Entropy Rate Superpixel Segmentation

Ming-Yu Liu[†] Oncel Tuzel[‡] Srikumar Ramalingam[‡] Rama Chellappa[†]
[†]University of Maryland College Park [‡]Mitsubishi Electric Research Labs
{mingyliu,rama}@umiacs.umd.edu {oncel,ramalingam}@merl.com

Abstract

We propose a new objective function for superpixel segmentation. This objective function consists of two components: entropy rate of a random walk on a graph and a balancing term. The entropy rate favors formation of compact and homogeneous clusters, while the balancing function encourages clusters with similar sizes. We present a novel graph construction for images and show that this construction induces a matroid— a combinatorial structure that generalizes the concept of linear independence in vector spaces. The segmentation is then given by the graph topology that maximizes the objective function under the matroid constraint. By exploiting submodularity and monotonicity of the objective function, we develop an efficient greedy algorithm. Furthermore, we prove an approximation bound of $\frac{1}{2}$ for the optimality of the solution. Extensive experiments on the Berkeley segmentation benchmark show that the proposed algorithm outperforms the state of the art in all the standard evaluation metrics.

1. Introduction

Superpixel segmentation is an important module for many computer vision applications such as object recognition [14], image segmentation [19, 8], and single view 3D reconstruction [7, 18]. A superpixel is commonly defined as a perceptually uniform region in the image.

The major advantage of using superpixels is computational efficiency. A superpixel representation greatly reduces the number of image primitives compared to the pixel representation. For instance, in an L-label labeling problem, the solution space for a pixel representation is L^n where n is the number of pixels— typically 10^6; in contrast, the solution space for a superpixel representation is L^m where m is the number of superpixels— typically 10^2 ($<< 10^6$). Moreover, superpixel segmentation provides the spatial support for computing region based features.

The desired properties of superpixel segmentation depends on the application of interest. Here we list some general properties required by various vision applications:

- Every superpixel should overlap with only one object.
- The set of superpixel boundaries should be a superset of object boundaries.
- The mapping from pixels to superpixels should not reduce the achievable performance of the intended application.
- The above properties should be obtained with as few superpixels as possible.

In this paper, we study the superpixel segmentation as a clustering problem. In order to satisfy the above requirements, we present a new clustering objective function which consists of two terms: (1) the entropy rate of a random walk on a graph; (2) a balancing term on the cluster distribution. The entropy rate favors compact and homogeneous clusters— encouraging division of images on perceptual boundaries and favoring superpixels overlapping with only a single object; whereas the balancing term encourages clusters with similar sizes— reducing the number of unbalanced superpixels.

Our clustering formulation leads to an efficient algorithm with a provable bound on the optimality of the solution. We show that our objective function is a monotonically increasing submodular function. Submodularity is the discrete analogue of convexity in continuous domains. Knowing whether a function is submodular enables us to better understand the underlying optimization problem. In general, *maximization* of submodular functions leads to NP-hard problems, for which the global optimum is difficult to obtain. Nevertheless, by using a greedy algorithm and exploiting the matroid structure present in our formulation, we obtain a bound of $\frac{1}{2}$ on the optimality of the solution. Recently, maximization of submodular functions has been used in sensor placement [6] and outbreak detection [9] problems.

1.1. Related Work

Graph-based image segmentation work of Felzenszwalb and Huttenlocher (FH) [4], mean shift [2], and watershed [22] are three of the most popular superpixel segmentation algorithms. FH and watershed are extremely fast;

mean shift is robust to local variations. However, they produce superpixels with irregular sizes and shapes which tend to straddle multiple objects as pointed out in [10, 21].

Ren and Malik [19] propose using Normalized Cut (NCut) [20] for superpixel segmentation. NCut has the nice property of producing superpixels with similar sizes and compact shapes which is preferred for some vision algorithms [19, 14]. One drawback of NCut is its computational requirement— it takes several minutes for segmenting an image of moderate (481x321) size. Levinshtein et al. [10] propose TurboPixel as an efficient alternative to achieve a similar regularity. TurboPixel is based on evolving boundary curves from seeds uniformly placed in the image. Recently Veksler et al. [21] formulate superpixel segmentation as a GraphCut [1] problem. The regularity is enforced through a dense patch assignment technique for allowable pixel labels.

These methods produce nice image tessellations as shown in [19, 10, 21]. Nevertheless, they tend to sacrifice fine image details for their preference for smooth boundaries. This is reflected in the low boundary recall as reported in [10, 21]. In contrast, our balancing objective, which regularizes the cluster sizes, avoids the over-smoothing problem and hence preserves object boundaries.

Moore et al. [13, 12] propose an alternative approach for obtaining superpixels aligned with a grid. In [13], a greedy algorithm is used to sequentially cut images along some vertical and horizontal strips; whereas in [12], the problem is solved with a GraphCut algorithm.

1.2. Contribution

The main contributions of the paper are listed below:

- We pose the superpixel segmentation problem as a maximization problem on a graph and present a novel objective function on the graph topology. This function consists of an entropy rate and a balancing term for obtaining superpixels with commonly desired properties.
- We prove that the entropy rate and the balancing functions are monotonically increasing and submodular.
- By embedding our problem in a matroid structure and using the properties of the objective function, we present an efficient greedy algorithm with an approximation bound of $\frac{1}{2}$.
- Our algorithm significantly outperforms the state of the art with respect to the standard metrics on the Berkeley segmentation benchmark— a reduced under-segmentation error up to 50%, a reduced boundary miss rate up to 40%, and a tighter achievable segmentation accuracy bound. In addition, the presented algorithm is highly efficient— takes about 2.5 seconds to segment an image of size 481x321.

The paper is organized as follows. We review the notations and background materials in Section 2. The problem formulation and the optimization scheme are given in Sections 3 and 4 respectively. We present the experiment results in Section 5 and conclude the paper in Section 6.

2. Preliminaries

Graph representation: We use $G = (V, E)$ to denote an undirected graph where V is the vertex set and E is the edge set. The vertices and edges are denoted by v_i and $e_{i,j}$ respectively. The similarity between vertices is given by the weight function $w : E \to \mathbb{R}^+ \cup \{0\}$. In an undirected graph, the edge weights are symmetric, that is $w_{i,j} = w_{j,i}$.

Graph partition: A graph partition S refers to a division of the vertex set V into disjoint subsets $\mathcal{S} = \{S_1, S_2, ..., S_K\}$ such that $S_i \cap S_j = \emptyset$ for $i \neq j$ and $\bigcup_i S_i = V$. We pose the graph partition problem as a subset selection problem. Our goal is to select a subset of edges $A \in E$ such that the resulting graph (V, A) consists of K connected components/subgraphs.

Entropy: The uncertainty of a random variable is measured by entropy H. Entropy of a discrete random variable X with a probability mass function p_X is defined by

$$H(X) = -\sum_{x \in \mathcal{X}} p_X(x) \log p_X(x) \tag{1}$$

where $\mathcal{X}$ is the support of the random variable X. The conditional entropy $H(X|Y)$ quantifies the remaining uncertainty of a random variable X given that the value of a correlated random variable Y is known. It is defined as

$$\begin{aligned} H(X|Y) &= \sum_{y \in \mathcal{Y}} p_Y(y) H(X|Y = y) \\ &= -\sum_{y \in \mathcal{Y}} p_Y(y) \sum_{x \in \mathcal{X}} p_{X|Y}(x|y) \log p_{X|Y}(x|y) \end{aligned} \tag{2}$$

where $\mathcal{Y}$ is the support of Y and $p_{X|Y}$ is the conditional probability mass function.

