Modularity Based Image Segmentation

Shijie Li and Dapeng Wu, *Fellow, IEEE*

***Abstract*—To address the problem of segmenting an image into sizeable homogeneous regions, this paper proposes an efficient agglomerative algorithm based on modularity optimization. Given an over-segmented image that consists of many small regions, our algorithm automatically merges those neighboring regions that produce the largest increase in modularity index. When the modularity of the segmented image is maximized, the algorithm stops merging and produces the final segmented image. To preserve the repetitive patterns in a homogeneous region, we propose a feature based on the histogram of states of image gradients, and use it together with the color feature to characterize the similarity of two regions. By constructing the similarity matrix in an adaptive manner, the over-segmentation problem can be effectively avoided. Our algorithm is tested on the publicly available Berkeley Segmentation Data Set as well as the Semantic Segmentation Data Set and compared with other popular algorithms. Experimental results have demonstrated that our algorithm produces sizable segmentation, preserves repetitive patterns with appealing time complexity, and achieves object-level segmentation to some extent.**

***Index Terms*—Image segmentation, community detection, modularity, clustering.**

I. Introduction

DUE to the wide use of digital cameras and smart phones, digital images are much more ubiquitous than they were several decades ago, resulting in the urgent need for the analysis of images, such as object recognition, image searching, indexing, and categorization. As a very important step for these high level image analysis tasks, image segmentation is a preprocessing process to group image pixels into some sizable homogeneous regions so that the complexity of further analysis can be substantially reduced. Image segmentation has received considerable attention since the problem was proposed, *e.g.*, [1]–[8], yet it still remains to be a challenging problem due to the following reasons: 1) image segmentation is an ill-defined problem and the optimal segmentation is user or application dependent; 2) image segmentation is time consuming in that each image includes a large number of pixels, especially for high resolution images, and this prevents image segmentation from being applied to real-time applications. In fact, Gestalt principles [9] and some cognition and psychological studies [10] have pointed out that several key factors affect perceptual grouping a lot, for example, proximity, similarity, regularity, *i.e.*, the repetitive patterns, relative size and etc. In this paper, we will take all these factors into consideration, and develop a computational efficient algorithm. Moreover, comprehensive evaluations of the segmentation performance under various metrics are also presented.

S. Li and D. Wu are with Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611. Correspondence author: Prof. Dapeng Wu, wu@ece.ufl.edu, http://www.wu.ece.ufl.edu. We would like to thank the editor and the anonymous referees for their careful comments and useful suggestions that significantly improve the quality of the paper.

A. Previous Work

In the literature, lots of image segmentation algorithms have been proposed. Here we only give a brief review of some of the popular algorithms.

The *Mean Shift* algorithm [3] treats image segmentation as a problem of clustering by detecting the modes of the probability density function in the feature space. Each pixel in the image is transformed to the joint spatial-range feature space by concatenating the pixel color value and its spatial coordinates into a single vector. Then the mean shift procedure is applied in this feature space to yield a convergence point for each pixel. All the pixels whose convergence points are closer than the spatial bandwidth h_s and the range bandwidth h_r are claimed to be in the same segment. In addition, minimum segment size is enforced to guarantee sizable segmentation. This method is usually fast. However, it is very sensitive to the bandwidth parameter h_r and h_s, and often results in over-segmentation. Since it is based on the density estimation of the color feature for all the pixels, some smooth changes in brightness and texture or the regularities of different colors will converge to different modes, though they belong to the same segment visually.

Another line of work view the segmentation problem from the perspective of graph partition. In this framework, the image is regarded as an undirected weighted graph, while each pixel is treated as a node in the graph and the edge weights measure the similarity or dissimilarity between nodes. Felzenszwalb and Huttenlocher (*F&H*) [4] consider each pixel as a single component in the initial stage, and arrange the edge weights of dissimilarity in non-decreasing order. For each iteration, the algorithm merges component C_1 and C_2 connected by the current edge if the corresponding edge weight is less than:

$$\min(\text{Int}(C_1) + \tau(C_1), \text{Int}(C_2) + \tau(C_2)), \tag{1}$$

where $\text{Int}(C)$ is the *internal difference* of component C, defined as the largest weight in the minimum spanning tree of component C; $\tau(C) = k/|C|$, and k is a constant parameter to control the minimize size of the segment. This algorithm gives nearly linear time complexity, however, it is very difficult to tune the parameter k for optimal segmentation.

Normalized Cut [2], as another popular graph partition based approach, incorporates the global information of the image into the segmentation process by studying the spectral characteristics of the graph. Given an affinity matrix W with each entry representing the similarity of two pixels, *Normalized Cut* tries to solve the generalized eigenvector problem:

$$(D - W)\mathbf{y} = \lambda D\mathbf{y}, \tag{2}$$

where D is a diagonal matrix with its diagonal entry $D_{ii} = \sum_j W_{ij}$. Then the segmentation is achieved by clustering

the eigenvectors. Due to the high computational cost, it can only deal with images of relatively small size. A variant is the *Multiscale Normalized Cut* approach [11], which allows to deal with larger images by compressing large images into multiple scales. However, it has the same problem with *Normalized Cut*, specifically, it 1) often breaks uniform or smooth regions where the eigenvectors have smooth gradients; 2) has a high time complexity; 3) needs a predefined number of segments, which itself is a challenging problem to deal with. More graph based segmentation can be found in the literature, *e.g.*, [12]–[14]. However, they all need human intervention, *e.g.*, they need a user to specify the number of regions resulting from image segmentation.

The *Watershed* segmentation method regards the gradient magnitude of an image as a topographic surface. The pixels where a water drop starts from would drain to the same local intensity minimum are in one segment, which is called *catchment basins*. The lines separating the catchment basins are the watersheds, namely, the boundaries. Various improved methods are available, *e.g.*, [15]–[18], but these methods are generally sensitive to noise and easily lead to over-segmentation.

The *Compression-based Texture Merging (CTM)* method [19] fits the image textures using the Gaussian Mixture Model, and employs the principle of *Minimum Description Length* to find the optimal segmentation, which gives the minimum coding length under a certain distortion ratio. Later, the *Texture and Boundary Encoding-based Segmentation* algorithm [6] improves the *CTM* by considering a hierarchy of multiple window sizes and offering more precise coding length computation. To be specific, it only encodes the texture information inside the non-overlapping windows in each region and also encodes the boundary with the adaptive chain code. However, this greatly increases the computational time cost; besides, the texture feature used in these two algorithms is essentially the pure color information from the cut-off windows and neglects the regularities inside the image, thus it may split the object with some regularities of different color.

Another broad class of methods address the image segmentation problem via solving the *Partial Differential Equations* (PDEs). Most of PDE based methods are carried out by the active contour model or snakes [20], where the basic idea is to evolve a curve (object boundary) such that an energy function is minimized. The energy function usually contains the internal energy as well as the external energy. The internal energy controls the smoothness of the curve, whereas the external energy guides the curve toward the true object boundary. Moreover, lots of researches have made improvements over this model (cf. [21]–[24]), however, generally, these methods are very sensitive to the noise, the model parameters and suffer from high computational cost.

