# Segmentation Using Superpixels: A Bipartite Graph Partitioning Approach

Zhenguo Li Xiao-Ming Wu Shih-Fu Chang
Dept. of Electrical Engineering, Columbia University, New York, NY
{zgli,xmwu,sfchang}@ee.columbia.edu

## Abstract

*Grouping cues can affect the performance of segmentation greatly. In this paper, we show that superpixels (image segments) can provide powerful grouping cues to guide segmentation, where superpixels can be collected easily by (over)-segmenting the image using any reasonable existing segmentation algorithms. Generated by different algorithms with varying parameters, superpixels can capture diverse and multi-scale visual patterns of a natural image. Successful integration of the cues from a large multitude of superpixels presents a promising yet not fully explored direction. In this paper, we propose a novel segmentation framework based on bipartite graph partitioning, which is able to aggregate multi-layer superpixels in a principled and very effective manner. Computationally, it is tailored to unbalanced bipartite graph structure and leads to a highly efficient, linear-time spectral algorithm. Our method achieves significantly better performance on the Berkeley Segmentation Database compared to state-of-the-art techniques.*

## 1. Introduction

Image segmentation is a fundamental low-level vision problem with a great potential in applications. While human can parse an image into coherent regions easily, it is found rather difficult for automatic vision systems. Despite a variety of segmentation techniques have been proposed, it remains challenging for any single method to do segmentation successfully due to the broad diversity and ambiguity of visual patterns in a natural image.

Methods developed under different motivations can behave quite differently. For example, Comaniciu and Meer's Mean Shift [4] seeks the modes of a non-parametric probability distribution in a feature space, and appears to well respect object details though it tends to split an object into pieces. Felzenszwalb and Huttenlocher's graph-based method (FH) [10] merges regions greedily, and tends to return gross segmentation. Shi and Malik's Normalized Cuts (Ncut) [26] aims to minimize the association between groups while maximizing the association within groups. It favors balanced partitions, at the risk of breaking object boundaries or large uniform regions (e.g. sky and grass) into chunks due to its "normalization" prior. See Section 4 for examples of the above observations.

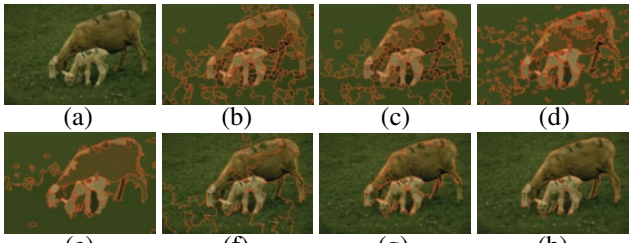



Figure 1. Segmentation using superpixels. (a) Input image. (b–e) Superpixels generated by over-segmenting the image using Mean Shift (b–c) and FH (d–e), each with different parameters. (f–g) Segmentations of Mean Shift and FH, respectively. (h) Segmentation of our proposed method in aggregating the superpixels.

Many studies have pointed out the synergistic effects of fusing complementary information, under ensemble learning [23], multi-kernel learning [16], etc. In this paper, we combine different segmentation methods for image segmentation (Fig. 1), which, to our knowledge, has not been sufficiently exploited. Our main contributions are summarized as follows.

1. We propose a novel segmentation framework which is able to aggregate multi-layer superpixels in a principled and effective manner. More generally, it can incorporate additional group constraints such as those specifying a set of pixels belong to the same group.

2. We show that spectral clustering [26, 22] can be highly efficient on unbalanced bipartite graphs, in contrast to the cases on general graphs. We achieve this by developing a novel graph partitioning technique that is tailored to the specific unbalanced graph structure.

3. We achieve highly competitive results on the Berkeley Segmentation Database with large performance gains ranging from 0.8146 to 0.8319 in PRI, and 12.21 to 11.29 in BDE, and very close to the best one in VoI and GCE (Section 4).

978-1-4673-1228-8/12/$31.00 ©2012 IEEE

The rest of the paper is organized as follows. Related work is reviewed in Section 2 and our model is presented in Section 3. Experimental results are reported in section 4 and the paper is concluded in Section 5.

## 2. Related Work

Over-segmentation occurs when image regions are segmented into smaller regions, each referred to as a "superpixel" [24]. Superpixels are usually expected to align with object boundaries, but this may not hold strictly in practice due to faint object boundaries and cluttered background. However, assuming the employed segmentation technique is reasonable, most pixels, if not all, within a superpixel can still belong to one coherent region (*superpixel cues*). In this sense, a superpixel imposes some "soft" constraints on "good" segmentations.

What roles can superpixels play in segmentation? First, superpixel cues enforce local smoothness since superpixels generally occupy consecutive image areas in which pixels are likely to be grouped together. Second, large elongated superpixels (e.g. the ones from FH in Fig. 1(e)) incorporate long-range grouping cues, which has shown to improve segmentation substantially [25, 2, 5]. Third, and more importantly, superpixels generated by different methods with varying parameters can capture diverse and multi-scale visual contents of a natural image. Enforcing such a large collection of superpixels simultaneously is expected to achieve synergistic effects. Other benefits include robust feature representation due to larger supports as well as the efficiency brought by the relatively small number of superpixels.

Superpixels have been exploited to aid segmentation in several different guises. In most cases, they are used to initialize segmentation [29, 28, 1, 8, 21]. However, an unsatisfactory over-segmentation often degrades performance substantially. In this paper, we tackle this by using multiple over-segmentations. Another approach works with multiple binary segmentations [20, 9, 14], relying on a strong assumption that each superpixel corresponds to one *whole* coherent region, which seems hard to achieve for real images. In [15], superpixels are incorporated in semi-supervised learning to derive a dense affinity matrix over pixels for spectral clustering, which can be computationally intensive. In contrast, our method is much more efficient by using unbalanced bipartite graphs. Also related is the work in [11] from machine learning, where bipartite graph partitioning is used for cluster ensemble. Our work differs in the way of constructing and partitioning the bipartite graph, and the new image segmentation application.

## 3. Superpixel Aggregation

In this section, we propose a novel graph-based segmentation framework which is able to efficiently integrate cues from multi-layer superpixels simultaneously. We rely on two simple observations, 1) pixels within one superpixels tend to belong one coherent region (superpixel cues); and 2) neighboring pixels which are close in feature space tend to belong to one coherent region (smoothness cues). We show that both cues can be effectively encoded using one bipartite graph. We further develop an efficient algorithm for unbalanced bipartite graph partitioning.