Entropy rate: The entropy rate quantifies the uncertainty of a stochastic process $\boldsymbol{X} = \{X_t | t \in T\}$ where T is some index set. For a discrete random process, the entropy rate is defined as an asymptotic measure

$$\mathcal{H}(\boldsymbol{X}) = \lim_{t \to \infty} H(X_t | X_{t-1}, X_{t-2}, ..., X_1), \tag{3}$$

which measures the remaining uncertainty of the random process after observing the past trajectory. For a stationary stochastic process, the limit in (3) always exists. In the case of a stationary $1st$-order Markov process, the entropy rate has a simple form $\mathcal{H}(\boldsymbol{X}) = \lim_{t \to \infty} H(X_t|X_{t-1}) = \lim_{t \to \infty} H(X_2|X_1) = H(X_2|X_1)$. The first equality is due to the $1st$-order Markov property whereas the second equality is a consequence of stationarity. For more details, one can refer to [3, pp.77].

Random walks on graphs: Let $\boldsymbol{X} = \{X_t | t \in T, X_t \in V\}$ be a random walk on the graph $G = (V, E)$ with a nonnegative similarity measure w. We use a random walk model described in [3, pp.78]— the transition probability is defined as

$$p_{i,j} = Pr(X_{t+1} = v_j | X_t = v_i) = w_{i,j}/w_i \quad (4)$$

where $w_i = \sum_{k:e_{i,k} \in E} w_{i,k}$ is the sum of incident weights of the vertex v_i, and the stationary distribution is given by

$$\boldsymbol{\mu} = (\mu_1, \mu_2, ..., \mu_{|V|})^T = (\frac{w_1}{w_T}, \frac{w_2}{w_T}, ..., \frac{w_{|V|}}{w_T})^T \quad (5)$$

where $w_T = \sum_{i=1}^{|V|} w_i$ is the normalization constant. For a disconnected graph, the stationary distribution is not unique. However, $\boldsymbol{\mu}$ in (5) is always a stationary distribution. It can be easily verified through $\boldsymbol{\mu} = P^T \boldsymbol{\mu}$ where $P = [p]_{i,j}$ is the transition matrix. The entropy rate of the random walk can be computed by applying (2)

$$\begin{aligned} \mathcal{H}(\boldsymbol{X}) &= H(X_2|X_1) = \sum_i \mu_i H(X_2|X_1 = v_i) \\ &= -\sum_i \mu_i \sum_j p_{i,j} \log p_{i,j} = -\sum_i \frac{w_i}{w_T} \sum_j \frac{w_{i,j}}{w_i} \log \frac{w_{i,j}}{w_i} \\ &= -\sum_i \sum_j \frac{w_{i,j}}{w_T} \log \frac{w_{i,j}}{w_T} + \sum_i \frac{w_i}{w_T} \log \frac{w_i}{w_T} \end{aligned} \quad (6)$$

Submodularity: Let E be a finite set. A set function $F : 2^E \rightarrow \mathbb{R}$ is submodular if

$$F(A \cup \{a_1\}) - F(A) \geq F(A \cup \{a_1, a_2\}) - F(A \cup \{a_2\}) \quad (7)$$

for all $A \subseteq E$, $a_1, a_2 \in E$ and $a_1, a_2 \notin A$. This property is also referred as the diminishing return property, which says that the impact of a module is less if used in a later stage.

Monotonically increasing set function: A set function F is monotonically increasing if $F(A_1) \leq F(A_2)$ for all $A_1 \subseteq A_2$.

Matroid: A matroid is an ordered pair $M = (E, \mathcal{I})$ consisting of a finite set E and a collection $\mathcal{I}$ of subsets of E satisfying the following three conditions: (1) $\emptyset \in \mathcal{I}$, (2) If $I \in \mathcal{I}$ and $I' \subseteq I$, then $I' \in \mathcal{I}$, and (3) If I_1 and I_2 are in $\mathcal{I}$ and $|I_1| < |I_2|$, then there is an element e of $I_2 - I_1$ such that $I_1 \cup e \in \mathcal{I}$. Note that there are several other definitions for matroids which are equivalent. For more details, one can refer to [17, pp.7∼15].

Later in the paper we prove that our objective function is monotonically increasing and submodular.

3. Problem Formulation

We consider clustering as a graph partitioning problem. To partition the image into K superpixels, we search for a graph topology that has K connected subgraphs and maximizes the proposed objective function.

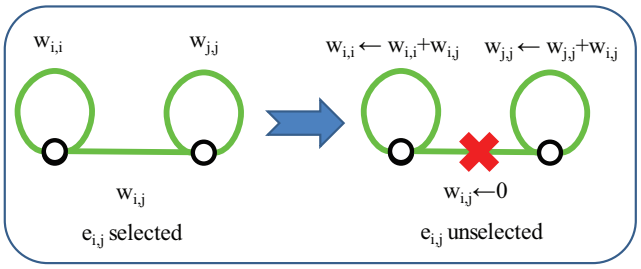


Figure 1. Illustration of the graph construction. If an edge $e_{i,j}$ is unselected in cluster formation, its weight is redistributed to the loops of the two vertices.

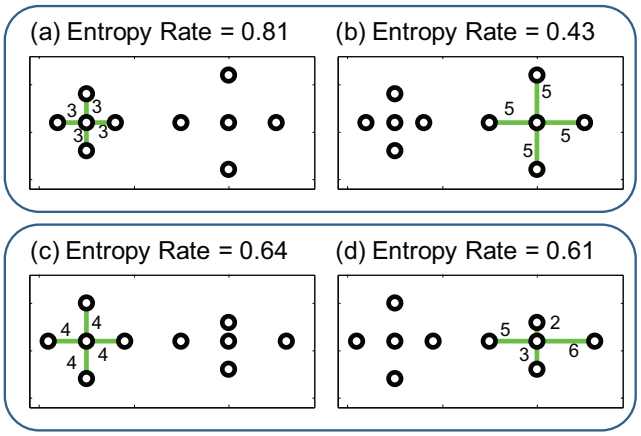


Figure 2. We show the role of entropy rate in obtaining compact and homogeneous clustering. We use a Gaussian kernel to convert the distances, the numbers next to the edges, to similarities. Each of these clustering outputs contains six different clusters shown as connected components. As described in Section 3, every vertex has a loop which is not shown. The entropy rate of the compact cluster in (a) has a higher objective value than that of the less compact one in (b). The entropy rate of the homogeneous cluster in (c) has a higher objective value than that of the less homogeneous one in (d).

3.1. Graph Construction

We map an image to a graph $G = (V, E)$ with vertices denoting the pixels and the edge weights denoting the pairwise similarities given in the form of a similarity matrix. Our goal is to select a subset of edges $A \subseteq E$ such that the resulting graph, $G = (V, A)$, contains exactly K connected subgraphs. In addition, we also assume that every vertex of the graph has a self loop, although they are not necessary for the graph partition problem. When an edge is not included in A, we increase the edge weight of the self loop of the associated vertices in such a way that the total incident weight for each vertex remains constant (See Figure 1).

