# Extract and Merge: Superpixel Segmentation with Regional Attributes

Jianqiao An[1,2], Yucheng Shi[1,2], Yahong Han[1,2,3[0000−0003−2768−1398]†], Meijun Sun[1[0000−0002−8691−8677]], and Qi Tian[4[0000−0002−7252−5047]]

[1] College of Intelligence and Computing, Tianjin University, Tianjin, China
[2] Tianjin Key Lab of Machine Learning, Tianjin University, Tianjin, China
[3] Peng Cheng Laboratory, Shenzhen, China
{anjianqiao, yucheng, yahong, sunmeijun}@tju.edu.cn
[4] Noah's Ark Lab, Huawei Technologies
tian.qi1@huawei.com

**Abstract.** For a certain object in an image, the relationship between its central region and the peripheral region is not well utilized in existing superpixel segmentation methods. In this work, we propose the concept of **regional attribute**, which indicates the location of a certain region in the object. Based on the regional attributes, we propose a novel superpixel method called **Extract and Merge (EAM)**. In the extracting stage, we design square windows with a side length of a power of two, named **power-window**, to extract regional attributes by calculating boundary clearness of objects in the window. The larger windows are for the central regions and the smaller ones correspond to the peripheral regions. In the merging stage, power-windows are merged according to the defined attraction between them. Specifically, we build a graph model and propose an efficient method to make the large windows merge the small ones strategically, regarding power-windows as vertices and the adjacencies between them as edges. We demonstrate that our superpixels have fine boundaries and are superior to the respective state-of-the-art algorithms on multiple benchmarks.

**Keywords:** superpixel, power-window, boundary clearness, graph model

## 1 Introduction

Superpixel segmentation is to divide an image into several fragments without intersecting, as illustrated in Fig. 1. The major advantage of using superpixels is computational efficiency. It can be a helpful pre-process of numerous computer-vision tasks such as image semantic segmentation [12, 5], object detection [14, 17], salient object detection [18, 21], tracking [19] etc. Some work in the field of deep learning that specifically utilizes the features of the superpixels has also been proposed in recent years. SPN [8] employs superpixel segmentation as a pooling layout to reflect low-level image structure for learning and inferring

†Corresponding author

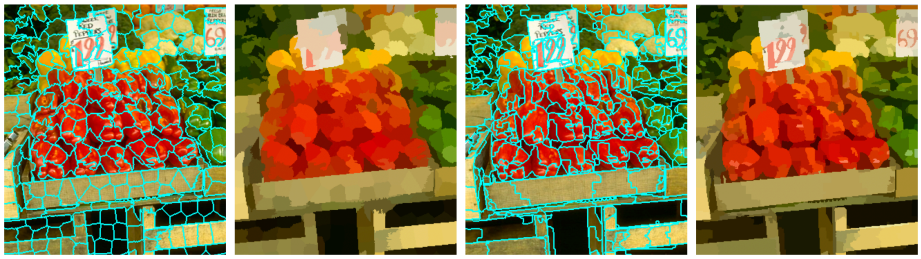

**Fig. 1.** The images in the first and third columns show superpixel boundaries and the ones in the second and fourth columns show superpixels colored by their mean colors. From left to right, the results are generated by SLIC [1] and the proposed EAM, respectively, with both 300 superpixels.

semantic segmentation. However, this requires the superpixel boundaries generated by algorithms can be as close as possible to the object boundary of the original image under various complicated conditions, so that the semantic information expressed by superpixels is almost equivalent to that expressed by the original image.

In light of fundamental importance of superpixels in the computer vision, various representative superpixel segmentation algorithms have been proposed and widely applied, including unsupervised methods [1, 20, 13, 2] and deep learning based methods [16, 6] etc. Deep learning based methods relies on the training set, which is inefficient and lacks generalization ability. Here we list some disadvantages of existing superpixel methods:

1) Pixels are directly merged into superpixels which are not well resistant to noise and complex textures in the images.
2) Most of the algorithms focus on the color and spatial information but ignore the deeper relationship between image pixels.
3) The superpixels cannot be flexibly allocated according to the regional features, thus complicated details cannot be well segmented (see Fig. 1).