B. Our Approach

Motivated by the limitations exposed in the existing work, our approach takes the following three aspects into consideration: 1) time complexity; 2) regularity preservation; 3) the prevention of over-segmentation. Inspired by the application of *community detection* algorithms in large scale networks, we attempt to view an image from the perspective of a network. For a network, *modularity* [25] is a crucial quantity, which is used to evaluate the performance of various community detection algorithms. In more detail, the larger the modularity of a network is, the more accurate the detected communities are. Considering the efficient calculation of modularity in the community detection algorithm, similarly, we regard image segmentation problem as a community detection problem, and the optimal segmentation is achieved when the modularity of the image is maximized.

Although modularity has been applied to image segmentation by some researchers recently, *e.g.*, [26], [27], it still faces similar problems as other segmentation algorithms mentioned above, due to the ignorance of the inherent properties of images (see Section II-B for more details). Different from the existing algorithms based on modularity, we identify the differences between community detection and image segmentation, start from 'superpixels', and propose a new texture feature from low level cues to capture the regularities for the visually coherent object and encode it into the similarity matrix; moreover, the similarity among regions of pixels is constructed in an adaptive manner so as to avoid over-segmentation. Compared with other existing segmentation algorithms, our proposed algorithm can automatically detect the number of regions/segments in an image, produces sizable regions with coherent regularities preserved, and achieves better semantic level segmentation to some extent. The contributions of this paper are the following:

- An efficient agglomerative segmentation algorithm incorporating the advantage of community detection and the inherent properties of images is developed. The algorithm enjoys low time complexity as well as comparable performance;
- A new texture feature, namely, *Histogram of States* (HoS) is proposed to capture the regularities in the image. The HoS feature, together with the color feature, encodes better similarity measure from the semantic level, and is more likely to preserve regularities in the object;
- An adaptive similarity matrix construction is proposed to avoid over-segmentation. In each iteration, the similarity between two regions of pixels is recalculated to reevaluate the color and texture similarity. In this way, it can effectively avoid breaking visually coherent regions, which share some regularities or have smooth changes in color or texture caused by shadow or perspectives.

The remainder of this paper is organized as follows. In Section II, we first briefly introduce the concepts of *modularity* and *community detection*, and discuss the existing methods based on modularity optimization in details. Section III first explains the relation between image segmentation and community detection, followed by the proposed overall algorithm and the detailed technical points. Section IV shows the experimental results on the publicly available Berkeley Segmentation Data Set (BSDS5000) [7] as well as the Semantic Segmentation Data Set (SSDS) [28]; both qualitative evaluation and

quantitative evaluation from three levels are provided. Finally, we conclude our paper in Section V.

II. Related Work

The concepts of *Modularity* and *Community Detection* become very popular in the field of network science during the past decade. Due to its application in large scale networks, recently, researches try to explore the possibilities of applying these concepts to image segmentation, where millions of pixels are included. In this section, we first review these two concepts, and then discuss two recent approaches based on *modularity* optimization in details.

A. Modularity and Community Detection

Modularity was first defined by *M.E.J. Newman* in [25] for the analysis of weighted networks. For a weighted network G with the weighted adjacency matrix A, the *modularity* Q is defined by:

$$Q = \frac{1}{2m}\sum_{i,j}[A_{i,j} - \frac{k_i k_j}{2m}]\delta(c_i, c_j), \tag{3}$$

where A_{ij} represents the weight between node i and node j; $m = \frac{1}{2}\sum_{ij} A_{ij}$ represents the total weights of the network; $k_i = \sum_j A_{ij}$ is the weighted degree of the node i; c_i is the community label to which node i belongs; $\delta(c_i, c_j)$ is 1 if node i and node j are in the same community, otherwise it's 0. Intuitively, modularity means to evaluate the difference between the actual probability of the connectivity of two nodes in the same community and the estimated probability under the assumption that the two nodes are connected randomly.

Community Detection becomes a hot topic in network science during the past few years, for example, social networks. A community is a group of nodes from the network, where nodes in the same community are densely connected with each other, and nodes in different communities are sparsely connected. Communities are of vital importance in a network, since they may represent some functional modules in the network. For example, a community in the social network may represent a group of friends sharing the same hobbies; a community in the citation network may reveal the related work in a certain research area. To uncover the interconnection of the nodes in a network, *Community Detection* algorithms aim to find a partition of the network such that every partition can well represent certain community property.

Since the first proposal of *modularity*, it has been widely used to evaluate the performance of *community detection* algorithms and also works as an optimization index for *community detection*. For example, *Louvain method* [29] is based on *modularity increase* to detect the communities. The *modularity increase* caused by merging community j into community i can be computed by Equation (4):

$$\begin{aligned} \Delta Q_{ij} &= [\frac{\sum_{in}+k_{j,in}}{2m} - (\frac{\sum_{tot}+k_j}{2m})^2] \\ &\quad -[\frac{\sum_{in}}{2m} - (\frac{\sum_{tot}}{2m})^2 - (\frac{k_j}{2m})^2] \\ &= \frac{1}{2m}(k_{j,in} - \frac{\sum_{tot} k_j}{m}), \end{aligned} \tag{4}$$

where $\sum_{in}$ in the total weights of the edges inside community i; $\sum_{tot}$ is the total weights of the edges incident to nodes in community i; $k_{j,in}$ is the sum of the weights from community j to community i; other notations are the same as defined in Equation (3).

As is shown in Figure 1, the basic idea of *Louvain method* is to iteratively repeat the process of *Modularity Optimization* in *Phase 1* and *Community Aggregation* in *Phase 2* below:

- *Phase 1:* Modularity Optimization
  At the beginning of this phase, the network is composed of several communities (each community is a single node initially, after several iterations, each community is a group of nodes), for each community i and its connected nodes $N_i = \{j|A_{ij} > 0\}$, compute the potential modularity increase ΔQ_{ij} if we merge community j ($\forall j \in N_i$) into community i, according to Equation (4). Find the maximum modularity increase caused by merging community j^* and community i and merge these two communities. Repeat this process until no modularity increase for all the communities in the network;
- *Phase 2:* Community Aggregation
  To reconstruct the network, merge the communities sharing the same label and relabel them; treat the communities with the same label as a single node in the network and recompute the weighted adjacency matrix by summing over all the weights connecting two communities.

The above processes are repeated until there is no modularity increase caused by merging any two communities.

B. Related Approaches for Image Segmentation

Recently, *modularity* optimization has been applied in image segmentation. [26] explores the possibility of directly applying *modularity* maximization to image segmentation, where a top-down spectral method with an iterative rounding scheme is proposed for fast computation. Such a scheme can reduce the computational cost to some extent, compared with the practically used *exchange heuristic* [30]. However, it can only deal with images of relatively small size on normal PCs, due to the involved manipulation of a dense modularity matrix. Besides, direct application of modularity maximization to image segmentation is known to result in serious over-segmentation.

To address the over-segmentation problems, [27] proposes to use a weighted modularity, where the modularity computation only occurs locally within a pre-defined distance. Moreover, an approximation of the Louvain method, the so-called *basic iteration*, is used for faster computation. However, the newly introduced distance parameter depends heavily on the images and the objects. For different images with different object sizes, the distance parameter is *ad-hoc*, and it is very difficult to choose a universal distance parameter.