3.2. Entropy Rate

We use the entropy rate of the random walk on the constructed graph as a criterion to obtain compact and homogeneous clusters. The proposed construction leaves the stationary distribution of the random walk (5) unchanged

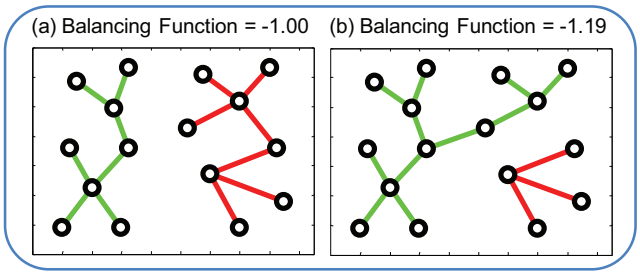


Figure 3. We show the role of the balancing function in obtaining clusters of similar sizes. The connected components show the different clusters. The balancing function has a higher objective value for the balanced clustering in (a) compared to the less balanced one in (b).

where the set functions for the transition probabilities $p_{i,j} : 2^E \to \mathbb{R}$ are given below:

$$p_{i,j}(A) = \begin{cases} \frac{w_{i,j}}{w_i} & \text{if } i \neq j \text{ and } e_{i,j} \in A, \\ 0 & \text{if } i \neq j \text{ and } e_{i,j} \notin A, \\ 1 - \frac{\sum_{j:e_{i,j}\in A} w_{i,j}}{w_i} & \text{if } i = j. \end{cases} \tag{8}$$

Consequently, the entropy rate of the random walk on $G = (V, A)$ can be written as a set function:

$$\mathcal{H}(A) = -\sum_i \mu_i \sum_j p_{i,j}(A) \log(p_{i,j}(A)) \tag{9}$$

Although inclusion of any edge in set A increases the entropy rate, the increase is larger when selecting edges that form compact and homogeneous clusters, as shown in Figure 2.

We establish the following result on the entropy rate of the random walk model.

Theorem 1. *The entropy rate of the random walk on the graph $\mathcal{H} : 2^E \to \mathbb{R}$ is a monotonically increasing submodular function under the proposed graph construction.*

It is easy to see that the entropy rate is monotonically increasing, since the inclusion of any edge increases the uncertainty of a jump of the random walk. The diminishing return property comes from the fact that the increase in uncertainty from selecting an edge is less in a later stage because it is shared with more edges. The details of the proof is given in Appendix A.

3.3. Balancing Function

We utilize a balancing function that encourages clusters with similar sizes. Let A be the selected edge set, N_A is the number of connected components in the graph, and Z_A be the distribution of the cluster membership. For instance, let the graph partitioning for the edge set A be $\mathcal{S}_A = \{S_1, S_2, ..., S_{N_A}\}$. Then the distribution of Z_A is equal to

$$p_{Z_A}(i) = \frac{|S_i|}{|V|}, i = \{1, ..., N_A\}, \tag{10}$$

and the balancing term is given by

$$\mathcal{B}(A) \equiv H(Z_A) - N_A = -\sum_i p_{Z_A}(i) \log(p_{Z_A}(i)) - N_A. \tag{11}$$

The entropy $H(Z_A)$ favors clusters with similar sizes; whereas N_A favors fewer number of clusters. In Figure 3 we show an example of this preference where a more balanced partitioning is preferred for a fixed number of clusters.

Similar to the entropy rate, the balancing function is also a monotonically increasing and submodular function as shown in the following theorem:

Theorem 2. *The balancing function $\mathcal{B} : 2^E \to \mathbb{R}$ is a monotonically increasing submodular function under the proposed graph construction.*

The proof is given in Appendix B.

The objective function combines the entropy rate and the balancing function and therefore favors compact, homogeneous, and balanced clusters. The clustering is achieved via optimizing the objective function with respect to the edge set:

$$\begin{aligned} \max_A \quad & \mathcal{H}(A) + \lambda \mathcal{B}(A) \\ \text{subject to} \quad & A \subseteq E \text{ and } N_A \geq K, \end{aligned} \tag{12}$$

where $\lambda \geq 0$ is the weight of the balancing term. Linear combination with nonnegative coefficients preserves submodularity and monotonicity [15], therefore the objective function is also submodular and monotonically increasing. The additional constraint on the number of connected subgraphs enforces exactly K clusters since the objective function is monotonically increasing.

4. Optimization

In this section, we present a greedy optimization scheme for the proposed objective function and analyze its optimality and complexity.

4.1. Greedy Heuristic

One standard approach for maximizing a submodular function is through a greedy algorithm [15]. The algorithm starts with an empty set (a fully disconnected graph, $A = \emptyset$) and sequentially adds edges to the set. At each iteration, it adds the edge that yields the largest gain. The iterations are stopped when the number of connected subgraphs reaches a preset number, $N_A = K$.

In order to achieve additional speedup, we put an additional constraint on the edge set A such that it can not include cycles. This constraint immediately ignores additional edges within a connected subgraph and reduces the

number of evaluations in the greedy search. Notice that these edges do not change the partitioning of the graph. Although this constraint leads to a smaller solution space (only tree-structure subgraphs are allowed) compared to the original problem, in practice the clustering results are very similar.

This cycle-free constraint together with the cluster number constraint $N_A \geq K$ leads to an independent set definition which induces a matroid $M = (E, \mathcal{I})$. We prove this via the following theorem:

Theorem 3. *Let E be the edge set, and let $\mathcal{I}$ be the set of subsets $A \subseteq E$ which satisfies: (1) A is cycle-free and (2) A constitutes a graph partition with more than or equal to K connected components. Then the pair $M = (E, \mathcal{I})$ is a matroid.*

The proof is given in Appendix C.

Maximization of a submodular function subject to a matroid constraint has been an active subject in combinatorial optimization; it is shown in Fisher et al. [5] that the greedy algorithm gives an $\frac{1}{2}$ approximation bound. Following the same argument, we achieve the same ($\frac{1}{2}$ approximation) guarantee on the proposed greedy algorithm. A pseudocode is given in Algorithm 1.

Data: $G = (V, E)$, $w : E \to \mathbb{R}^+$, K, and λ
Result: A
$A \leftarrow \emptyset$
for $N(A) > K$ **do**

$$\hat{a} \leftarrow \underset{\{a\} \cup A \in \mathcal{I}}{arg\,max} \; \mathcal{F}(A \cup \{a\}) - \mathcal{F}(A)$$
$$A \leftarrow A \cup \{\hat{a}\}$$

end

Algorithm 1: Pseudocode of the greedy algorithm. The objective function is defined as $\mathcal{F} \equiv \mathcal{H} + \lambda\mathcal{B}$.

4.2. Efficient Implementation

In each iteration, the greedy algorithm selects the edge that yields the largest gain in the objective function subject to the matroid constraint. A naive implementation of the algorithm, as given in Algorithm 1, loops $O(|E|)$ times to add a new edge into A. At each loop, it scans through the edge list to locate the edge with the largest gain; therefore the complexity of the algorithm is $O(|E|^2)$[1]. Since we map an image into a grid graph (8-connected), the complexity of the algorithm is $O(|V|^2)$. By exploiting the submodularity of the objective function, we can achieve a more efficient implementation which is called lazy greedy [9].