Innovatively, we divide the image region of an object into several categories, from the central region to the peripheral region. The concept of **regional attribute** is proposed to describe the labels for each category. Instead of directly merging independent original image pixels, we extract regional attributes of each image region through a proposed structure called Power-Window and merge them into superpixels. Superpixels generated in this way are more disciplined and more consistent with human vision. The contributions of this paper are summarized as follows:

1) **Extract and Merge (EAM)**, a novel method focusing on **regional attributes** of objects in an image is proposed.
2) An ingenious structure called **power-window** is proposed to classify pixels by regional attribute. Moreover, the definition and solution of **boundary clearness** is proposed to help accomplish this extract regional attributes.

3) A graph model is built on power-windows and the solution called **attraction competition** is designed to merge windows into flexible and diverse superpixels.
4) Experimental comparisons with several state-of-the-art superpixel segmentation methods are taken on the BSDS500 dataset. The results show that EAM keeps almost all the details of the original images and performs favorably against the state-of-the-art.

## 2 Related Work

The superpixel segmentation methods can be roughly divided into two categories: unsupervised methods and deep learning methods. Unsupervised methods include graph-based approach and clustering-based approach.

**Graph-based methods:** Prominent graph-based approaches for image segmenation rely on pixel graphs. Superpixels are generated by minimizing a cost function using a graph model, in which pixels are vertices and pixel-level similarities are treated as edge weights. The normalized cuts (NC) algorithm [7] is one of the representative works of graph-based methods, which creates superpixels by recursively computing normalized cuts for the pixel graph. However, NC suffers from the high computational complexity cost. Felzenszwalb and Huttenlocher (FH) [4] propose a minimum spanning tree based segmentation approach. Their algorithm progressively joins components until a stopping criterion is met, which prevents the spannning tree from covering the whole image. Although the FH method preserves boundaries well, it often generates both extremely large and small segments. The ERS [10] algorithm presents a superpixel segmentation method by maximizing an objective function of entropy rate on graph topology, which generates homogeneous superpixels and adhere to the boundaries. A greedy algorithm is used to obtain solutions. Lazy random walk (LRW) [13] adheres to object boundary well and preserves texture regions by translating input image into a graph. The graph vertex is the image pixel, and then superpixels are initialized and optimized iteratively by an energy function. In general, separate image pixels are treated as vertices in these graph-based methods, causing them susceptible to noise interference.

**Clustering-based methods:** Series of clustering-based superpixel methods are developed based on clustering techniques. SLIC [1] produces superpixels by adopting k-means clustering approach in the five dimensional CIELab color and position feature space to cluster pixels. The LSC [20] method projects the five dimensional features to a ten dimensional space and performs clustering in the projected space. Normalized cuts formulation is adopted in LSC to generate the final superpixels. The SEEDS [2] algorithm uses uniform blocks as initial approximation of superpixels and iteratively exchanges neighboring blocks in a coarse-to-fine manner based on an objective function. However, these clustering-based methods are still in units of a single pixel, affecting the quality of the superpixels.

**Deep learning methods:** Recently, deep learning based methods for superpixel segmentation have been proposed. Two representative deep learning

based superpixel segmentation methods are SEAL [16] and SSN [6]. In SEAL, a new loss function is proposed to take the segmentation error into account for affinity learning and Pixel Affinity Net is designed for affinity prediction. SSN presents a differentiable superpixel sampling model which can be integrated into end-to-end trainable networks.

Individual pixels are the basic unit in all of these algorithms, which cause the algorithms to be insensitive to semantic information. Pre-extracting the central and peripheral regions, and then merging these regions (instead of independent pixels) into superpixels is demonstrated by us to be a smarter way.

## 3 The Proposed Method

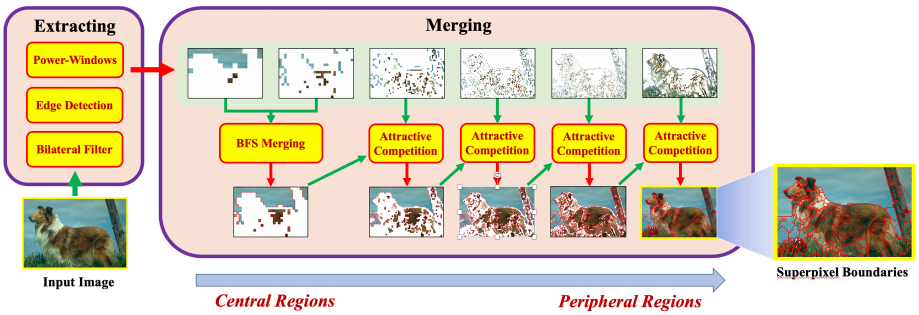


**Fig. 2.** The EAM pipeline consists of two parts: the extracting process on the left and the merging process on the right. The green arrow in the figure indicates the input and the red arrow indicates the output. Refer to Fig. 3 and Fig. 4 for more details of the extracting stage.