### 3.1. Bipartite Graph Construction

We construct a bipartite graph over pixels and superpixels collectively, as shown in Fig. 2. To enforce superpixel cues, we connect a pixel to a superpixel if the pixel is included in that superpixel. To enforce smoothness cues, we could simply connect neighboring pixels weighted by similarity, but this would end up with redundancy because the smoothness regarding neighboring pixels within superpixels were incorporated when enforcing superpixel cues. It may also incur complex graph partitioning due to denser connections on the graph. To compensate for the smoothness on the neighboring pixels across superpixels, we connect neighboring superpixels that are close in feature space.

Formally, denote $\mathcal{S}$ as a set of (multi-layer) superpixels over an image $I$, and let $G = \{X, Y, B\}$ be a bipartite graph with node set $X \cup Y$, where $X := I \cup \mathcal{S} = \{\mathbf{x}_i\}_{i=1}^{N_X}$ and $Y := \mathcal{S} = \{\mathbf{y}_j\}_{j=1}^{N_Y}$ with $N_X = |I| + |\mathcal{S}|$ and $N_Y = |\mathcal{S}|$, the numbers of nodes in $X$ and $Y$, respectively. The across-affinity matrix $B = (b_{ij})_{N_X \times N_Y}$ between $X$ and $Y$ is constructed as follows:

$$b_{ij} = \alpha, \text{ if } \mathbf{x}_i \in \mathbf{y}_j, \mathbf{x}_i \in I, \mathbf{y}_j \in \mathcal{S}; \quad (1)$$

$$b_{ij} = e^{-\beta d_{ij}}, \text{ if } \mathbf{x}_i \sim \mathbf{y}_j, \mathbf{x}_i \in \mathcal{S}, \mathbf{y}_j \in \mathcal{S}; \quad (2)$$

$$b_{ij} = 0, \text{ otherwise}, \quad (3)$$

where $d_{ij}$ denotes the distance[1] between the features of superpixels $\mathbf{x}_i$ and $\mathbf{y}_j$, $\sim$ denotes a certain neighborhood between superpixels[2], and $\alpha > 0$ and $\beta > 0$ are free parameters controlling the balance between the superpixel cues and the smoothness cues, respectively. By this construction, a pixel and the superpixels containing it are likely to be grouped together due to the connections between them. Two neighboring (defined by $\sim$) superpixels are more likely to be grouped together if they are closer in feature space. Particularly, superpixels are included in both parts of the graph to enforce the smoothness over superpixels.

Compared to the multi-layer and multi-scale graph models respectively presented in [15] and [5], our graph model

[1] For example, if color space is used as the feature space, and a superpixel $\mathbf{x}_i$ ($\mathbf{y}_j$) is represented by the average color $\mathbf{c}_i$ ($\mathbf{c}_j$) of the pixels within it, then $d_{ij} = \|\mathbf{c}_i - \mathbf{c}_j\|_2$. We use this setting in this paper, but note that our method applies to other features as well.

[2] For example, $\mathbf{x} \sim \mathbf{y}, \mathbf{x} \in \mathcal{S}, \mathbf{y} \in \mathcal{S}$, if $\mathbf{x} = \mathbf{y}$, or $\mathbf{y}$ is adjacent to $\mathbf{x}$ and is most similar to $\mathbf{x}$ in terms of (average) color. This neighborhood relationship is adopted in this paper.

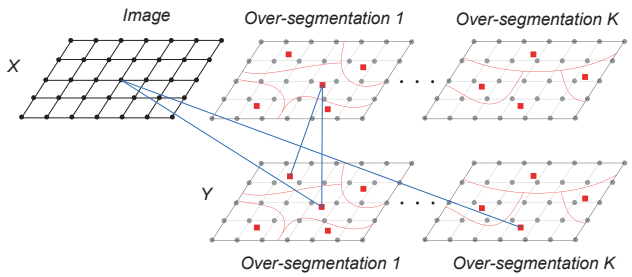



Figure 2. The proposed bipartite graph model with $K$ over-segmentations of an image. A black dot denotes a pixel while a red square denotes a superpixel.

has a distinct bipartite structure, which allows highly efficient graph partitioning as shown later. Besides, our graph model is much sparser because pixels are only connected to superpixels, while in [15] and [5], neighboring pixels are also connected to each other. Common to the three methods is that they all incorporate long-range grouping cues.

### 3.2. Bipartite Graph Partitioning

Given the above bipartite graph $G = \{X, Y, B\}$, the task is to partition it into $k$ groups, where $k$ is manually specified. Each group includes a subset of pixels and/or superpixels, and the pixels from one group form a segment. Various techniques can be employed for such a task. Here we use spectral clustering [26, 22] since it has been successfully applied to a variety of applications including image segmentation [26].

Spectral clustering captures essential cluster structure of a graph using the spectrum of graph Laplacian. Mathematically, it solves the generalized eigen-problem [26]:

$$L\mathbf{f} = \gamma D\mathbf{f}, \tag{4}$$

where $L$ is the graph Laplacian and $D$ is the degree matrix[3]. For a $k$-way partition, the bottom[4] $k$ eigenvectors are computed. Clustering is performed by applying $k$-means [22] or certain discretization technique [30].

To solve (4), one can simply treat the bipartite graph constructed above as a general sparse[5] graph and apply the Lanczos method [13] to $\bar{W} := D^{-1/2}WD^{-1/2}$, the normalized affinity matrix, which takes $O(k(N_X + N_Y)^{3/2})$ running time empirically [26].

Alternatively, it was shown, in the literature of document clustering, that the bottom $k$ eigenvectors of (4) can be derived from the top $k$ left and right singular vectors of the normalized across-affinity matrix $\bar{B} := D_X^{-1/2}BD_Y^{-1/2}$, where $D_X = \text{diag}(B\mathbf{1})$ and $D_Y = \text{diag}(B^\top\mathbf{1})$ are the degree matrices of $X$ and $Y$, respectively [7, 31]. This partial SVD, done typically by an iterative process alternating between matrix-vector multiplications $\bar{B}\mathbf{v}$ and $\bar{B}^\top\mathbf{u}$, is essentially equivalent to the Lanczos method on $\bar{W}$ [13]. It does not bring substantial benefit on solving (4) and is still subject to a complexity of $O(k(N_X + N_Y)^{3/2})$ (Fig.3).