[1]Note that an edge gain can be computed in constant time.

Initially, we compute the gain of adding each edge to A and construct a max heap structure. At each iteration, the edge with the maximum gain is popped from the heap and included to A. The inclusion of this edge affect the gains of some of the remaining edges in the heap; therefore, the heap needs to be updated. However, the submodular property allows an efficient update of the heap structure. The key observation is that, throughout the algorithm, the gain for each edge can never increase— the diminishing return property. Therefore, it is sufficient to keep a heap structure where the gain of the top element is updated but not necessarily the others. Since the top element of the heap is updated and the values for the other elements can only decrease, the top element is the maximum value.

Although the worst case complexity of the lazy greedy algorithm is $O(|V|^2 \log |V|)$, in practice the algorithm runs much faster than the naive implementation. On average, very few updates are performed on the heap at each iteration, and hence the complexity of the algorithm approximates $O(|V| \log |V|)$. In our experiments, it provides a speedup by a factor of 200–300 for image size 481x321 and on average requires 2.5 seconds.

5. Experiments

We conducted the experiments on the Berkeley segmentation benchmark [11]. The benchmark contains 300 images with human-labeled ground truth segmentations.

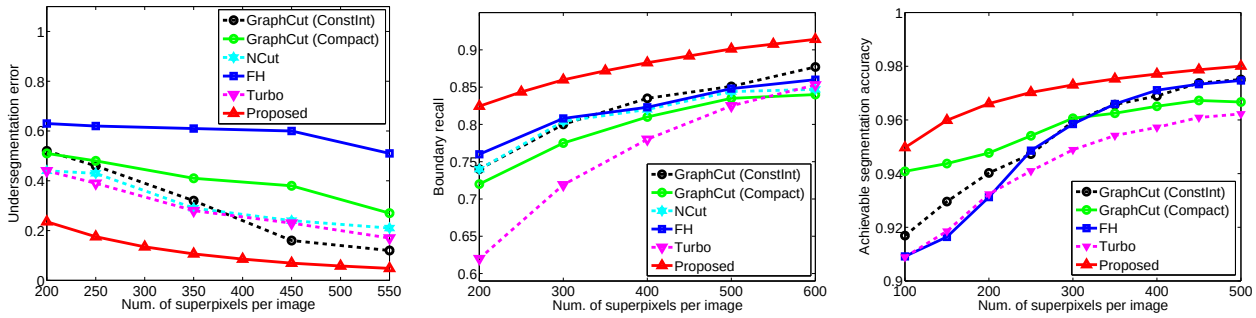
Superpixel segmentation has a different goal than object segmentation, therefore the performance metrics are also different. We use three standard metrics which were commonly used for evaluating the quality of superpixels: undersegmentation error [10, 21], boundary recall [19] and achievable segmentation accuracy [16]. For the sake of completeness we first describe these metrics. We use $\mathcal{G} = \{G_1, G_2, ..., G_{n_\mathcal{G}}\}$ to represent a ground truth segmentation with n_G segments and $|G_i|$ denotes the segment size.

- **Undersegmentation error (UE)** measures fraction of pixel leak across ground truth boundaries. It evaluates the quality of segmentation based on the requirement that a superpixel should overlap with only one object. We utilize the undersegmentation error metric used in Veksler et al. [21],

$$UE_\mathcal{G}(\mathcal{S}) = \frac{\sum_i \sum_{k:S_k \cap G_i \neq \emptyset} |S_k - G_i|}{\sum_i |G_i|}. \quad (13)$$

  For each ground truth segment G_i we find the overlapping superpixels S_k's and compute the size of the pixel leaks $|S_k - G_i|$'s. We then sum the pixel leaks over all the segments and normalize it by the image size $\sum_i |G_i|$.

- **Boundary recall (BR)** measures the percentage of the natural boundaries recovered by the superpixel bound-



(a) (b) (c)

Figure 4. Performance metrics: (a) undersegmentation error curves (b) boundary recall curves (c) achievable segmentation accuracy curves. The proposed algorithm performs significantly better than the state of the art in all the metrics at all the superpixel counts.

Figure 5. Superpixel segmentation examples. The images contain 100 superpixels. The ground truth segments are color-coded and blended on the images. The superpixels (boundaries shown in white) respect object boundaries and tend to divide an image into similar-sized regions.

aries. We compute BR using

$$BR_{\mathcal{G}}(\mathcal{S}) = \frac{\sum_{p\in\delta\mathcal{G}} \mathbb{I}(\min_{q\in\delta\mathcal{S}}\|p-q\| < \epsilon)}{|\delta\mathcal{G}|}, \quad (14)$$

which is the ratio of ground truth boundaries that have a nearest superpixel boundary within an ϵ-pixel distance. We use $\delta\mathcal{S}$ and $\delta\mathcal{G}$ to denote the union sets of superpixel boundaries and ground truth boundaries respectively. The indicator function $\mathbb{I}$ checks if the nearest pixel is within ϵ distance. In our experiments we set $\epsilon = 2$.

- **Achievable segmentation accuracy (ASA)** is a performance upperbound measure. It gives the highest accuracy achievable for object segmentation that utilizes superpixels as units. To compute ASA we label each superpixel with the label of the ground truth segment that has the largest overlap. The fraction of correctly labeled pixels is the achievable accuracy,

$$ASA_{\mathcal{G}}(\mathcal{S}) = \frac{\sum_k \max_i |S_k \cap G_i|}{\sum_i |G_i|}. \quad (15)$$

These performance metrics are plotted against the number of superpixels in an image. Algorithms producing better performances with a smaller number of superpixels is more preferable.

We use a Gaussian kernel to convert pixel differences to similarities $exp(-\frac{d(v_i,v_j)^2}{2\sigma^2})$ where $d(v_i, v_j)$ is defined as the intensity difference multiplied by the spatial distance. We also present a method to automatically adjust the balancing parameter λ. Given an initial user-specified value λ', the final balancing parameter λ is adjusted based on: (1) the number of superpixels K and (2) a data dependent dynamic parameter β which is computed from the input image. The cluster number K is introduced for emphasizing more on the balancing term when large numbers of superpixels are required. The data dependent term is given by the ratio of the maximal entropy rate increase and the maximal balancing term increase upon including a single edge into the graph $\beta = \frac{\max_{e_{i,j}} \mathcal{H}(e_{i,j})-\mathcal{H}(\emptyset)}{\max_{e_{i,j}} \mathcal{B}(e_{i,j})-\mathcal{B}(\emptyset)}$ and compensates for the magnitude difference between the two terms in the objective function. The final balancing parameter is given by $\lambda = \beta K \lambda'$. Throughout the experiments we use $\lambda' = 0.5$

and Gaussian kernel bandwidth $\sigma = 5.0$ and generate the results. Later in the section we analyze the effect of these parameters on the results.

In the first experiment, we compare our results with FH [4], GraphCut superpixel [21], Turbopixels [10] and NCut superpixel [19] methods using the three evaluation metrics. The results were obtained by averaging over all the 300 gray images in the dataset.