Both of these two methods focus on how to apply modularity optimization to segmentation, and ignore the differences between community detection and image segmentation. Specifically, both methods start from single pixel, thus, the computational cost, though reduced to some extent by using different computational algorithms, is still too expensive, especially for the first one or two iterations. Furthermore,

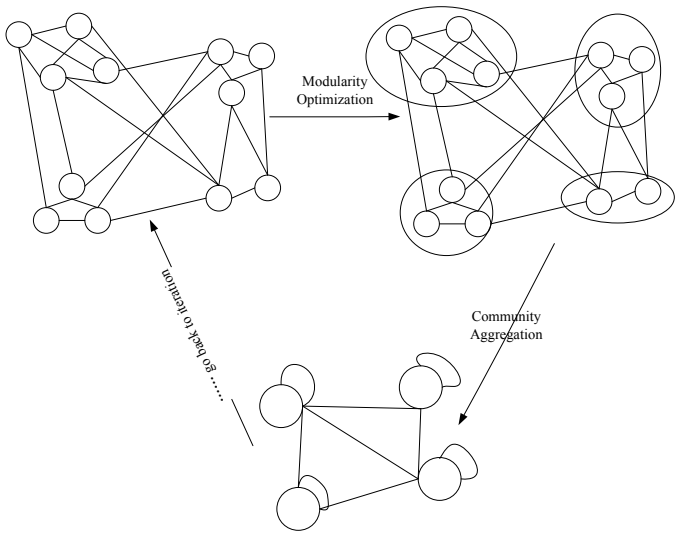


Fig. 1. **Illustration of the process for Louvain Method:** for each iteration, the algorithm first identifies the nodes which belong to the same community by optimizing the modularity; and then aggregate the community to construct a new network. This process stops only when no labeling changes.

since they only capture the color feature, they often break the regularities inside the object and leads to over-segmentation, even with properly chosen distance parameter in [27].

III. Algorithm Description

Image segmentation is related to community detection to some extent. Similar to nodes in the same community, the pixels inside the same segment also share some properties in common, like pixel color value. In this sense, we can treat each homogeneous image segment as a *community*, and think of image segmentation as a community detection problem.

However, due to the inherent properties of images, segmentation is not exactly a community detection problem and directly apply community detection algorithms to image segmentation will lead to awful performance. The differences between image segmentation and community detection can be revealed from the following aspects: 1) different from single node in a community, single pixel cannot capture these regularities in each visually homogeneous segment; 2) the pixels inside the same segment possibly have completely different properties, like color; while for communities, a community is a group of nodes share exactly similar properties. Take the face image as an example, the whole face should be treated as one segment for the purpose of image segmentation. In contrast, for community detection, the eye pixels would be treated as a separate community, while other parts of the face would be treated as another community due to the fact that the pixel color value property of the eyes is totally different from that of other parts of the face; 3) compared with communities, images share some a priori information, say, adjacent regions are more likely to belong to the same segment; 4) as the aggregation process goes on, more pixels are included in one region and the texture inside the region keeps updating, while the properties of the aggregated communities do not change much.

To address the above mentioned problems, we propose an efficient agglomerative image segmentation algorithm, taking advantage of the efficient calculation of the *modularity* optimization in community detection and the inherent properties of images. The algorithm starts from a set of over-segmented regions, thus, runs very fast, and produces sizable segmentation with the regularities inside the same object preserved. The overview of the proposed segmentation algorithm is summarized in Algorithm 1. And the detailed presentation of some technical points for our algorithm is as follows.

A. Superpixels

The agglomerative algorithm can start the aggregation process by treating each single pixel as a community, however, it turns out that this will be too much time consuming, especially for the first *Louvain* iteration. Fortunately, this is indeed not necessary, because no texture information is included for single pixel. Therefore, instead, we start with 'superpixels', which can reduce the computational cost as well as capture the regularities. Superpixels are a set of very small and homogeneous regions of pixels. Initializing with superpixels can greatly reduce the time complexity without affecting the segmentation performance. Hence, we first employ a pre-processing step to oversegment the image into a set of superpixels. This preprocessing step can be achieved by simple *K-Means* clustering algorithm (K is set to be a relative large value, *e.g.*, 200 or more) or other superpixels generating algorithms. In our implementation, we use a publicly available code [31] to get the superpixel initialization.

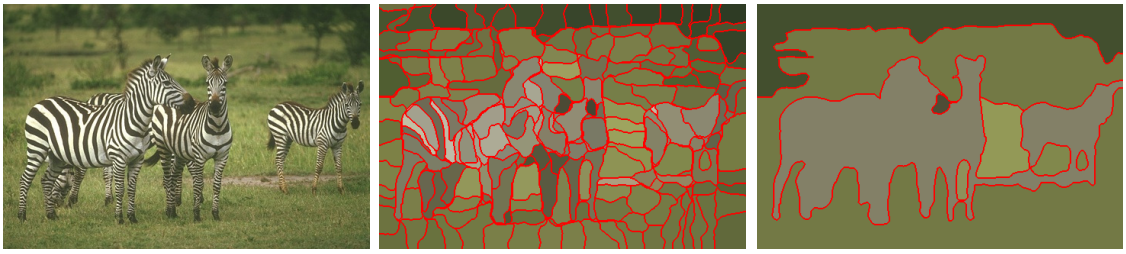
As is shown in the middle of Figure 2, the superpixel generation step usually gives more than 200 oversegmented regions on average. This step can greatly reduce the complexity to only consider about 200 nodes in the first iteration for our algorithm. The right column of Figure 2 shows the segmentation result given by our proposed algorithm, where only around 10 homogeneous regions with similar regular patterns inside are left. This fact demonstrates that the segmentation results are indeed the effects of our proposed algorithm rather than the superpixel generation algorithm.

Algorithm 1 Modularity based image segmentation

Input: Given a color image I and its oversegmented initialization with a set of superpixles $R = \{R_1, ..., R_n\}$

while Pixel labels still change **do**
  Reconstruct the neighborhood system for each region in R.
  Recompute the histogram of states texture feature and estimate the distribution of the color feature for each region.
  Adaptively update the similarity matrix W according to Equation (10), $W_{ij} \neq 0$ only if R_i and R_j are adjacent regions in I.
  while modularity increase still exists by merging any two adjacent regions **do**
    for each region $R_i \in R$ **do**
      Compute the modularity increase caused by merging region R_i with any of its neighboring regions according to Equation (4) and find the neighboring region R_j, which gives the largest modularity increase among all of the neighboring regions of R_i.
      Merge region R_i and region R_j by setting the labels of pixels in these two regions to be of the same label
    end for
  end while
  Update the region labels to get a new set of regions $R = \{R_1, ..., R_m\}$, where m is current number of regions;
end while

Output: The set of image segments R.



(a) Original image (b) Superpixel initialization (c) Our segmentation result

Fig. 2. **Comparison of superpixel and segmentation result.**

B. Choice of Color Space

To capture different aspects of the color, various color spaces are proposed in the literature [32], such as *RGB, L*a*b, YUV, HSV* and *XYZ*. To achieve good segmentation performance, the choice of color space is very important. Among all the color spaces, the *L*a*b* color space is known to be in accordance with human visual system and perceptually uniform, hence the image representation in this color space has been widely used in the field of image processing and computer vision. Due to this facet, all of our discussions of the algorithm are in the *L*a*b* color space. Later, in the experimental evaluation section, we have also validated that the segmentation performance in *L*a*b* color space is much better than that in the *RGB* color space.