### 3.3. Transfer Cuts

The above methods do not exploit the structure that the bipartite graph can be unbalanced, i.e., $N_X \neq N_Y$. In our case, $N_X = N_Y + |I|$, and $|I| \gg N_Y$ in general. Thus we have $N_X \gg N_Y$. This unbalance can be translated into the efficiency of partial SVDs without losing accuracy. One way is by using the dual property of SVD that a left singular vector can be derived from its right counterpart, and vice versa[6]. In this paper, we pursue a "sophisticated" path which not only exploits such unbalance but also sheds light on spectral clustering when operated on *bipartite* graphs.

We point out an essential equivalence between the eigen-problem (4) on the original bipartite graph and the one on a much smaller graph over superpixels *only*:

$$L_Y\mathbf{v} = \lambda D_Y\mathbf{v}, \tag{5}$$

where $L_Y = D_Y - W_Y$, $D_Y = \text{diag}(B^\top\mathbf{1})$, and $W_Y = B^\top D_X^{-1}B$. $L_Y$ is *exactly* the Laplacian of the graph $G_Y = \{Y, W_Y\}$ because $D_Y = \text{diag}(B^\top\mathbf{1}) = \text{diag}(W_Y\mathbf{1})$. It should be noted that our analysis in this section applies to spectral clustering on any bipartite graph.

Our main result states that the bottom $k$ eigenvectors of (4) can be obtained from the bottom $k$ eigenvectors of (5), which can be computed efficiently given the much smaller scale of (5). Formally, we have the following Theorem 1.

**Theorem 1.** *Let $\{(\lambda_i, \mathbf{v}_i)\}_{i=1}^k$ be the bottom $k$ eigenpairs of the eigen-problem (5) on the bipartite graph $G = \{X, Y, B\}$, $0 = \lambda_1 \leq \cdots \leq \lambda_k < 1$. Then $\{(\gamma_i, \mathbf{f}_i)\}_{i=1}^k$ are the bottom $k$ eigenpairs of the eigen-problem (4) on the graph $G_Y = \{Y, B^\top D_X^{-1}B\}$, where $0 \leq \gamma_i < 1$, $\gamma_i(2 - \gamma_i) = \lambda_i$, and $\mathbf{f}_i = (\mathbf{u}_i^\top, \mathbf{v}_i^\top)^\top$ with $\mathbf{u}_i = \frac{1}{1-\gamma_i}P\mathbf{v}_i$ and $P := D_X^{-1}B$ is the associated transition probability matrix from $X$ to $Y$.*

*Proof.* See the Appendix. □

It is interesting to note that, if $B$ is by construction a transition probability matrix from $Y$ to $X$ (i.e. non-negative

[3] $L := D - W$ and $D := \text{diag}(W\mathbf{1})$ with $W$ denoting the affinity matrix of the graph and $\mathbf{1}$ a vector of ones of appropriate size.

[4] Bottom (top) eigenvectors refer to the ones with smallest (largest) eigenvalues. Similar arguments apply to singular vectors.

[5] In our bipartite graph model, a pixel is connected to only $K$ superpixels for $K$ over-segmentations of an image. Given $K \ll (N_X + N_Y)$, the graph $\{V, W\}$ is actually highly sparse. In our experiments in Section 4, $K = 5$ or 6, and $N_X + N_Y > 481 \times 321 = 154401$.

[6] Formally, let $\mathbf{v}_i$ and $\mathbf{u}_i$ be the $i$-th right and left singular vector of $\bar{B}$, with singular value $\lambda_i$. Then $\bar{B}\mathbf{v}_i = \lambda_i\mathbf{u}_i$ The top $k$ right singular vectors $\mathbf{v}_1, \cdots, \mathbf{v}_k$ are exactly the top $k$ eigenvectors of $\bar{B}^\top\bar{B}$, which can be computed efficiently by the Lancsoz method or even a full eigenvalue decomposition given the much smaller size ($N_Y \times N_Y$) of $\bar{B}^\top\bar{B}$.

Table 1. Complexities of eigen-solvers for Ncut on bipartite graphs

| Algorithm | Complexity |
|---|---|
| Lanczos [26] | $O(k(N_X + N_Y)^{3/2})$ |
| SVD [31] | $O(k(N_X + N_Y)^{3/2})$ |
| Ours | $2k(1 + d_X)N_X$ operations $+ O(kN_Y^{3/2})$ |

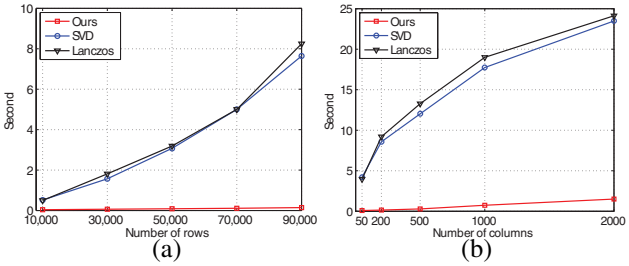
and the sum of the entries in each column is 1.), then the induced affinity matrix $W_Y$ over $Y$, $W_Y := B^\top D_X^{-1} B$, is exactly the one-step transition probability matrix on the graph $G_Y = \{Y, B^\top D_X^{-1} B\}$, or the two-step transition probability matrix on the original bipartite graph $G = \{X, Y, B\}$ following $Y \to X \to Y$ [17].

By Theorem 1, computing $\mathbf{f}_i$ from $\mathbf{v}_i$ needs $2N_X d_X + 2N_X$ operations, following the execution order $\frac{1}{1-\gamma_i}(D_X^{-1}(B\mathbf{v}_i))$, where $d_X$ is the average number[7] of edges connected to each node in $X$. So it takes $2k(1 + d_X)N_X$ operations for computing $\mathbf{f}_1, \ldots, \mathbf{f}_k$ from $\mathbf{v}_1, \ldots, \mathbf{v}_k$, plus a cost of $O(kN_Y^{3/2})$ for deriving $\mathbf{v}_1, \ldots, \mathbf{v}_k$ with the Lanczos method [26].