Figure 4(a) shows the undersegmentation error curves. The curves for the other methods are duplicated from the original paper [21]. The proposed algorithm outperforms the state of the art at all the superpixel counts where the error rate is reduced by more than 50%. It achieves an undersegmentation error of 0.13 with 350 superpixels while the same performance is achieved with 550 superpixels using GraphCut superpixel segmentation [21]. With 550 superpixels, our undersegmentation error is 0.06.

In Figure 4(b), we plot the boundary recall curves. Again, the curves for the other methods are duplicated from the original paper [21]. The proposed algorithm reduces the missed boundaries by more than 30% compared to the state of the art at all the superpixel counts. The recall rates of the presented algorithm are 82% and 92% with 200 and 600 superpixels respectively. The recall rates with the same superpixel counts are 76 and 86 percents with FH.

In Figure 4(c), we plot the achievable segmentation accuracy curves. In this experiment we generated the curves for the other methods using the original implementations. The proposed algorithm yields a better achievable segmentation upperbound at all the superpixel counts—particularly for smaller number of superpixels. The ASA is 95% with 100 superpixels where the same accuracy can only be achieved with 200 superpixels for the other algorithms.

In the second experiment, we evaluate the segmentation results visually. Several examples are shown in Figure 5 where the images are partitioned into 100 superpixels. For better visualization, the ground truth segments are color-coded and blended on the images, and the superpixel boundaries recovered by the algorithm are superimposed in white color. It is difficult to notice pixel leaks and the superpixels tend to divide an image into similar-sized regions which are important for region based feature descriptors.

In Figure 6, we show an example of superpixel hierarchy. The proposed algorithm starts with each pixel as a separate cluster and gradually combines clusters to construct larger superpixels. This agglomerative nature generates a superpixel hierarchy during segmentation. The hierarchy is useful for many vision applications such as interactive editing or algorithms that utilize information from multiple superpixel segmentations. One such example is presented in Kohli et al. [8].

In the third experiment, we analyze the effects of the balancing term, λ', and the kernel bandwidth, σ, parameters

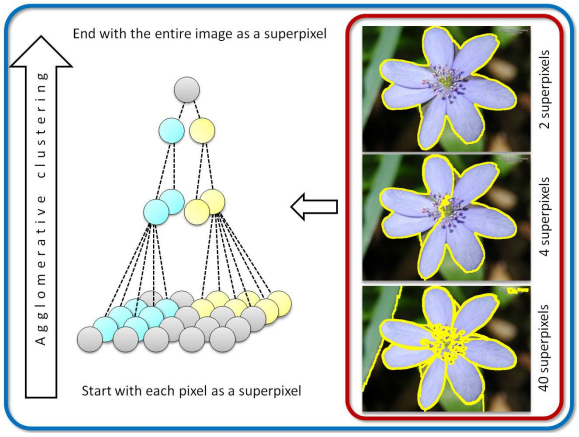


Figure 6. Superpixel hierarchy. The proposed algorithm generates a superpixel hierarchy during segmentation. The figure shows the segmentations with 40, 4, and 2 superpixels, their corresponding levels in the hierarchy, and the merging from the lower level superpixels to the higher level superpixels.

on the quality of segmentation. We observe that competitive segmentation results are achieved with a wide range of parameter selection.

In Figure 7, we plot the performance curves with different λ' values for a fixed $\sigma = 5.0$. We observed that smaller λ' results in better boundary recall rates especially for smaller superpixel counts, while the results are largely invariant to this parameter for larger superpixel counts. We further observed that better performances on undersegmentation error and achievable segmentation accuracy are achieved with a larger λ'. In general, there is a tradeoff among different metrics based on the λ' parameter, and empirically we found that $\lambda' = 0.5$ yields a good compromise among these metrics.

In Figure 8, we plot the performance curves with different σ values for a fixed $\lambda' = 0.5$. We observed that a large range of σ values results in comparable performances, namely from 0.5 to 5. The superpixels are largely insensitive to the selection of the σ parameter.

The proposed algorithm is among the fastest superpixel segmentation algorithms and takes an average of 2.5 seconds to segment an image on the Berkeley benchmark (481×321 pixels) on an Intel Core 2 Duo E8400 processor. Compared to the state of the art methods, it is faster than the Graphcut superpixel [21] (6.4 seconds), turbopixel [10](15 seconds), and NCut (5 minutes), whereas it is slower than FH [4](0.5 seconds).

6. Conclusion

We formulated the superpixel segmentation problem as an optimization problem on graph topology. We proposed a novel objective function based on the entropy rate of a random walk on the graph. We derived an efficient algorithm

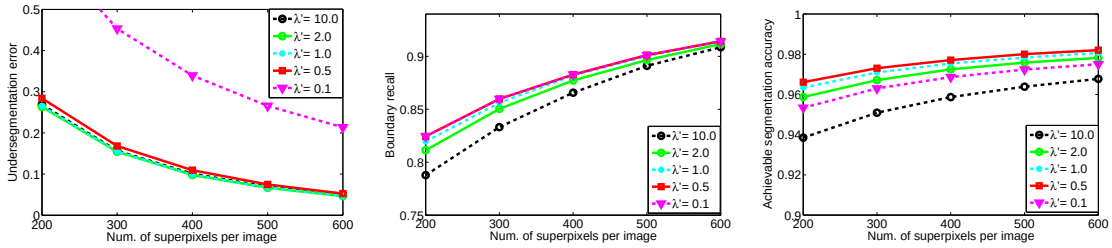


Figure 7. Effect of the balancing preference on the performance metrics.

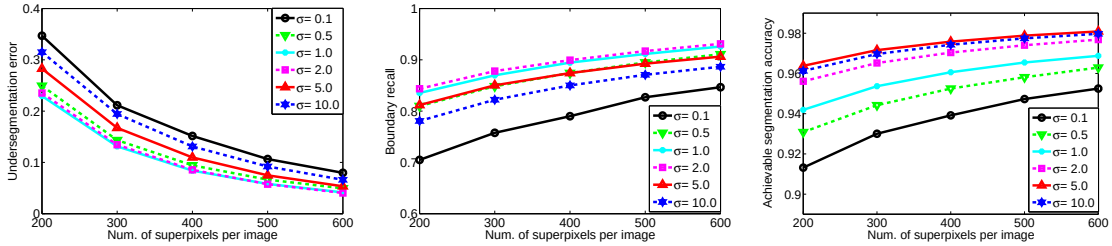


Figure 8. Effect of the kernel bandwidth on the performance metrics.

with a bound on the optimality of the solution. In future, we plan to investigate the applicability of the proposed formulation for general clustering problems.

7. Acknowledgement

We thank Dr. David W. Jacobs and Dr. Yuichi Taguchi for their valuable discussions and suggestions. This research was mostly conducted at MERL with support from MERL. Prof. Rama Chellappa was supported by MURI from the Office of Naval Research under the Grant N00014-10-1-0934.

A. Proof of Theorem 1

The proof contains two parts. The first part proves the monotonic increasing property. In the second part, we prove the submodularity.