C. Neighborhood System Construction

Different from normal networks, such as social networks or citation networks, images have self-contained spatial a priori information, *i.e.*, spatial coherent regions are more likely to be regarded as a single segment, while regions far away from each other are more likely to belong to different segments. Hence, different from *Louvain method* where two regions are considered to be neighbors as long as the similarity weight between them are nonzero, we have instead constructed a different neighborhood system by incorporating this spatial a prior information of images. To be specific, we only consider the possibility of merging neighboring regions in the image during each aggregation process. To achieve this, for each region in the image, we only consider the adjacent regions of this region to be its neighbors and store its neighboring regions using an *adjacent list*. The adjacent regions are defined to be the regions that share at least one pixel with the current region. In the following processes for the similarity matrix construction and aggregation, we only consider the current region and the regions in its neighborhood system.

D. Features for Similarity

Color is the most straightforward and important feature for segmentation, so we use the pixel value in the *L*a*b* color space as one of the features for computing the similarity. However, the color feature alone cannot achieve good segmentation performance, since it does not consider the repetitive patterns of different colors in some homogeneous object. For example, in the case of a *zebra* in Figure 2, the black and white stripe regularities on zebra would be treated as a whole part according to human's perception. Simply using color feature will break down these regularities into different segments. To address this problem, we not only employ the color feature, but we have also proposed a novel texture feature to capture the regularities in the image. Our proposed texture depicter is

motivated by the *Histogram of Oriented Gradients* (HoG) [33] for pedestrian detection, however, instead of constructing a histogram of gradients, we construct a *Histogram of States* (HoS) for each region following the three steps below.

1) Compute the gradient magnitude and 360 degree orientation information for each pixel in the image and then eliminate those magnitude without any texture information by thresholding the magnitude information;
2) At each pixel position, use a 5×5 sliding window[1], and form a histogram of the oriented gradient by dividing 360 degree orientation into 8 bins, then each pixel is represented by a 8 dimensional binary orientation vector with each dimension indicating whether this orientation exists in the current sliding window. In this sense, the texture information for each pixel belongs to one of the $2^8 = 256$ states;
3) For each region, we construct a 256 dimensional histogram of states vector, each dimension counts the number of such state in the segment, then normalize it by the total number of pixels in the segment.

An experimental comparison between segmentation with HoG and segmentation with HoS is shown in Figure 3. It is clear that with HoG used in encoding the similarity, the visually coherent pyramid together with the desert are broken. In contrast, HoS leads to better visual performance, owing to the robustness of HoS to some small noise (because HoS thresholds the gradient magnitude with small values and uses a 8-dimensional binary vector to indicate the existence of the corresponding orientation), and HoS's capability of better capturing the texture information (a sliding overlapping window, rather than a dense grid (for HoG), is used in HoS).

Remarks: HoS can be treated as a special case of the general Bag-of-Words model. For the general Bag-of-Words model, one important step is to construct the 'Words'. In our case, each of the 256 states is used as a 'Word'. For each region, we calculate the occurring frequency of each 'Word', and form a normalized histogram to represent the texture for this region.

E. Similarity Measure

We use different similarity measures for the two features. For the color feature, each pixel is represented by a three dimensional vector in the *L*a*b* color space. To measure the similarity between two regions of pixels, we assume that the pixel value in the same region follows a three dimensional Gaussian distribution, for example, pixels in region R_i and region R_j follow two Gaussian distributions, respectively, *i.e.*, $R_i \sim N(\mu_1, \Sigma_1)$ and $R_j \sim N(\mu_2, \Sigma_2)$. In the literature, lots of distance measures for distributions are studied, here we take a closer look at the following popular distance measures:

- Kullback-Leibler (KL) Divergence
  Kullback-Leibler (KL) Divergence is an information criteria to measure the divergence of two probabilistic density distributions $p(x)$ and $q(x)$. It is defined as:
  $$d_{KL} = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}. \tag{5}$$
  We can use this criteria to measure the distance of two regions of pixels. However, the drawback of this measure is that when $p(x)$ and $q(x)$ have different supports, then $q(x)$ can be 0 and the divergence measure becomes ill-posed.
- Earth Mover's Distance
  Earth Mover's Distance (EMD) [34] is a popular measure to describe the distance of two image distributions, and hence a very good metric to measure the distance of two regions of pixel value. It is proved that for two independent Gaussian distributions, EMD is equivalent to the Mallows distance [35] defined in Equation (6).
  $$\begin{aligned} &d_{EMD}(N(\mu_1, \Sigma_1), N(\mu_2, \Sigma_2))^2 \\ &= (\mu_1 - \mu_2)^T(\mu_1 - \mu_2) + \text{Tr}(\Sigma_1 + \Sigma_2 - 2(\Sigma_1\Sigma_2)^{\frac{1}{2}}). \end{aligned} \tag{6}$$
- Mean Distance
  Mean Distance (MD) is a trivial heuristic measure, however, it's a good approximation to the Earth Mover's distance. For MD, we simply use the mean of the pixel value for the two regions to approximate the Earth Mover's distance as in Equation (7), which neglects the difference of the covariance matrices, and hence is faster than computing the EMD.
  $$d_{MD}(N(\mu_1, \Sigma_1), N(\mu_2, \Sigma_2))^2 = (\mu_1 - \mu_2)^T(\mu_1 - \mu_2). \tag{7}$$

In our implementation, we can use both *Earth Mover's Distance (EMD)* and *Mean Distance (MD)* to compute the distance between the two color feature distributions for two regions of pixels. To transform the above distribution distance into similarity measure, we simply use a Gaussian type radius basis function in Equation (8):

$$W_{ij}(\text{color}) = \exp\{\frac{-\text{dist}(R_i, R_j)}{2\sigma^2}\}, \tag{8}$$

where $\text{dist}(R_i, R_j)$ measures the distance between the pixel value distributions for region R_i and R_j. In our experiments, we set $\sigma = 0.08$ and use the *Mean Distance*.

For our proposed *Histogram of States* (HoS) texture feature, each region is represented by a 256 dimensional vector, *i.e.*, for region R_i and R_j, the HoS feature vector $h_i, h_j \in R^{256}$. We use the cosine similarity measure to measure the similarity between the regions, as indicated in Equation (9).

$$W_{ij}(\text{texture}) = \cos(h_i, h_j) = \frac{h_i^T h_j}{\|h_i\| \cdot \|h_j\|}. \tag{9}$$

F. Adaptive Similarity Matrix Construction

In the traditional *Louvain method* for *community detection*, a consistent similarity matrix is used to update the new similarity matrix by simply summing over the weights of nodes in two different communities during each iteration. However, in our algorithm, we propose to construct the similarity matrix

[1]We test the window size with 5, 7 and 9 pixels, all these choices provide comparable satisfactory results qualitatively as well as quantitatively, thus we choose 5×5 sliding window for computational efficiency

(a) Original image (b) Segmentation with HoG (c) Segmentation with HoS

Fig. 3. **Comparison of segmentation results with HoG and HoS.**

adaptively. We maintain an adaptive similarity matrix during each iteration by recomputing the similarity between regions again according to Equation (8) and (9). The reason for this is because during the aggregation process, the region keeps expanding, and the similarity measure computed from the previous iteration might not suitable for current iteration. By maintaining an adaptive similarity matrix, we reevaluate the similarity between current regions and hence can effectively overcome the problem of splitting the non-uniformly distributed color or texture, which should be grouped into the same segment from the perspective of human vision system. In this way, over-segmentation is avoided.