So far we have presented three methods to solve (4) for the bottom $k$ eigenvectors, whose complexities are listed in Table 1, where we can see that only our method comes with a linear time (w.r.t. $N_X$) with a small constant. To compare their performance in practice, we test on a series of bipartite graphs $G = \{X, Y, B\}$ of different sizes. The results are shown in Fig. 3. For Fig. 3(a), $N_Y$ is fixed to 200 while $N_X$ ranges from $10,000$ to $90,000$. For Fig. 3(b), $N_X$ is fixed to $100,000$ while $N_Y$ ranges from 50 to 2000. For each pair $\{N_X, N_Y\}$, $B$ is randomly generated in MATLAB with $B = rand(N_X, N_Y)$, and entries other than the 5 largest per row are set to zeros (i.e., each node in $X$ is connected to 5 nodes in $Y$). For each graph, 10 eigenvectors of (4) are computed. The results reported are averaged over 10 graphs. For the Lanczos method and SVD, we use $eigs$ and $svds$ in MATLAB; for our method we also use the Lanczos method (i.e. $eigs$) to solve (5). For all the three methods, the tolerance for convergence is set to $1e-10$. From Fig. 3, we can see that our cost is much less than those of SVD and the Lanczos method in both cases. The costs of SVD and the Lanczos are quite close to each other. The small slope of our method in Fig. 3(a) confirms that the constant factor of our linear complexity is quite small.

We summarize our approach to bipartite graph partitioning in Algorithm 1, which we call Transfer Cuts (Tcut) since it transfers the eigen-operation from the original graph to a smaller one. In step 5, one may employ the discretization technique in [30] which is tailored to Ncut, or apply $k$-means to the rows of the matrix $F := (\mathbf{f}_1, \ldots, \mathbf{f}_k)$ after each row is being normalized to unit length, which is justified by stability analysis [22]. In our experiments, we found

[7] In our bipartite graph, $d_X$ is approximately equal to the number of over-segmentations or layers.



(a) (b)

Figure 3. Cost vs. Solver to (4).

the performances were comparable in both cases. So we only report results on the latter case since it is more efficient.

**Algorithm 1** Transfer Cuts

**Input**: A bipartite graph $G = \{X, Y, B\}$ and a number $k$.
**Output**: A $k$-way partition of $G$.

1: Form $D_X = \text{diag}(B\mathbf{1})$, $D_Y = \text{diag}(B^\top \mathbf{1})$, $W_Y = B^\top D_X^{-1} B$, and $L_Y = D_Y - W_Y$.
2: Compute the bottom $k$ eigenpairs $\{(\lambda_i, \mathbf{v}_i)\}_{i=1}^k$ of $L_Y \mathbf{v} = \lambda D_Y \mathbf{v}$.
3: Obtain $\gamma_i$ such that $0 \le \gamma_i < 1$ and $\gamma_i(2 - \gamma_i) = \lambda_i$, $i = 1, \ldots, k$.
4: Compute $\mathbf{f}_i = (\mathbf{u}_i^\top, \mathbf{v}_i^\top)^\top$, with $\mathbf{u}_i = \frac{1}{1-\gamma_i} D_X^{-1} B \mathbf{v}_i$, $i = 1, \ldots, k$.
5: Derive $k$ groups of $X \cup Y$ from $\mathbf{f}_1, \ldots, \mathbf{f}_k$.

### 3.4. Proposed Image Segmentation Algorithm

Our segmentation procedures are listed in Algorithm 2, which we call Segmentation by Aggregating Superpixels (SAS). The main cost of SAS is in collecting superpixels (step 1) and bipartite graph partitioning (step 3). The cost of step 3 is linear in the number of pixels in the image with a small constant (Section 3.3), and is negligible compared to that of step 1 (Section 4). Potentially, there may be a group consisting of superpixels only. In such cases, the returned number of segments will decrease by 1. However, we never encounter such cases in our experiments probably because of the relatively strong connections between pixels and the superpixels containing them.

**Algorithm 2** Segmentation by Aggregating Superpixels

**Input**: An image $I$ and the number of segments $k$.
**Output**: A $k$-way segmentation of $I$.

1: Collect a bag of superpixels $\mathcal{S}$ for $I$.
2: Construct a bipartite graph $G = \{X, Y, B\}$ with $X = I \cup \mathcal{S}$, $Y = \mathcal{S}$, and $B$ defined in (1-3).
3: Apply Tcut in Algorithm 1 to derive $k$ groups of $G$.
4: Treat pixels from the same group as a segment.

## 4. Experimental Results

In this section, we evaluate the proposed image segmentation algorithm SAS on a benchmark image database, and compare it with state-of-the-art methods.

SAS requires a bag of superpixels, which are generated by Mean Shift [4] and FH [10], though other choices are also possible. The main reason of choosing them is that they are complementary and practically efficient. For Mean Shift, we generate three layers of superpixels with parameters $(h_s, h_r, M) \in \{(7,7,100),(7,9,100),(7,11,100)\}$ (denoted as MS), where $h_s$ and $h_r$ are bandwidth parameters in the spatial and range domains, and $M$ is the minimum size of each segment. For FH, we generate two or three layers of superpixels with parameters $(\sigma, c, M) \in \{(0.5,100,50),(0.8,200,100)\}$ (denoted as FH1) or $(\sigma, c, M) \in \{(0.8,150,50),(0.8,200,100),(0.8,300,100)\}$ (denoted as FH2), respectively, where $\sigma$ and $c$ are the smoothing and scale parameters, and $M$ is the minimum size of each segment. FH1 is selected if the variance of selected automatically according to the image variance in the LAB color space (feature space) using a threshold. We also report results on different combinations of superpixels.

For the graph, each pixel is connected to the superpixels containing it and each superpixel is connected to itself and its nearest neighbor in the feature space among its spatially adjacent superpixels (see Fig. 2). Each superpixel is represented by the average LAB color of the pixels within it. The inclusion and smoothness parameters $\alpha$ and $\beta$ are set empirically to $\alpha = 10^{-3}$ and $\beta = 20$. We also test the sensitivity of SAS w.r.t. the variation of these parameters. Like other graph partitioning methods, the number of segments is manually set for each image (e.g. [15]). All the results of SAS reported in this section use the same parameters as above, including $\alpha$, $\beta$, and the parameters for Mean Shift and FH in generating superpixels, unless otherwise stated.

### 4.1. Berkeley Database

We report results on the Berkeley Segmentation Database [18], which consists of 300 natural images of diverse scene categories. Each image is manually segmented by a number of different human subjects, and on average, five ground truths are available per image.