A.1. Proof of the monotonically increasing property

Proof. Let $A \subseteq E$ be a subset of edges and $a_1 \in E$ be any edge. We prove the monotonically increasing property by showing

$$\mathcal{H}(A \cup \{a_1\}) - \mathcal{H}(A) \geq 0. \tag{16}$$

WLOG we can assume that $a_1 = e_{1,2}$. Given $A \cup \{e_{1,2}\}$ is selected, the resulting loop weights for vertices v_1 and v_2 are given by

$$c_1 \equiv w_1 - \sum_{k:e_{1,k} \in A \cup \{e_{1,2}\}} w_{1,k} \tag{17}$$

and

$$c_2 \equiv w_2 - \sum_{k:e_{2,k} \in A \cup \{e_{1,2}\}} w_{2,k} \tag{18}$$

respectively. Applying the notations in (8), the increase in the entropy rate due to the addition of $e_{1,2}$ is equal to

$$\mathcal{H}(A \cup \{e_{1,2}\}) - \mathcal{H}(A) \tag{19}$$

$$= -\sum_i \mu_i \sum_j p_{i,j}(A \cup \{e_{1,2}\}) \log p_{i,j}(A \cup \{e_{1,2}\}) + \sum_i \mu_i \sum_j p_{i,j}(A) \log p_{i,j}(A) \tag{20}$$

$$= -\sum_i \sum_j \frac{w_i p_{i,j}(A \cup \{e_{1,2}\})}{w_T} \log p_{i,j}(A \cup \{e_{1,2}\}) - \sum_i \sum_j \frac{w_i p_{i,j}(A \cup \{e_{1,2}\})}{w_T} \log \frac{w_i}{w_T} + \sum_i \sum_j \frac{w_i p_{i,j}(A)}{w_T} \log \frac{w_i}{w_T} + \sum_i \sum_j \frac{w_i p_{i,j}(A)}{w_T} \log p_{i,j}(A)$$

$$= -\sum_i \sum_j \frac{w_i p_{i,j}(A \cup \{e_{1,2}\})}{w_T} \log \frac{w_i p_{i,j}(A \cup \{e_{1,2}\})}{w_T} + \sum_i \sum_j \frac{w_i p_{i,j}(A)}{w_T} \log \frac{w_i p_{i,j}(A)}{w_T} \tag{21}$$

Note that

$$
\begin{aligned}
&-\sum_i \sum_j \frac{w_i p_{i,j}(A\cup\{e_{1,2}\})}{w_T}\log\frac{w_i}{w_T} \\
&+\sum_i \sum_j \frac{w_i p_{i,j}(A)}{w_T}\log\frac{w_i}{w_T} && (22)\\
=&-\sum_i \frac{w_i}{w_T}\log\frac{w_i}{w_T}\sum_j p_{i,j}(A\cup\{e_{1,2}\}) \\
&+\sum_i \frac{w_i}{w_T}\log\frac{w_i}{w_T}\sum_j p_{i,j}(A) && (23)\\
=&0. && (24)
\end{aligned}
$$

by the definition of the transition probability $\sum_j p_{i,j}(A\cup\{e_{1,2}\}) = \sum_j p_{i,j}(A) = 1$. With some simple algebraic manipulations, (21) is reduced to

$$
\begin{aligned}
&= \Big\{\frac{w_{1,2}+c_1}{w_T}\log(\frac{w_{1,2}+c_1}{w_T}) - \frac{w_{1,2}}{w_T}\log(\frac{w_{1,2}}{w_T}) \\
&\quad -\frac{c_1}{w_T}\log(\frac{c_1}{w_T})\Big\} + \Big\{\frac{w_{2,1}+c_2}{w_T}\log(\frac{w_{2,1}+c_2}{w_T}) \\
&\quad -\frac{w_{2,1}}{w_T}\log(\frac{w_{2,1}}{w_T}) - \frac{c_2}{w_T}\log(\frac{c_2}{w_T})\Big\} && (25)\\
&= \frac{w_{1,2}+c_1}{w_T}\Big\{\frac{w_{1,2}}{w_{1,2}+c_1}\log(\frac{w_{1,2}+c_1}{w_{1,2}}) + \frac{c_1}{w_{1,2}+c_1}\cdot \\
&\quad \log(\frac{w_{1,2}+c_1}{c_1})\Big\} + \frac{w_{2,1}+c_2}{w_T}\Big\{\frac{w_{2,1}}{w_{2,1}+c_2}\log(\frac{w_{2,1}+c_2}{w_{2,1}}) \\
&\quad + \frac{c_2}{w_{2,1}+c_2}\log(\frac{w_{2,1}+c_2}{c_2})\Big\} && (26)\\
&\geq 0. && (27)
\end{aligned}
$$

Note that the terms within the two curly bracket pairs in (26) compute the entropy of the binary random variables and are nonnegative. □

A.2. Proof of the submodularity

Proof. We prove the submodularity by showing

$$\mathcal{H}(A\cup\{a_1\}) - \mathcal{H}(A) \geq \mathcal{H}(A\cup\{a_1,a_2\}) - \mathcal{H}(A\cup\{a_2\}). \quad (28)$$

Based on whether these edges share a common vertex or not, we have the two following cases.

Case (1): Let assume a_1 and a_2 have no common vertex. WLOG, we can assume that $a_1 = e_{1,2}$ and $a_2 = e_{3,4}$. Since the addition of $e_{3,4}$ has no effect on the loop weights of v_1 and v_2, we have

$$c_1 = w_1 p_{1,1}(A\cup\{e_{1,2},e_{3,4}\}) = w_1 p_{1,1}(A\cup\{e_{1,2}\}) \quad (29)$$

and

$$c_2 = w_2 p_{2,2}(A\cup\{e_{1,2},e_{3,4}\}) = w_2 p_{2,2}(A\cup\{e_{1,2}\}) \quad (30)$$

The increase in the entropy rate due to the addition of $e_{1,2}$ to $A\cup\{e_{3,4}\}$ is given by

$$
\begin{aligned}
&\mathcal{H}(A\cup\{e_{1,2},e_{3,4}\}) - \mathcal{H}(A\cup\{e_{3,4}\}) && (31)\\
&= \Big\{\frac{w_{1,2}+c_1}{w_T}\log(\frac{w_{1,2}+c_1}{w_T}) - \frac{w_{1,2}}{w_T}\log(\frac{w_{1,2}}{w_T}) \\
&\quad -\frac{c_1}{w_T}\log(\frac{c_1}{w_T})\Big\} + \Big\{\frac{w_{2,1}+c_2}{w_T}\log(\frac{w_{2,1}+c_2}{w_T}) \\
&\quad -\frac{w_{2,1}}{w_T}\log(\frac{w_{2,1}}{w_T}) - \frac{c_2}{w_T}\log(\frac{c_2}{w_T})\Big\} && (32)\\
&= \mathcal{H}(A\cup\{e_{1,2}\}) - \mathcal{H}(A) && (33)
\end{aligned}
$$