Figure 4 shows the segmentation results based on the consistent similarity matrix used in the community detection and our proposed adaptive similarity matrix, respectively (for all the segmentation result figures presented in this paper, red contours outline the boundary of different segments and each segment is shown by its mean color). It can be seen that with consistent similarity matrix, the images are segmented into homogeneous regions, but it tends to be oversegmented. This validates our analysis in the previous section. On the contrary, our proposed adaptive similarity matrix based segmentation gives much better segmentation results, visually.

During each iteration, we use a hybrid model to empirically combine the color feature and the HoS texture feature as below:

$$W_{ij} = a \times \sqrt{W_{ij}(\text{texture}) \times W_{ij}(\text{color})} + (1-a) \times W_{ij}(\text{color}), \tag{10}$$

where a is a balancing parameter. As Figure 5 shows, a given image can have multiple segmentation results with respect to different choices of a. When the texture information is not taken into consideration, *i.e.*, $a = 0$, the stripes on the zebra are broken into different segments. With the increasing value of a, more such stripe patterns are encoded into the similarity, thus better preserves the regularities. However, if a is too large, the trees and the grass in the image are merged into one segment. Considering the importance of color feature, we therefore give higher priority to the color feature.

Remarks: Recall that in [4], adaptive weighting scheme is also used through (1), where the adaptive threshold is decided by the largest edge weights in the minimum spanning tree of the two components C_1 and C_2. However, the adaptiveness here is caused by the aggregation process, and all the edge weights are only computed once at the beginning of the construction of the graph. In contrast, in our scheme, the similarity (edge weight) is reevaluated during each iteration, that is, the texture similarity and the color similarity are recomputed. This is very important because as the aggregation process goes on, the regularities inside one region keeps changing, and simply using the edge weights constructed at the very beginning cannot capture these changes. Hence, our adaptive similarity matrix construction scheme effectively helps preserve the regularities and avoids over-segmentation.

IV. Experimental Evaluation

In this section, extensive experiments have been done to evaluate the segmentation performance of our proposed algorithm, qualitatively as well as quantitatively. Moreover, we also discuss the time complexity of several different algorithms. The algorithms are tested on two datasets: 1) one dataset is the publicly available Berkeley Segmentation Data Set 500 (BSDS500) [7]. BSDS500 is comprised of 500 images, including 200 images for training, 200 images for testing and 100 images for validation. BSDS500 also provides ground-truth segmentations manually generated by several human subjects. For each image, 5 to 8 ground-truth segmentation maps are provided; 2) the other dataset is the Semantic Segmentation Data Set (SSDS) [28], which includes 100 images selected from BSDS500, and also contains the semantic level ground-truths that are generated by using the existing ground-truths of BSDS500 as well as an interactive segmentation tool. Figure 6 shows some sample images from BSDS500 and the corresponding ground-truth segmentations provided by BSDS500 and SSDS. It can be seen that some ground-truth segmentations provided by BSDS500 is of fine granularity, while SSDS gives better semantic level ground-truth segmentations instead.

A. Qualitative Evaluation

For qualitative evaluations, we present some figures of the segmentation results. We categorize all images in BSDS500 into four classes, namely, *people*, *animals*, *urban scenery* and *natural scenery*. Figure 7 presents some segmentation results of our proposed modularity based image segmentation algorithm on some randomly chosen images from the four different classes in BSDS500. From these qualitative results, we can see that the proposed algorithm produces sizeable segments for all the selected images; besides, the human

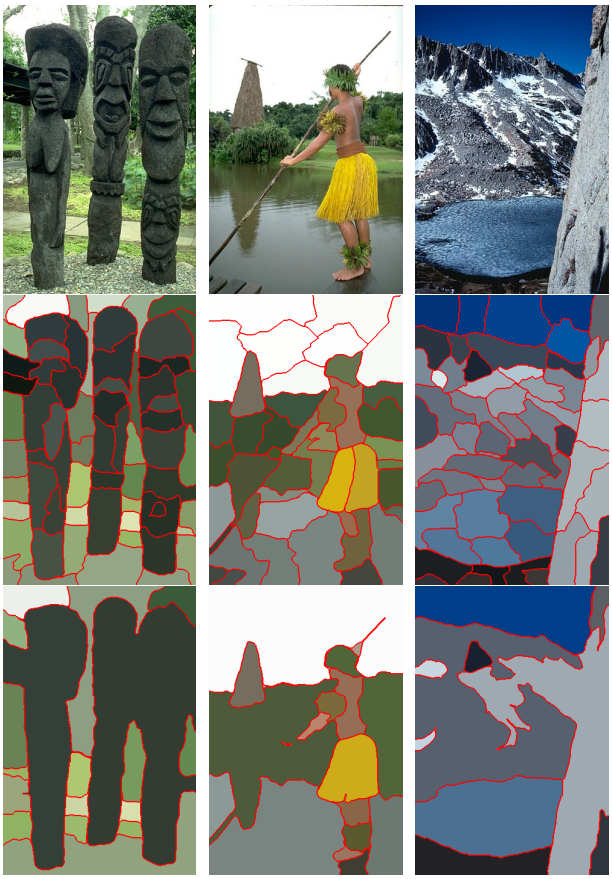
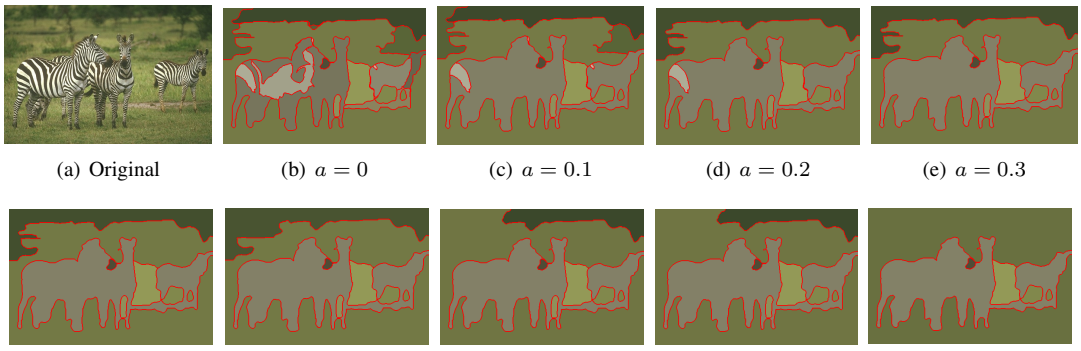


Fig. 4. **Segmentation results based on different similarity matrices.** *Top*: Original images. *Middle*: Segmentation results with consistent similarity matrix. *Bottom*: Segmentation results with adaptive similarity matrix.