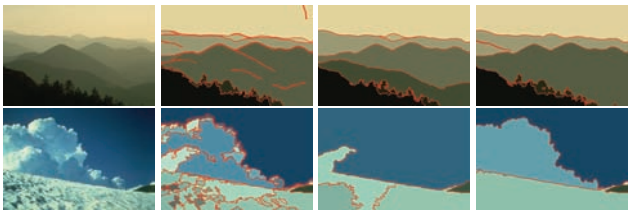
To quantify the segmentation results, we follow common practice (e.g. [15, 9, 28]) to use the four criteria: 1) Probabilistic Rand Index (PRI) [27], measuring the likelihood of a pair of pixels being grouped consistently in two segmentations; 2) Variation of Information (VoI) [19], computing the amount of information of one result not contained in the other; 3) Global Consistency Error (GCE) [18], measuring the extent to which one segmentation is a refinement of the other; and 4) Boundary Displacement Error (BDE) [12], computing the average displacement between the boundaries of two segmentations. A segmentation is better if PRI is larger and the other three are smaller, when compared to the ground truths.

Table 2. Performance evaluation of the proposed method (SAS) against other methods over the Berkeley Segmentation Database

| Methods | PRI | VoI | GCE | BDE |
|---|---|---|---|---|
| Ncut | 0.7242 | 2.9061 | 0.2232 | 17.15 |
| Mean Shift | 0.7958 | 1.9725 | 0.1888 | 14.41 |
| FH | 0.7139 | 3.3949 | **0.1746** | 16.67 |
| JSEG | 0.7756 | 2.3217 | 0.1989 | 14.40 |
| MNcut | 0.7559 | 2.4701 | 0.1925 | 15.10 |
| NTP | 0.7521 | 2.4954 | 0.2373 | 16.30 |
| SDTV | 0.7758 | 1.8165 | **0.1768** | 16.24 |
| TBES | 0.80 | 1.76 | N/A | N/A |
| UCM | 0.81 | **1.68** | N/A | N/A |
| MLSS | 0.8146 | 1.8545 | 0.1809 | **12.21** |
| SAS | **0.8319** | **1.6849** | **0.1779** | **11.29** |
| SAS(MS) | 0.7991 | 1.9320 | 0.2222 | 15.37 |
| SAS(FH1) | 0.8070 | 1.8690 | 0.2167 | 14.28 |
| SAS(FH2) | 0.8007 | 1.7998 | 0.2105 | 17.17 |
| SAS(MS+FH1) | **0.8266** | 1.7396 | 0.1868 | **11.83** |
| SAS(MS+FH2) | **0.8246** | **1.7144** | 0.1904 | 12.63 |

We compare the average scores of SAS and the ten benchmark algorithms, Ncut [26], Mean Shift [4], FH [10], JSEG [6], Multi-scale Ncut (MNcut) [5], Normalized Tree Partitioning (NTP) [28], Saliency Driven Total Variation (SDTV) [9], Texture and Boundary Encoding-based Segmentation (TBES) [21], Ultrametric Contour Maps (UCM) [1], and Multi-Layer Spectral Segmentation (MLSS) [15]. The scores of these algorithms are collected from [15, 1, 9, 28]. To see how SAS affected by the superpixels used, we also report the scores of SAS with different combinations of superpixels, where SAS(MS) represents SAS using the superpixels generated by Mean Shift alone, with parameters MS; SAS(MS+FH1) denotes SAS with the superpixels generated collectively by Mean Shift and FH, with parameters MS and FH1, respectively; and similar arguments hold for SAS(FH1), SAS(FH2), and SAS(MS+FH2). All our methods use the same parameters, including the number of segments for each image.

The scores are shown in Table 2, with the three best results highlighted in bold for each criterion. We see that SAS ranks first in PRI and BDE by a large margin compared to previous methods, and second in VoI and third in GCE with performance quite close to the best one. The scores of SAS with superpixels from a single algorithm (SAS(MS), SAS(FH1), SAS(FH2)) are not comparable to those from both, suggesting that complementary superpixels do improve SAS significantly (see Fig. 6). Although SAS with a fixed set of complementary superpixels (SAS(MS+FH1), SAS(MS+FH2)) is already highly competitive, a simple adaptive selection of superpixels based on the image variance can make it even better (SAS) (see Fig. 4). Consider-



(a) (b) (c) (d)

Figure 4. Benefit of adaptive superpixels. (a) Input images. (b) The 3rd layer superpixels of FH2 (i.e., with parameters $(0.8, 300, 100)$). (c) SAS(MS+FH1). (d) SAS(MS+FH2). Top row: SAS(MS+FH1) gives better result than SAS(MS+FH2) by not including the improper superpixels in (b) (since different mountains are joined by one big superpixel). Bottom row: SAS(MS+FH2) performs better than SAS(MS+FH1) by including the proper superpixels in (b) (because different parts of a large cloud area is connected by one big superpixel).

ing all four criteria, SAS appears to work best overall.

Some segmentation examples can be visualized in Fig. 5. The results of other algorithms are manually tuned using the authors' softwares, except for those of TBES and MLSS. For TBES, we use the results available on the authors' website. For MLSS, we use the parameters suggested by the authors. It can be seen from Fig. 5 that the segmentations of SAS are perceptually more satisfactory. While Mean Shift tends to preserve more object details, FH detects gross regions. TBES favors homogeneous regions and simple boundaries due to the minimization of "coding length". The Ncut based methods (Ncut, MNcut, MLSS) tend to break large uniform regions because of the "normalization" prior. Though also using the Ncut objective, SAS is much less affected by this prior by "relying" on superpixels alone. Indeed, a large uniform region can consist of a relatively small number of superpixels, thanks to the non-uniform superpixels produced by Mean Shift and FH (see Fig. 1). SAS and MLSS give better results than MNcut by including the superpixel cues. Compared with MLSS, SAS produces more robust results by using complementary superpixels and not connecting neighboring pixels.

Fig. 6 illustrates the performance of SAS w.r.t. different combinations of superpixels. We can see that though increasing layers of superpixels from a single algorithm generally improves SAS, combining superpixels from different algorithms improves it even more significantly. We also show results from MLSS (Fig. 6(k)) and TBES (Fig. 6(l)) for comparisons with ours (Fig. 6(j)).

