$ and $\sigma_I$, and a heatmap of Color-Nodes Segmentation accuracy for varying values of $\sigma_C$ and $\lambda$. 
to the dense $W$ used in $Ncuts$. Table 3.1 details the runtime of the above $80 \times 80$ pixel noisy concentric square experiments.

<table>
<thead>
<tr>
<th>Noise</th>
<th>$Ncuts$ (avg.)</th>
<th>Color-Node (avg.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.001</td>
<td>5.36 sec.</td>
<td>0.72 sec.</td>
</tr>
<tr>
<td>0.5</td>
<td>12.89 sec.</td>
<td>0.99 sec.</td>
</tr>
<tr>
<td>1</td>
<td>6.19 sec.</td>
<td>1.22 sec.</td>
</tr>
<tr>
<td>1.5</td>
<td>6.52 sec.</td>
<td>1.49 sec.</td>
</tr>
<tr>
<td>2</td>
<td>6.60 sec.</td>
<td>1.87 sec.</td>
</tr>
</tbody>
</table>

Tab. 3.1: Average runtime for one image segmentation using $Ncuts$ and Color-Nodes Segmentation on noisy synthetic $80 \times 80$ images. The rows of the table correspond to the test images in Figure 3.4.

### 3.4.3 Experiments on Textured Greyscale Images

In addition to performing well on segmenting noisy synthetic images, Color-Nodes Segmentation also performs well on segmenting greyscale textured images—that is, images that include patterns, such as a lattice or a brick wall. In particular, Color-Nodes Segmentation succeeds at segmenting images with multiple textures. The texture files used in this thesis were obtained from the Multiband Texture Database (He and Safia 2013).

Figure 3.5 compares an initial $80 \times 80$ dual-textured image, its ground-truth segmentation, the respective segmentations obtained by $Ncuts$ and Color-Nodes Segmentation, as well as the eigenvector corresponding to the second smallest eigenvalue of generalized eigensystem (3.2). For the $Ncuts$ segmentation, $\sigma_I = 9$ and $\sigma_X = 80$. For the Color-Nodes segmentation, $\sigma_C = 80$ and $\lambda = 80$. Note that though $Ncuts$ reproduces the initial test image, it isn’t able to properly segment the image, instead treating the lattice as the basis for its segmentation. By contrast, Color-Nodes segmentation is able to quite accurately segment the test image. In Figure 3.6, we compare the accuracy of $Ncuts$ and Color-Nodes Segmentation of five dual-textured test images for a variety of values of $\sigma_I$, $\sigma_X$, $\sigma_C$, and $\lambda$. Each image starts from a ground truth segmentation of a circle embedded in a square, and textures are applied to both regions of the image to generate the test image. As discussed above, the Color-Nodes Segmentation method largely results in a more accurate segmentation of textured images when compared to $Ncuts$. In particular, when it comes to pattern-textured images (the second and third test images in Figure 3.6, for
Fig. 3.5: (a) Initial $80 \times 80$ textured image. (b) Ground-truth segmentation. (c) $N_{cuts}$ segmentation. (d) Eigenvector found by Color-Nodes Segmentation. (e) Color-Nodes Segmentation example), Color-Nodes Segmentation is highly accurate. However, when it comes to irregular textures, such as the fourth and fifth test images in Figure 3.6, Color-Nodes Segmentation is less effective, though still more so than $N_{cuts}$.

Additionally, as seen in Table 3.2, Color-Nodes Segmentation is often faster than $N_{cuts}$, with one exception. Interestingly, it appears that pattern-textured images, such as the second and third test images in Figure 3.6, take significantly more time to segment, using both $N_{cuts}$ and Color-Nodes Segmentation, than other images.

<table>
<thead>
<tr>
<th>Test Image</th>
<th>$N_{cuts}$ (avg.)</th>
<th>Color-Node (avg.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16.66 sec.</td>
<td>0.90 sec.</td>
</tr>
<tr>
<td>2</td>
<td>4.23 sec.</td>
<td>10.04 sec.</td>
</tr>
<tr>
<td>3</td>
<td>55.36 sec.</td>
<td>5.27 sec.</td>
</tr>
<tr>
<td>4</td>
<td>4.42 sec.</td>
<td>0.97 sec.</td>
</tr>
<tr>
<td>5</td>
<td>5.44 sec.</td>
<td>0.99 sec.</td>
</tr>
</tbody>
</table>

Tab. 3.2: Average runtime for one image segmentation using $N_{cuts}$ and Color-Nodes Segmentation on textured $80 \times 80$ images. The rows of the table correspond to the test images in Figure 3.6.