(a) Original (b) $a = 0$ (c) $a = 0.1$ (d) $a = 0.2$ (e) $a = 0.3$

(f) $a = 0.4$ (g) $a = 0.5$ (h) $a = 0.6$ (i) $a = 0.7$ (j) $a = 0.8$

Fig. 5. **Our segmentation results for varying choices of a.**

face, animals, castle and mountains are all segmented into an integrate part. Even if some pixels have very different values inside the same segment, the similarity matrix encoding of the HoS texture feature successfully preserves the regularities and classifies those pixels into the same segment. In this sense, our proposed algorithm achieves object-level segmentation to some extent.

We also qualitatively compare our algorithm with other popular segmentation algorithms[2], including *Weighted Modularity Segmentation (WMS)* [27], *Mean-Shift (MS)* [3], *Multiscale Normalized Cut (MNC)* [11], *F&H* [4], *Marker Controlled Watershed* (MCW), *Compression based Texture Merging (CTM)* [19] and *Texture and Boundary Encoding-based Segmentation (TBES)* [6]. The segmentation results are presented in Figure 8. As can be seen, *MS*, *MNC*, *F&H* and *MCW* all show different extent of over-segmentation. In contrast,

[2]We have not compared with [26], because it takes too much memory, and cannot deal with 481×321 images in the BSDS500 and SSDS dataset on normal PC.

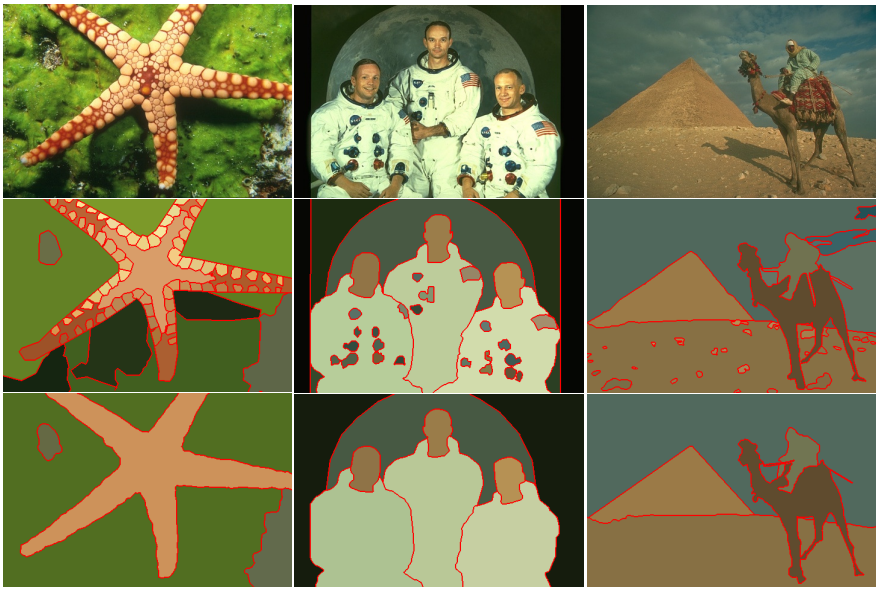
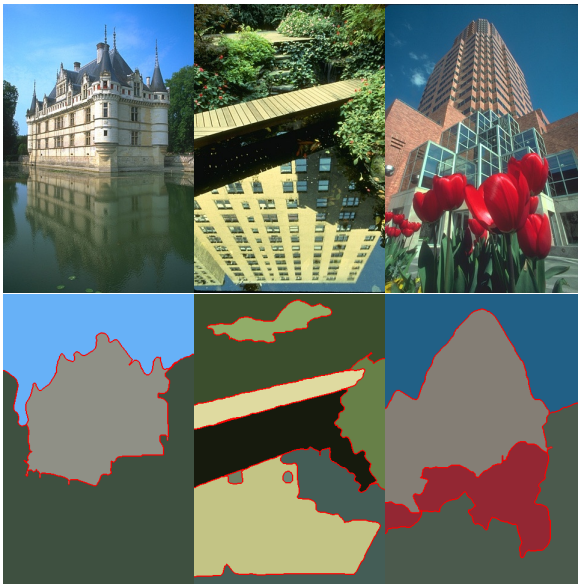
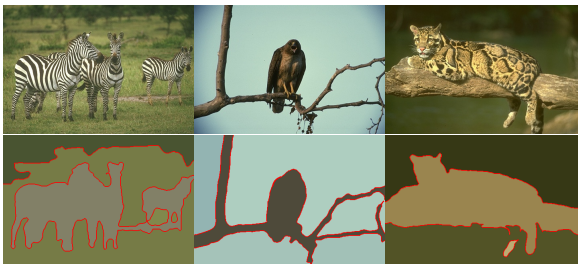


Fig. 6. **Sample images.** *Top*: Original images. *Middle*: Ground-truth segmentations from BSDS500. *Bottom*: Ground-truth segmentations from SSDS.

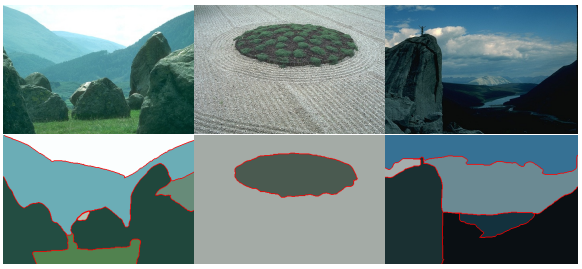
(a) People



(b) Urban Scenery



(c) Animals



(d) Natural Scenery

Fig. 7. **Modularity based segmentation results for different categories in BSDS500.** For each category, *Top:* Original images. *Bottom:* Segmentation results generated by our methods.

this issue is mitigated for other algorithms. Nevertheless, we observe that *WMS*, *CTM* and *TBES* break the regularities in

some homogeneous regions, such as the shirt and human face. Thanks to the HoS texture feature, our proposed algorithm well preserves these regularities.

B. Quantitative Evaluation

Qualitative evaluation is too subjective and the evaluations by different people vary a lot. In this paper, we quantitatively evaluate the segmentation performance from three levels, namely, *region level*, *boundary level* and *semantic level*, and the segmentation results are compared with seven other popular methods mentioned in Section IV-A. In the WMS implementation, we use the intervening contours method to construct the affinity matrix, and set the distance parameter to be $d_\Lambda = 20$. The MS is run with the same parameter setting used in [3] for large images, say, $(h_s, h_r) = (16, 7)$. For the MNC, we choose the number of segments to be $K = 20$, which is the average number of segments from the ground-truths. According to [4], for *F&H*, the Gaussian smoothing parameter is set to be $\sigma = 0.8$ In the CTM, we use the distortion rate $\varepsilon = 0.2$. The quantization level in TBES is set to be $\varepsilon = 150$ to be consistent with [6].

1) Region Level: We employ two commonly used region-based metrics for evaluating two pairs of segmentations, *i.e.*, the *Probabilistic Rand Index* (PRI) [36] and the *Variation of Information* (VOI) [37], which are described as below:

- The *Probabilistic Rand Index* (PRI) is a classical evaluation criteria for clusterings. PRI measures the probability that the pair of samples have consistent labels in the two segmentations. The range of PRI is [0,1], with larger value indicating greater similarity between two segmentations;
- The *Variation of Information* (VOI) measures how much we can know of one segmentation given another segmentation. VOI is defined by:

$$VOI(C, C') = H(C) + H(C') - 2I(C, C'), \quad (11)$$

where $H(C)$ and $H(C')$ are the entropy of segmentation C and C', respectively and $I(C, C')$ is the mutual information of two segmentations C and C'. The range of this metric is $[0, +\infty)$, and the smaller the value is, the more similar the two segmentations are.