Table 3 shows the results of SAS with varying prior parameters $\alpha$ and $\beta$. We can see that SAS is quite stable w.r.t. the two parameters. This is because appropriate superpixel cues can have larger impact on SAS than the relative weights used to encode them.

We also evaluate SAS using the boundary based F-measure (FM) and the region based segmentation covering (SC) suggested in [1]. FM compares boundaries between a

Table 3. Sensitivity of SAS w.r.t. the variation of parameters on Berkeley Segmentation Database

| $\alpha$ | $\{10^{-9}, 10^{-5}, 10^{-1}, 10^{3}\}$ | | | | $10^{-3}$ | | | |
|---|---|---|---|---|---|---|---|---|
| $\beta$ | 20 | | | | $2 \times \{10^{-5}, 10^{-1}, 10^{3}, 10^{7}\}$ | | | |
| PRI | 0.806 | 0.813 | 0.818 | 0.818 | 0.821 | 0.821 | 0.814 | 0.815 |
| VoI | 1.867 | 1.810 | 1.836 | 1.840 | 1.811 | 1.811 | 1.836 | 1.831 |
| GCE | 0.209 | 0.203 | 0.201 | 0.202 | 0.194 | 0.194 | 0.210 | 0.209 |
| BDE | 13.76 | 13.33 | 13.27 | 13.31 | 12.40 | 12.35 | 13.70 | 13.70 |

Table 4. Boundary and region benchmarks on Berkeley Segmentation Database

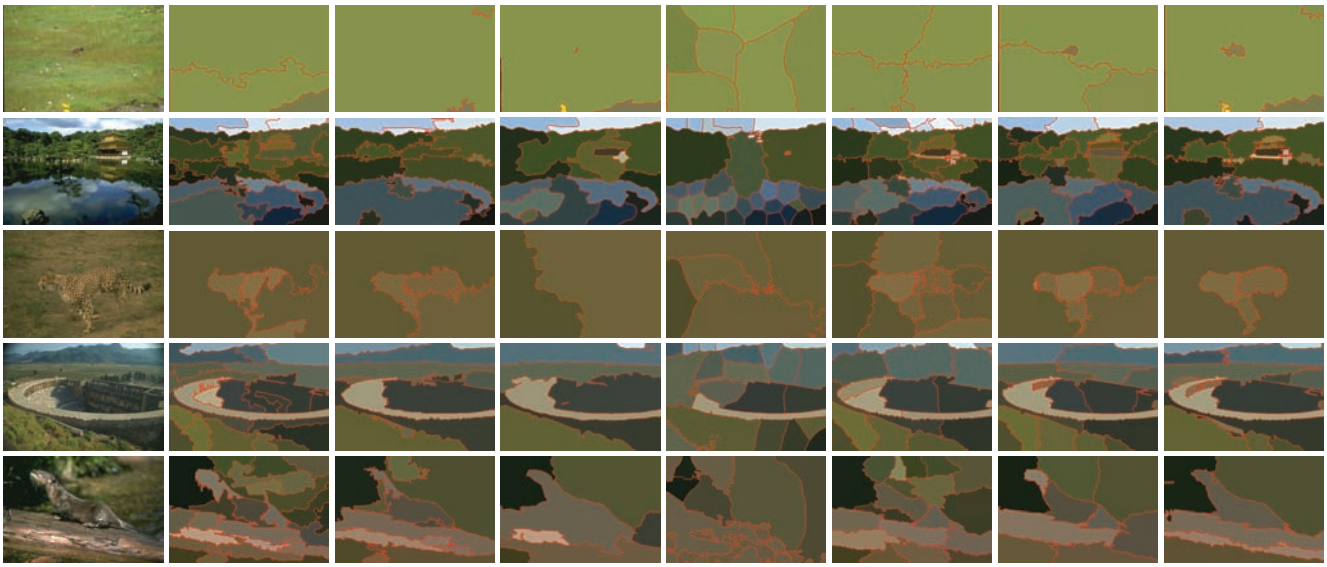
| Methods | FH | Ncut | Mean Shift | UCM | SAS |
|---|---|---|---|---|---|
| FM | 0.58 | 0.62 | 0.63 | **0.71** | **0.64** |
| SC | 0.51 | 0.44 | 0.54 | **0.58** | **0.62** |

machine segmentation and the ground truths while SC measures the region-wise covering of the ground truths by a machine segmentation. The parameters of SAS are fixed as above, except for the number of segments which is tuned for each image w.r.t. each metric. The scores are shown in Table 4, with the best two highlighted w.r.t. each metric. The scores of other methods are obtained from [1], with optimal parameters w.r.t. whole database. We can see that though with competitive results, SAS is not comparable to UCM in FM. This is because FM favors contour detectors like UCM over segmentation methods, as noted in [1]. In contrast, SAS outperforms UCM in region based SC.

On average, SAS takes 6.44 seconds to segment an image of size $481 \times 321$, where 4.11 seconds are for generating superpixels, and only 0.65 seconds for the bipartite graph partitioning with Tcut. In contrast, MNcut, MLSS, Ncut, and TBES usually take more than 30, 40, 150, and 500 seconds, respectively. All experiments run in MATLAB 7.11.0 (R2010b) on a laptop with 2.70 GHz CPU and 8GB RAM.

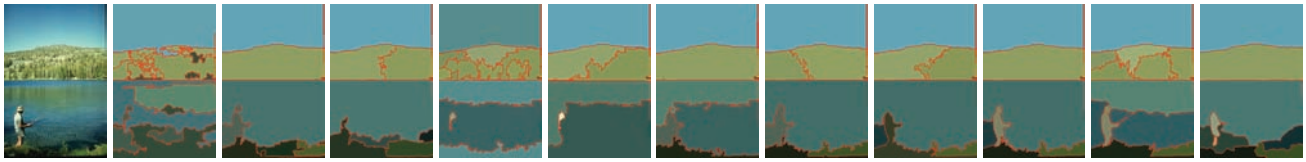
## 5. Conclusions

We have presented a novel method for image segmentation using superpixels. On one hand, the use of superpixels as grouping cues allows us to effectively encode complex image structures for segmentation. This is done by employing different segmentation algorithms and varying their parameters in generating superpixels. On the other hand, we fuse diverse superpixel cues in an effective manner using a principled bipartite graph partitioning framework. Compared with other Ncut based methods, our segmentation method is much less affected by the "normalization" prior of Ncut which tends to break large uniform regions in an image. Our another contribution is the development of a highly efficient spectral algorithm for bipartite graph partitioning, which is tailored to bipartite graph structures and provably much faster than conventional approaches, while providing novel insights on spectral clustering over bipartite graphs. Extensive experimental results on the Berkeley Segmentation Database have demonstrated the superior performance of the proposed image segmentation method,



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 5. Segmentation examples on Berkeley Segmentation Database. (a) Input images. (b) Mean Shift. (c) FH. (d) TBES. (e) Ncut. (f) MNcut. (g) MLSS. (h) SAS.