Case (2): Let assume a_1 and a_2 share a common vertex. WLOG, we can assume that $a_1 = e_{1,2}$ and $a_2 = e_{1,3}$ where v_1 is the shared vertex. The updated loop weights d_1 and d_2 for the vertices v_1 and v_2 are now given by

$$d_1 = w_1 - \sum_{k:e_{1,k}\in A\cup\{e_{1,2},e_{1,3}\}} w_{1,k} = c_1 - w_{1,3} \quad (34)$$

and

$$d_2 = w_2 - \sum_{k:e_{2,k}\in A\cup\{e_{1,2},e_{1,3}\}} w_{2,k} = c_2 \quad (35)$$

respectively. The increase in the entropy rate due to the addition of $e_{1,2}$ to $A\cup\{e_{3,4}\}$ is equal to

$$
\begin{aligned}
&\mathcal{H}(A\cup\{e_{1,2},e_{1,3}\}) - \mathcal{H}(A\cup\{e_{1,3}\}) && (36)\\
&= \Big\{\frac{w_{1,2}+d_1}{w_T}\log(\frac{w_{1,2}+d_1}{w_T}) - \frac{w_{1,2}}{w_T}\log(\frac{w_{1,2}}{w_T}) \\
&\quad -\frac{d_1}{w_T}\log(\frac{d_1}{w_T})\Big\} + \Big\{\frac{w_{2,1}+d_2}{w_T}\log(\frac{w_{2,1}+d_2}{w_T}) \\
&\quad -\frac{w_{2,1}}{w_T}\log(\frac{w_{2,1}}{w_T}) - \frac{d_2}{w_T}\log(\frac{d_2}{w_T})\Big\} && (37)
\end{aligned}
$$

By subtracting (37) from (25), we have

$$
\begin{aligned}
&\big\{\mathcal{H}(A\cup\{e_{1,2}\}) - \mathcal{H}(A)\big\} \\
&\quad -\big\{\mathcal{H}(A\cup\{e_{1,2},e_{1,3}\}) - \mathcal{H}(A\cup\{e_{1,3}\})\big\} && (38)\\
&= \Big\{\frac{w_{1,2}+d_1+w_{1,3}}{w_T}\log(\frac{w_{1,2}+d_1+w_{1,3}}{w_T}) \\
&\quad -\frac{d_1+w_{1,3}}{w_T}\log(\frac{d_1+w_{1,3}}{w_T})\Big\} \\
&\quad -\Big\{\frac{w_{1,2}+d_1}{w_T}\log(\frac{w_{1,2}+d_1}{w_T}) - \frac{d_1}{w_T}\log(\frac{d_1}{w_T})\Big\} && (39)\\
&= g(\frac{d_1+w_{1,3}}{w_T}) - g(\frac{d_1}{w_T}) > 0 && (40)
\end{aligned}
$$

The last equation is an application of the strictly increasing property of

$$g(x) = (x+\xi)\log(x+\xi) - x\log(x) \quad (41)$$

where $\xi = \frac{w_{1,2}}{w_T}$ in this case.

From (32) and (40), we arrive the proof. □

B. Proof of Theorem 2

The proof contains two parts. The first part proves that $\mathcal{B}$ is monotonically increasing. In the second part, we show that $\mathcal{B}$ is submodular.

B.1. Proof of the monotonically increasing property

Proof. Let $A \subseteq E$ be a subset of edges and $a_1 \in E$ be any edge. We prove the monotonically increasing property

$$\mathcal{B}(A \cup \{a_1\}) - \mathcal{B}(A) \geq 0. \tag{42}$$

WLOG we can assume that that $a_1 = e_{1,2}$ and that v_1 and v_2 be in the clusters S_i and S_j respectively. From (10), we know that the probability that a randomly picked vertex is in the clusters S_i is equal to

$$p_{Z_A}(i) = \frac{|S_i|}{|V|}. \tag{43}$$

Similarly,

$$p_{Z_A}(j) = \frac{|S_j|}{|V|}. \tag{44}$$

To further simplify the notations, denote $p_i = p_{Z_A}(i)$ and $p_j = p_{Z_A}(j)$. We only need to consider the case that the addition of $e_{1,2}$ combines two clusters since the balancing term remains the same otherwise.

Let assume the two vertices v_1 and v_2 are in two different clusters under A; i.e., $i \neq j$. Therefore, the addition of $e_{1,2}$ will merge S_i and S_j and the total number of clusters is decreased by 1. The probability that a randomly picked vertex is in the newly merged cluster is given by

$$p_{Z_A}(i) + p_{Z_A}(j) = \frac{|S_i| + |S_j|}{|V|}. \tag{45}$$

The increase in the balancing function can be computed via

$$\mathcal{B}(A \cup \{e_{1,2}\}) - \mathcal{B}(A) \tag{46}$$

$$= \Big\{ H(A \cup \{e_{1,2}\}) - H(A) \Big\} - \Big\{ N_{\mathcal{A} \cup \{e_{1,2}\}} - N_{\mathcal{A}} \Big\} \tag{47}$$

$$= -(p_i + p_j)\log(p_i + p_j) + p_i \log p_i + p_j \log p_j + 1 \tag{48}$$

$$= p_i \log \frac{p_i}{p_i + p_j} + p_j \log \frac{p_j}{p_i + p_j} + 1 \tag{49}$$

$$\geq (p_i + p_j)\log(\frac{p_i + p_j}{2(p_i + p_j)}) + 1 \tag{50}$$

$$= -(p_i + p_j) + 1 \tag{51}$$

$$\geq 0. \tag{52}$$

Note that the inequality in (50) is an application of the log-sum inequality. We have proven the monotonically increasing property. □

B.2. Proof of the submodularity

Proof. We prove the submodularity by showing

$$\mathcal{B}(A \cup \{a_1\}) - \mathcal{B}(A) \geq \mathcal{B}(A \cup \{a_1, a_2\}) - \mathcal{B}(A \cup \{a_2\}). \tag{53}$$

We are only interested in the cases that a_1 combines two clusters. For the case the a_1 does not combine clusters, both sides of (53) are zero and the diminishing return property trivially holds.

WLOG we can assume that $a_1 = e_{1,2}$ and $a_2 = e_{3,4}$. Depending on whether the addition of $e_{3,4}$ combines clusters or not, we need to discuss the following four cases.

Case 1: Let assume the addition of $e_{3,4}$ combines the clusters S_i and S_j. This means v_1 and v_2 are in the same cluster given $A \cup \{e_{3,4}\}$. Therefore the addition of $e_{1,2}$ has no effect on the graph partition. Both the number of clusters and the cluster membership distribution remain the same; the increase in the balancing function is zero

$$\mathcal{B}(A \cup \{e_{1,2}\}) - \mathcal{B}(A) \geq \mathcal{B}(A \cup \{e_{1,2}, e_{3,4}\}) - \mathcal{B}(A \cup \{e_{3,4}\}) = 0. \tag{54}$$

Case 2: Let assume the addition of $e_{3,4}$ combines some other clusters S_k and S_m where $|\{k, m\} \cap \{i, j\}| = 0$. Under this assumption the addition of $e_{1,2}$ merges the clusters S_i and S_j. From (48) and the relations,

$$p_{Z_{A \cup \{e_{3,4}\}}}(i) = p_{Z_A}(i) = \frac{|S_i|}{|V|} \tag{55}$$

and

$$p_{Z_{A \cup \{e_{3,4}\}}}(j) = p_{Z_A}(j) = \frac{|S_j|}{|V|}, \tag{56}$$

we have the following equality

$$\mathcal{B}(A \cup \{e_{1,2}\}) - \mathcal{B}(A) = \mathcal{B}(A \cup \{e_{1,2}, e_{3,4}\}) - \mathcal{B}(A \cup \{e_{3,4}\}) = 0. \tag{57}$$