We run the algorithms on the 100 validation images from BSDS500 once so as to make the result consistent with the BSDS300 [38] (previous version of the dataset). Since the ground-truth in BSDS500 has multiple segmentation maps, typically, 5 segmentation maps per image, we simply use the mean value of the metric calculated between the segmentation result and all the ground-truths for each image. To determine the balancing parameter a, we run our algorithm on each of the 100 images for varying a from 0.05 to 0.5 with 0.05 interval. It turns out that $a = 0.25$ strikes the best balance between the PRI and VOI metrics, thus we use 0.25 across all the following experiments. For each algorithm, we use the same parameter settings to run across all the images. The mean values of *PRI* and *VOI* are reported in Table I.

TABLE I
REGION LEVEL: QUANTITATIVE COMPARISON OF DIFFERENT ALGORITHMS ON BSDS

Algorithms	PRI (larger better)	VOI (smaller better)
Human	0.87	1.16
Our's (*L*a*b*)	0.777	1.879
Our's (*RGB*)	0.749	2.149
WMS	0.752	2.103
MS	0.772	2.004
MNC	0.742	2.651
F&H	0.770	2.188
MCW	0.753	2.203
CTM	0.735	1.978
TBES	0.785	2.002

In Table I, the third and forth rows show the performance of the proposed algorithm in different color spaces. Again, it has been validated that the algorithm works better in the *L*a*b* color space than in the *RGB* color space in terms of both *PRI* and *VOI*. Therefore, we run our algorithm in the *L*a*b* color space for all the following experiments. Also, in terms of *VOI*, the proposed algorithm achieves the best performance among all the popular segmentation algorithms; in terms of *PRI*, our algorithm has a close performance with *TBES*, and outperforms all the rest algorithms.

2) Boundary Level: We also evaluate the performance using the standard boundary-based methodology developed in [39]. This framework first gets the optimal boundary matching between the testing segmentation and the ground-truth, and then evaluates the results from two aspects: *Precision* and *Recall*. Given the testing segmentation C_{test} and the ground-truth segmentation C_{gt}, the *Precision* measures the fraction of detected boundary pixels that match the ground-truth boundaries, and is defined as:

$$\text{Precision} = \frac{|C_{test}| \bigcap |C_{gt}|}{|C_{test}|}, \quad (12)$$

where $|C|$ means the number of boundary pixels in the segmentation C. Similarly, the *Recall* is defined as:

$$\text{Recall} = \frac{|C_{test}| \bigcap |C_{gt}|}{|C_{gt}|}, \quad (13)$$

which measures the fraction of ground-truth boundary pixels that are detected. To summarize these two indices, the global F_α*-measure*, defined in Equation (14), is used to measure the harmonic mean of the *Precision* and *Recall*. We set $\alpha = 0.5$ and use it for all the experiments.

$$F_\alpha = \frac{\text{Precision} \cdot \text{Recall}}{(1-\alpha) \cdot \text{Recall} + \alpha \cdot \text{Precision}}. \quad (14)$$

Table II lists the *Precision* and *Recall* values for different algorithms under the same settings as the region level evaluation. It can be seen that our algorithm obtains the highest precision with a value of 0.733, indicating that most of our generated boundaries match the ground-truth segmentation manually generated by human subjects. However, the result reports the lowest recall value for our algorithms compared with other methods, which leads to a relatively lower $F_{0.5}$-*measure* compared to *CTM* and *TBES*. The reason for this is probably because the *Recall* is sensitive to under-segmentation.

(a) Original (b) Our's (c) WMS (d) MS (e) MNC

(f) F&H (g) MCW (h) CTM (i) TBES

Fig. 8. **Qualitative comparison of segmentation results by some popular methods.**

In our algorithm, we encode the HoS feature into the similarity matrix to preserve regularities, and update the similarity matrix in an adaptive fashion to produce sizable segmentation, which leads to some extent of under-segmentation. On the contrary, other methods exhibit different degrees of over-segmentation, hence obtain larger *Recall* values.

TABLE II
BOUNDARY LEVEL: QUANTITATIVE COMPARISON OF DIFFERENT ALGORITHMS ON BSDS

Algorithms	Precision	Recall	$F_{0.5}$-measure
Our's	0.733	0.508	0.600
WMS	0.491	0.714	0.580
MS	0.431	0.750	0.548
MNC	0.575	0.602	0.588
F&H	0.442	0.762	0.560
MCW	0.480	0.726	0.578
CTM	0.702	0.532	0.607
TBES	0.730	0.522	0.609

3) Semantic Level: As is pointed out in [8], the ground-truth segmentations provided by BSDS500 are too granular and designed for boundary detection and general segmentation. Therefore, we perform the semantic level evaluation on the SSDS proposed in [28]. In addition to the semantic level ground-truths, [28] also provides an easy-to-compute *Precision* and *Recall* boundary evaluation metric to replace the corresponding one based on optimal boundary matching in [39]. We run the algorithm once for the 100 images in SSDS, and use the evaluation software provided by [28] to compute the *Precision*, *Recall* and the combined $F_{0.5}$*-measure*. The results are listed in Table III as below:

TABLE III
SEMANTIC LEVEL: QUANTITATIVE COMPARISON OF DIFFERENT ALGORITHMS ON SSDS

Algorithms	Precision	Recall	$F_{0.5}$-measure
Our's	0.408	0.500	0.435
WMS	0.272	0.302	0.286
MS	0.168	0.399	0.226
MNC	0.312	0.264	0.277
F&H	0.235	0.331	0.271
MCW	0.228	0.342	0.274
CTM	0.387	0.458	0.407
TBES	0.404	0.492	0.430

As is shown in Table III, our algorithm achieves the best performance in terms of all three indicies compared with other algorithms. We have a relatively higher *Recall* value on this dataset. This is reasonable since the ground-truths are object level, which caters to our algorithm. This result demonstrates that our proposed algorithm achieves better semantic level segmentation than others. We have also noted that *TBES* obtains very close performance with the proposed algorithm.

C. Time Complexity

Considering the large amount of pixels to deal with for images, lower time complexity without impacting the performance much is always preferred, especially in the situation where real time application is needed. We compare the run time of our proposed algorithm with *CTM* and *TBES*, since these three algorithms start with superpixels. Given the same superpixel initializations. we run the algorithms over the 100 validation images from BSDS500 once, and then compute the mean time and the lower/upper bound of the 95% confidence interval for the run time, which are listed in Table IV. All

the algorithms are implemented in Matlab and run on the 2.4 GHz Intel processor with 4 *GB* of RAM. It can be seen that our algorithm runs consistently faster than *CTM* and *TBES*, specifically, about 3 times faster than *CTM* and more than 40 times faster than *TBES* on average. Besides, from the lower and upper bound, we can see that the variation of the run time for *CTM* and *TBES* is much larger than our proposed algorithm. In the experiment, we have also observed that, on average, it takes about 5 to 6 iterations before the algorithm converges and as the aggregation process goes on, less time is needed for each iteration, since there are fewer regions to be aggregated after each iteration.