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l)

Figure 6. Segmentation with different combinations of superpixels. (a) Input image. (b) SAS(1, MS). (c) SAS(2, MS). (d) SAS(3, MS). (e) SAS(1, FH2). (f) SAS(2, FH2). (g) SAS(3, FH2). (h) SAS(1, MS & FH2). (i) SAS(2, MS & FH2). (j) SAS(3, MS & FH2). (k) MLSS. (l) TBES. Notations: SAS($i$, MS) denotes SAS using the first $i$ layers superpixels of Mean Shift; SAS($i$, MS & FH2) denotes SAS using both the first $i$ layers superpixels of Mean Shift and FH2.

in terms of both quantitative and perceptual criteria. Future work should study the selection of superpixels more systematically and the incorporation of high-level cues.

**Acknowledgments.** This work is supported in part by Office of Naval Research (ONR) grant #N00014-10-1-0242.

**Appendix.** To prove Theorem 1, we need several results. The next lemma describes the distribution of the eigenvalues of (4).

**Lemma 1.** *The eigenvalues of* (4) *are in* $[0, 2]$. *Particularly, if* $\gamma$ *is its eigenvalue, so is* $2 - \gamma$.

*Proof.* From (4), we have $D^{-1/2}LD^{-1/2}(D^{1/2}\mathbf{f}) = \gamma(D^{1/2}\mathbf{f})$, meaning that $\gamma$ is an eigenvalue of (4) iff it is an eigenvalue of the normalized graph Laplacian $\bar{L} := D^{-1/2}LD^{-1/2}$, whose eigenvalues are in $[0, 2]$ [3].

Direct calculation shows that if $(\gamma, \mathbf{f})$ is an eigenpair of (4), so is $(2 - \gamma, (\mathbf{f}|_X^\top, -\mathbf{f}|_Y^\top)^\top)$. □

By Lemma 1, the eigenvalues of (4) are distributed symmetrically w.r.t. 1, in $[0, 2]$, implying that half of the eigenvalues are not larger than 1. Therefore, it is reasonable to assume that the $k$ smallest eigenvalues are less than 1, as $k \ll (N_X + N_Y)/2$ in general. The following lemma will be used later to establish correspondence between the eigenvalues of (4) and (5).

**Lemma 2.** *Given* $0 \le \lambda < 1$, *the value of* $\gamma$ *satisfying* $0 \le \gamma < 1$ *and* $\gamma(2 - \gamma) = \lambda$ *exists and is unique. Such* $\gamma$ *increases along with* $\lambda$.

*Proof.* Define $h(\gamma) := \gamma(2 - \gamma)$, $\gamma \in [0, 1]$. Then $h(\cdot)$ is continuous, and $h(0) = 0$, $h(1) = 1$. Since $0 \le \lambda < 1$, there must exist $0 \le \gamma < 1$ such that $\gamma(2 - \gamma) = \lambda$. The uniqueness comes from the strictly increasing monotonicity of $h(\cdot)$ in $[0, 1]$. For the same reason, $\gamma$ satisfying $\gamma(2 - \gamma) = \lambda$ increases when $\lambda$ increases. □

The following theorem states that an eigenvector of (4), if confined to $Y$, will be an eigenvector of (5).

**Theorem 2.** *If* $L\mathbf{f} = \gamma D\mathbf{f}$, *then* $L_Y\mathbf{v} = \lambda D_Y\mathbf{v}$, *where* $\lambda := \gamma(2 - \gamma)$, $\mathbf{v} := \mathbf{f}|_Y$.

*Proof.* Denote $\mathbf{u} := \mathbf{f}|_X$. From $L\mathbf{f} = \gamma D\mathbf{f}$, we have (a) $D_X\mathbf{u} - B\mathbf{v} = \gamma D_X\mathbf{u}$ and (b) $D_Y\mathbf{v} - B^\top\mathbf{u} = \gamma D_Y\mathbf{v}$. From (a), we have $\mathbf{u} = \frac{1}{1-\gamma}D_X^{-1}B\mathbf{v}$. Substituting it into (b) yields $(D_Y - B^\top D_X^{-1}B)\mathbf{v} = \gamma(2 - \gamma)D_Y\mathbf{v}$, namely, $L_Y\mathbf{v} = \lambda D_Y\mathbf{v}$. □

The converse of Theorem 2 is also true, namely, an eigenvector of (5) can be extended to be one of (4), as detailed in the following theorem.

**Theorem 3.** *Suppose* $L_Y\mathbf{v} = \lambda D_Y\mathbf{v}$ *with* $0 \le \lambda < 1$. *Then* $L\mathbf{f} = \gamma D\mathbf{f}$, *where* $\gamma$ *satisfies* $0 \le \gamma < 1$ *and* $\gamma(2-\gamma) = \lambda$, *and* $\mathbf{f} = (\mathbf{u}^\top, \mathbf{v}^\top)^\top$ *with* $\mathbf{u} = \frac{1}{1-\gamma} D_X^{-1} B\mathbf{v}$.