3.4.4 Experiments on Real Images

On color images, Color-Nodes Segmentation performs quite well, outperforming $N_{cuts}$ on many images, both in terms of speed and segmentation accuracy. Figure 3.7 demonstrates the results of $N_{cuts}$ and Color-Nodes Segmentation run on five different RGB color images. Four of the images were obtained from the Visual Geometry Group (Gulshan et al. 2010), and the fifth from (David 2019). Each
Fig. 3.6: A comparison of *Ncuts* and Color-Nodes Segmentation for various textured images. From left to right: the ground truth segmentation, the test image, a heatmap of *Ncuts* accuracy for varying values of $\sigma_X$ and $\sigma_I$, and a heatmap of Color-Nodes Segmentation accuracy for varying values of $\sigma_C$ and $\lambda$. 
image was resized to a size of 80 × 80, in order to obtain size consistency—and segmentation time consistency—with the previous tests, and $K$—means clustering with 256 clusters was run on each image in order to standardize the number of colors present. Note that all of the color-image tests, Color-Nodes Segmentation produces a more accurate and less noisy segmentation than Ncuts. In particular, note that as seen in the previous section, Ncuts has difficulty segmenting textured images, while Color-Nodes Segmentation is more easily able to discern between foreground and background, even when colors are present. Furthermore, even when Ncuts generates a largely accurate segmentation, some extraneous pixels are added to the segment. For example, in the first test image of Figure 3.7, Ncuts includes a number of additional pixels, possibly background stars, in the foreground, along with the otherwise fairly accurately segmented Andromeda Galaxy. On the other hand, Color-Nodes Segmentation is able to separate the Andromeda Galaxy-region from the rest of the image, resulting in a better foreground-background segmentation.

Additionally, as with noisy and textured synthetic greyscale images, Color-Nodes Segmentation runs significantly faster than Ncuts when segmenting color images, as seen in Table 3.3.

<table>
<thead>
<tr>
<th>Test Color Image</th>
<th>Ncuts (avg.)</th>
<th>Color-Node (avg.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.29 sec.</td>
<td>1.16 sec.</td>
</tr>
<tr>
<td>2</td>
<td>4.35 sec.</td>
<td>1.15 sec.</td>
</tr>
<tr>
<td>3</td>
<td>4.24 sec.</td>
<td>0.97 sec.</td>
</tr>
<tr>
<td>4</td>
<td>4.25 sec.</td>
<td>1.09 sec.</td>
</tr>
<tr>
<td>5</td>
<td>4.28 sec.</td>
<td>1.10 sec.</td>
</tr>
</tbody>
</table>

Tab. 3.3: Average runtime for one image segmentation using Ncuts and Color-Nodes Segmentation on colored 80 × 80 images. The rows of the table correspond to the test images in Figure 3.7.

3.5 Conclusion and Future Directions

On synthetic noisy and textured images, Color-Nodes Segmentation performs significantly better, both in terms of accuracy of segmentation and segmentation time. Separating the color relations and distance relations of pixels allows for a sparse matrix, as well as the ability to connect distant, but similarly colored pixels, something
Fig. 3.7: Left to Right: Initial $80 \times 80$ image color image, $N_{cuts}$ segmentation, and $N_{cuts}$ with Color-Nodes Segmentation.
$Ncuts$ prevents from occurring, due to distant pixels having an edge weight close to 0.

Future work could include improving the running time of Color-Nodes Segmentation, perhaps by further sparsifying or otherwise changing the matrix $W_{\text{color}}$. Additionally, one of the notable features of segmentations arising from the Color-Nodes approach is a rounded segmented region. Segments often appear more rounded the ground-truth segment, and sharp corners are usually not seen in Color-Nodes Segmentations. This may be due to the fact that a pixel is connect to its immediate horizontal and vertical—but not diagonal—neighbors. Future work could focus on how to preserve corners and other "sharp" regions of segments while using the Color-Nodes approach. Finally, with our new method of building an image graph, we could apply the contour detection method described in (Arbeláez et al. 2011), noting that the eigenvector obtained from Color-Nodes Segmentation, which carries boundary information, is a lot less sensitive to noise as compared to the eigenvector obtained from $Ncuts$, and so may result in more accurate contour detection. Additionally, due to the sparsity of $W_{\text{color}}$, contour detection using $W_{\text{color}}$ will likely be faster than contour detection using the standard $Ncuts$ adjacency matrix.
References