TABLE IV
95% CONFIDENCE INTERVAL ANALYSIS FOR TIME COMPLEXITY (IN UNIT OF SECONDS)

Algorithms	Lower Bound	Mean	Upper Bound
Our's	18.81	20.40	22.00
CTM	25.63	67.08	108.53
TBES	586.74	842.86	1062.97

V. CONCLUSION

In this paper, we have proposed an efficient image segmentation algorithm taking advantages of the scalability of modularity optimization and the inherent properties of images. Adopting the bottom-up framework, the proposed algorithm automatically detects the number of segments in the image, and by employing the color feature as well as the proposed *Histogram of States* (HoS) texture feature, it adaptively constructs the similarity matrix among different regions, optimizes the modularity and aggregates the neighboring regions iteratively. The optimal segmentation is achieved when no modularity increase occurs by aggregating any neighboring regions. Results of extensive experiments have validated that the proposed algorithm gives impressive qualitative segmentation results; besides, it is reported that the new algorithm achieves the best performance among all the experimented popular methods in terms of *VOI* and *Precision* on BSDS500. Since the algorithm aims to avoid over-segmentation, it produces low *Recall* value. In addition, it is demonstrated that the new algorithm can preserve regularities in the object and achieve the best performance from the semantic level on SSDS. What's more, our proposed algorithm provides appealing time complexity and runs consistently faster than *CTM* and *TBES* under the same experiment settings.

REFERENCES

[1] B. Bhanu and J. Peng, "Adaptive integrated image segmentation and object recognition," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 30, no. 4, pp. 427–441, 2000.

[2] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[3] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1271–1283, 2010.

[4] P. Felzenszwalb and D. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[5] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, "From contours to regions: An empirical evaluation," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. IEEE, 2009, pp. 2294–2301.

[6] S. Rao, H. Mobahi, A. Yang, S. Sastry, and Y. Ma, "Natural image segmentation with adaptive texture and boundary encoding," *Asian Conference on Computer Vision, ACCV.*, pp. 135–146, 2010.

[7] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2011.

[8] H. Zhu, J. Zheng, J. Cai, and N. M. Thalmann, "Object-level image segmentation using low level cues," *IEEE Transactions on Image Processing*, 2013.

[9] M. Wertheimer, "Laws of organization in perceptual forms," *A Source Book of Gestalt Psychology*, pp. 71–88, 1938.

[10] D. D. Hoffman and M. Singh, "Salience of visual parts," *Cognition*, vol. 63, no. 1, pp. 29–78, 1997.

[11] T. Cour, F. Benezit, and J. Shi, "Spectral segmentation with multiscale graph decomposition," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR.*, vol. 2. IEEE, 2005, pp. 1124–1131.

[12] L. Grady and E. L. Schwartz, "Isoperimetric graph partitioning for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 3, pp. 469–475, 2006.

[13] J. Wang, Y. Jia, X.-S. Hua, C. Zhang, and L. Quan, "Normalized tree partitioning for image segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008*. IEEE, 2008, pp. 1–8.

[14] C. Couprie, L. Grady, L. Najman, and H. Talbot, "Power watershed: A unifying graph-based optimization framework," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 7, pp. 1384–1399, 2011.

[15] L. Vincent and P. Soille, "Watersheds in digital spaces: an efficient algorithm based on immersion simulations," *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 583–598, 1991.

[16] V. Grau, A. Mewes, M. Alcaniz, R. Kikinis, and S. K. Warfield, "Improved watershed transform for medical image segmentation using prior information," *IEEE Transactions on Medical Imaging*, vol. 23, no. 4, pp. 447–458, 2004.

[17] X.-C. Tai, E. Hodneland, J. Weickert, N. V. Bukoreshtliev, A. Lundervold, and H.-H. Gerdes, "Level set methods for watershed image segmentation," in *Scale Space and Variational Methods in Computer Vision*. Springer, 2007, pp. 178–190.

[18] V. Osma-Ruiz, J. I. Godino-Llorente, N. Sáenz-Lechón, and P. Gómez-Vilda, "An improved watershed algorithm based on efficient computation of shortest paths," *Pattern Recognition*, vol. 40, no. 3, pp. 1078–1090, 2007.

[19] A. Yang, J. Wright, Y. Ma, and S. Sastry, "Unsupervised segmentation of natural images via lossy data compression," *Computer Vision and Image Understanding*, vol. 110, no. 2, pp. 212–225, 2008.

[20] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[21] C. Xu and J. L. Prince, "Snakes, shapes, and gradient vector flow," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 359–369, 1998.

[22] ——, "Generalized gradient vector flow external forces for active contours," *Signal processing*, vol. 71, no. 2, pp. 131–139, 1998.

[23] T. F. Chan and L. A. Vese, "Active contours without edges," *IEEE Transactions on Image Processing*, vol. 10, no. 2, pp. 266–277, 2001.

[24] B. Li and S. T. Acton, "Active contour external force using vector field convolution for image segmentation," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2096–2106, 2007.

[25] M. Newman, "Analysis of weighted networks," *Physical Review E*, vol. 70, no. 5, p. 056131, 2004.

[26] W. Li, "Modularity segmentation," in *Neural Information Processing*. Springer, 2013, pp. 100–107.

[27] A. Browet, P.-A. Absil, and P. Van Dooren, "Community detection for hierarchical image segmentation," in *Combinatorial Image Analysis*. Springer, 2011, pp. 358–371.

[28] H. Li, J. Cai, T. N. A. Nguyen, and J. Zheng, "A benchmark for semantic image segmentation," in *IEEE International Conference on Multimedia & Expo, ICME.*, 2013.

[29] V. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, p. P10008, 2008.

[30] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations research*, vol. 21, no. 2, pp. 498–516, 1973.

[31] G. Mori, "Guiding model search using segmentation," in *The Proceedings of the 10th IEEE International Conference on Computer Vision, ICCV.*, vol. 2. IEEE, 2005, pp. 1417–1423.
[32] K. Plataniotis and A. Venetsanopoulos, *Color image processing and applications*. Springer, 2000.
[33] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR.*, vol. 1. IEEE, 2005, pp. 886–893.
[34] J. Puzicha, J. Buhmann, Y. Rubner, and C. Tomasi, "Empirical evaluation of dissimilarity measures for color and texture," in *The Proceedings of the 7th IEEE International Conference on Computer Vision, ICCV.*, vol. 2. IEEE, 1999, pp. 1165–1172.
[35] D. Dowson and B. Landau, "The frechet distance between multivariate normal distributions," *Journal of Multivariate Analysis*, vol. 12, no. 3, pp. 450–455, 1982.
[36] C. Pantofaru and M. Hebert, "A comparison of image segmentation algorithms," *Robotics Institute*, p. 336, 2005.
[37] M. Meil, "Comparing clusterings: an axiomatic view," in *The Proceedings of the 22nd International Conference on Machine Learning*. ACM, 2005, pp. 577–584.
[38] 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 *The Proceedings of the 8th IEEE International Conference on Computer Vision, ICCV.*, vol. 2, July 2001, pp. 416–423.
[39] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530–549, 2004.