*Proof.* First, by Lemma 2, $\gamma$ exists and is unique. To prove $L\mathbf{f} = \gamma D\mathbf{f}$, it suffices to show (a) $D_X\mathbf{u} - B\mathbf{v} = \gamma D_X\mathbf{u}$ and (b) $D_Y\mathbf{v} - B^\top\mathbf{u} = \gamma D_Y\mathbf{v}$. (a) holds because $\mathbf{u} = \frac{1}{1-\gamma} D_X^{-1} B\mathbf{v}$ is known. Since $L_Y\mathbf{v} = \lambda D_Y\mathbf{v}$, $\lambda = \gamma(2-\gamma)$, $L_Y = D_Y - B^\top D_X^{-1} B$, $\mathbf{u} = \frac{1}{1-\gamma} D_X^{-1} B\mathbf{v}$, and $\gamma \neq 1$, we have $\gamma(2-\gamma)D_Y\mathbf{v} = L_Y\mathbf{v} = (D_Y - B^\top D_X^{-1} B)\mathbf{v} = D_Y\mathbf{v} - (1-\gamma)B^\top\mathbf{u} \iff (1-\gamma)^2 D_Y\mathbf{v} = (1-\gamma)B^\top\mathbf{u} \iff (1-\gamma)D_Y\mathbf{v} = B^\top\mathbf{u}$, which is equivalent to (b). □

*Proof of Theorem 1.* By Theorem 3, $\mathbf{f}_i$ is the eigenvector of (4) corresponding to the eigenvalue $\gamma_i$, $i = 1, \ldots, k$. By Lemma 2, $\gamma_1 \le \cdots \le \gamma_k < 1$. So our proof is done if $\gamma_i$'s are the $k$ smallest eigenvalues of (4). Otherwise assume $(\gamma_1, \ldots, \gamma_k) \neq (\gamma'_1, \ldots, \gamma'_k)$, where $\gamma'_1, \ldots, \gamma'_k$ denote the $k$ smallest eigenvalues of (4) with non-decreasing order whose corresponding eigenvectors are $\mathbf{f}'_1, \ldots, \mathbf{f}'_k$, respectively. Let $\gamma'_j$ be the first in $\gamma'_i$'s that $\gamma'_j < \gamma_j < 1$, i.e., $\gamma'_i = \gamma_i$, $\forall i < j$. Without loss of generality, assume $\mathbf{f}'_i = \mathbf{f}_i$, $\forall i < j$. Denote $\lambda^* = \gamma'_j(2-\gamma'_j)$. Then by Lemma 2, $\lambda^* < \lambda_j$ because $\gamma'_j < \gamma_j$. By Theorem 2, $\lambda^*$ is an eigenvalue of (5) with eigenvector $\mathbf{v}^* := \mathbf{f}'_j|_Y$. Note that $\mathbf{v}^*$ is different from $\mathbf{v}_i$, $\forall i < j$, showing that $(\lambda^*, \mathbf{v}^*)$ is the $j$-th eigenpair of (5), which contradicts the fact that $(\lambda_j, \mathbf{v}_j)$ is the $j$-th eigenpair of (5). □

## References


[1] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. In *CVPR*, 2009. 2, 5, 6

[2] A. Barbu and S. Zhu. Graph partition by swendsen-wang cuts. In *ICCV*, pages 320–327, 2003. 2

[3] F. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997. 7

[4] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. PAMI*, pages 603–619, 2002. 1, 5

[5] T. Cour, F. Benezit, and J. Shi. Spectral segmentation with multiscale graph decomposition. In *CVPR*, 2005. 2, 3, 5

[6] Y. Deng and B. Manjunath. Unsupervised segmentation of color-texture regions in images and video. *IEEE Trans. PAMI*, 23(8):800–810, 2001. 5

[7] I. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD*, 2001. 3

[8] L. Ding and A. Yilmaz. Interactive image segmentation using probabilistic hypergraphs. *Pattern Recognition*, 43(5):1863–1873, 2010. 2

[9] M. Donoser, M. Urschler, M. Hirzer, and H. Bischof. Saliency driven total variation segmentation. In *ICCV*, 2009. 2, 5

[10] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004. 1, 5

[11] X. Fern and C. Brodley. Solving cluster ensemble problems by bipartite graph partitioning. In *ICML*, 2004. 2

[12] J. Freixenet, X. Muñoz, D. Raba, J. Martí, and X. Cufí. Yet another survey on image segmentation: Region and boundary information integration. *ECCV*, pages 21–25, 2002. 5

[13] G. Golub and C. Van Loan. *Matrix computations*. Johns Hopkins University Press, 1996. 3

[14] A. Ion, J. Carreira, and C. Sminchisescu. Image segmentation by figure-ground composition into maximal cliques. In *ICCV*, 2011. 2

[15] T. Kim and K. Lee. Learning full pairwise affinities for spectral segmentation. In *CVPR*, 2010. 2, 3, 5

[16] G. Lanckriet, N. Cristianini, P. Bartlett, L. Ghaoui, and M. Jordan. Learning the kernel matrix with semidefinite programming. *JMLR*, 5:27–72, 2004. 1

[17] W. Liu, J. He, and S. Chang. Large graph construction for scalable semi-supervised learning. In *ICML*, 2010. 4

[18] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, pages 416–423, 2001. 5

[19] M. Meila. Comparing clusterings: an axiomatic view. In *ICML*, pages 577–584. ACM, 2005. 5

[20] B. Mičušík and A. Hanbury. Automatic image segmentation by positioning a seed. In *ECCV*, pages 468–480, 2006. 2

[21] H. Mobahi, S. Rao, A. Yang, S. Sastry, and Y. Ma. Segmentation of natural images by texture and boundary compression. *IJCV*, pages 1–13, 2011. 2, 5

[22] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2001. 1, 3, 4

[23] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11(1):169–198, 1999. 1

[24] X. Ren and J. Malik. Learning a classification model for segmentation. In *ICCV*, pages 10–17, 2003. 2

[25] E. Sharon, A. Brandt, and R. Basri. Fast multiscale image segmentation. In *CVPR*, pages 70–77, 2000. 2

[26] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. PAMI*, 22(8):888–905, 2000. 1, 3, 4, 5

[27] R. Unnikrishnan, C. Pantofaru, and M. Hebert. Toward objective evaluation of image segmentation algorithms. *IEEE Trans. PAMI*, 29(6):929–944, 2007. 5

[28] J. Wang, Y. Jia, X. Hua, C. Zhang, and L. Quan. Normalized tree partitioning for image segmentation. In *CVPR*, 2008. 2, 5

[29] A. Yang, J. Wright, Y. Ma, and S. Sastry. Unsupervised segmentation of natural images via lossy data compression. *CVIU*, 110(2):212–225, 2008. 2

[30] S. X. Yu and J. Shi. Multiclass spectral clustering. In *ICCV*, pages 313–319, 2003. 3, 4

[31] H. Zha, X. He, C. Ding, H. Simon, and M. Gu. Bipartite graph partitioning and data clustering. In *CIKM*, 2001. 3, 4