Case 3: Let assume the addition of $e_{3,4}$ combines a cluster to either S_i or S_j. In this case, the addition of $e_{1,2}$ merges S_i and S_j. However, the cluster membership distribution are different. WLOG, let us assume that $e_{3,4}$ combines the clusters S_k and S_i. Let us also denote $p_k = p_{Z_A}(k)$. The increase in the balancing function due to the addition of $e_{1,2}$ can then be computed in

$$\mathcal{B}(A \cup \{e_{1,2}, e_{3,4}\}) - \mathcal{B}(A \cup \{e_{3,4}\}) \tag{58}$$

$$= -(p_i + p_j + p_k)\log(p_i + p_j + p_k) + (p_i + p_k)\log(p_i + p_k) + p_j \log(p_j) + 1 \tag{59}$$

By comparing (59) and (48), we have

$$\Big\{\mathcal{B}(A\cup\{e_{1,2}\})-\mathcal{B}(A)\Big\}-\Big\{\mathcal{B}(A\cup\{e_{1,2},e_{3,4}\}) - \mathcal{B}(A\cup\{e_{3,4}\})\Big\} \tag{60}$$

$$= (p_i+p_j+p_k)\log(p_i+p_j+p_k)-(p_i+p_j)\log(p_i+p_j) - ((p_i+p_k)\log(p_i+p_k)-p_i\log p_i)$$

$$= g(p_i+p_j)-g(p_i) \tag{61}$$

$$\geq 0 \tag{62}$$

Note that the last inequality is an application of strictly increasing property of the function

$$g(x)=(x+\xi)\log(x+\xi)-x\log(x). \tag{63}$$

Case 4: Let assume the addition of $e_{3,4}$ does not combine any cluster. Therefore we have

$$\mathcal{B}(A\cup\{e_{1,2}\})-\mathcal{B}(A)=\mathcal{B}(A\cup\{e_{1,2},e_{3,4}\})-\mathcal{B}(A\cup\{e_{3,4}\}). \tag{64}$$

From showing the diminishing return property for the four cases, we complete the proof. □

C. Proof of Theorem 3

The proof is given by showing that M satisfies the three matroid conditions.

C.1. Proof of the matroid condition C1

Proof. The set $\emptyset$ is an independent set of the matroid — it contains no cycles and constitutes a $|V|$-partition where $N_\emptyset = |V| > K$. □

C.2. Proof of the matroid condition C2

Proof. Let $I \in \mathcal{I}$ and $I' \subseteq I$. This implies that I has no cycles and $N_I \geq K$. Since $I' \subseteq I$, the set I' can be obtained by removing edges from the set I. The removal of edges does not add cycles and increases the number of connected components. Therefore the set I' also contains no cycles and $N_{I'} \geq K$. □

C.3. Proof of the matroid condition C3

Proof. Let I_1 and I_2 be two independent sets in $\mathcal{I}$ such that $|I_1| < |I_2|$. The independent set assumptions implies the associated connected components of I_1 and I_2 satisfy the following relations.

- $N_{I_1} = |V| - |I_1| \geq K$.
- $N_{I_2} = |V| - |I_2| \geq K$.
- $|I_1| < |I_2| \implies |V| - |I_1| > |V| - |I_2| \geq K$

From the above statements we have the following equations.

$$N_{I_1} > N_{I_2} \tag{65}$$

$$N_{I_1} = |V| - |I_1| \geq K+1 \tag{66}$$

We now prove that there exists some $e \in I_2 - I_1$ such that $N_{I_1\cup\{e\}} \geq K$ and the set $I_1 \cup \{e\}$ is cycle-free; i.e., $I_1 \cup \{e\}$ is an independent set. Since adding one edge to a graph decreases the number of connected components by at most one, Equation (66) implies $N_{I_1\cup\{e\}} \geq K$ for $e \in I_2 - I_1$. The remaining part of the proof is achieved by contradiction.

Let us assume there is no edge $e \in I_2 - I_1$ such that $I_1 \cup \{e\}$ is cycle-free. In other words, adding any edge e in $I_2 - I_1$ to the set I_1 will result in a cycle and leaves the number of connected components in the graph unchanged, $N_{I_1\cup\{e\}} = N_{I_1}$. Thus by adding all the edges from $I_2 - I_1$ to the set I_1, the number of connected components will remain as N_{I_1}.

$$N_{I_1\cup(I_2-I_1)} = N_{I_1\cup I_2} = N_{I_1}. \tag{67}$$

We know that the set $I_1 \cup I_2$ can also be obtained by adding edges from $I_1 - I_2$ to I_2. Since adding edges to a graph can only decrease the number of connected components, we have the following relation

$$N_{I_1\cup I_2} \leq N_{I_2} \implies N_{I_1} \leq N_{I_2}. \tag{68}$$

This contradicts Equation (65) and thus the theorem is proved.

□

References

[1] Y. Boykov, O. Veksler, and R. Zabih. Efficient approximate energy minimization via graph cut. *TPAMI*, 20(12):1222–1239, 2001. 2

[2] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *TPAMI*, 24(5):603–619, 2002. 1

[3] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley, 2 edition, 1991. 2, 3

[4] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004. 1, 7

[5] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of the approximations for maximizing submodular set functions - ii. *Mathematical Programming*, pages 73–87, 1978. 5

[6] C. Guestrin, A. Krause, and A. P. Singh. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, pages 235–284, 2008. 1

[7] D. Hoiem, A. N. Stein, A. A. Efros, and M. Hebert. Recovering occlusion boundaries from a single image. In *ICCV*, 2007. 1

[8] P. Kohli, L. Ladicky, and P. Torr. Robust higher order potentials for enforcing label consistency. *IJCV*, 82:302–324, 2009. 1, 7

[9] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *SIGKDD*, pages 420–429, 2007. 1, 5

[10] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, and S. J. Dickinson. Fast superpixels using geometric flows. *TPAMI*, 31(12):2290–2297, 2009. 2, 5, 7

[11] 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*, 2001. 5

[12] A. P. Moore, S. J. Prince, and J. Warrell. "lattice cut" – construcitng superpixles using layer constraints. In *CVPR*, 2010. 2

[13] A. P. Moore, S. J. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel lattices. In *CVPR*, 2008. 2

[14] G. Mori, X. Ren, A. A. Efros, and J. Malik. Recovering human body configurations: Combining segmentation and recognition. In *CVPR*, 2004. 1, 2

[15] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, pages 265–294, 1978. 4

[16] S. Nowozin, P. Gehler, and C. Lampert. On parameter learning in crf-based approaches to object class image segmentation. In *ECCV*, 2010. 5

[17] J. Oxley. *Matroid Theory*. Oxford University Press, 1992. 3

[18] S. Ramalingam, P. Kohli, K. Alahari, and P. H. S. Torr. Exact inference in multi-label crfs with higher order cliques. In *CVPR*, 2008. 1

[19] X. Ren and J. Malik. Learning a classification model for segmentation. In *ICCV*, 2003. 1, 2, 5, 7

[20] J. Shi and J. Malik. Normalized cuts and image segmentation. *TPAMI*, 22(8):888–905, 2000. 2

[21] O. Veksler and Y. Boykov. Superpixels and supervoxels in an energy optimization framework. In *ECCV*, 2010. 2, 5, 7

[22] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *TPAMI*, 13(6):583 –598, 1991. 1