In this section we will describe in detail the EAM method. Our method (See Fig. 2) consists of two stages, extracting and merging. The extracting stage is essentially a process of pixel classification. According to the proposed concept of regional attribute, we use squares with various sizes (power-window) to classify pixels into several categories. The pixels in each window are regarded as a whole, and they have the same regional attribute. The larger the window is, the closer it is to the central region of a certain object, and vice versa. In the merging stage, the classified power-windows are merged according to their regional attributes. The power-windows of the central region are first merged. Then we build graph model to define the attraction between the power-windows and merge the remaining windows according to their regional attribute from central to peripheral. In this way, we can generate superpixels that are not disorganized while at the same time maximize the consistency with the human visual object boundaries. The degree of detail as well as the number of superpixels can be determined by one parameter $\epsilon_{delta}$, under the condition of semi-manual intervention, which will be further explained later.

### 3.1 Extracting

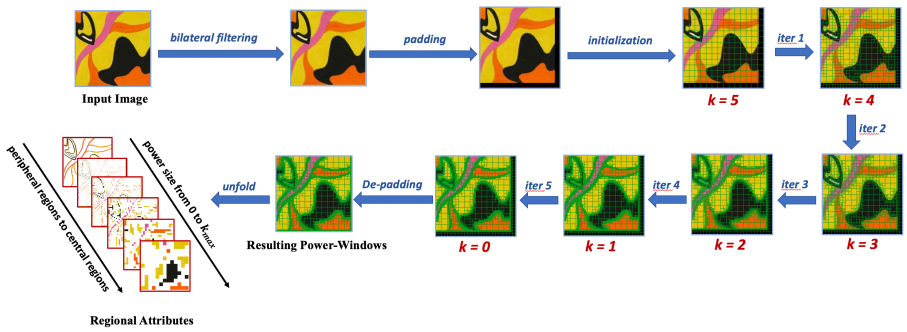


**Fig. 3.** To show the process of extracting more clearly, we deliberately choose an image with simple and clear color. The size of the input image is $321 \times 321$, hence we take the value of $k_{max}$ to be 5 and iterate 5 times in total to get the regional attributes.

We perform bilateral filtering [15] on the original image before extracting. The bilateral filtering method can remove the noise in the image while keeping the boundary definition as much as possible, which can greatly help our subsequent extracting process. The method can be easily used by calling the *opencv* package in python.

**Power-Window**: In order to extract the regional attributes of the pixels in the image, we designed a structure of $(k_{max}+1)$ different sizes of square windows with a side length of $2^0, 2^1, ..., 2^{k_{max}}$ (called **power-window**) to assist the process. $k$ is called the **power size** of a power-window with the side length of $2^k$. We define that

$$k_{max} = \lfloor \log_2 \frac{min(h,w)}{10} \rfloor \tag{1}$$

where $h$ is the height of the image and $w$ is the width. Each power-window can be defined as a triple $(x, y, k)$, where $(x, y)$ represents the coordinate of the upper left corner of the window and $k$ represents its power size. The algorithm is executed as follows. First, we initialize the original image (size $h \times w$) into a series of power-windows with power size of $k_{max}$, $\frac{h}{k_{max}}$ rows and $\frac{w}{k_{max}}$ columns. If $h$ or $w$ are not divisible by $k_{max}$, we pad black pixels to the bottom and right sides of the original image corresponding to the amount missing (padding) and remove them (de-padding) when the stage is complete. Then we enumerate the windows in order $k$ from $k_{max}$ to 0. For the current window with power size $k$, if its *boundary clearness* $< \epsilon_{bc}$, the attribute $k$ of this region will be determined. Otherwise it will be divided into four power-windows with power size $k-1$ for the next iteration and so on, until the power size equals to 1. Boundary clearness is a quantity that we define to determine whether a power-window contains a single object, which will be explained in more detail in the following section.

And $\epsilon_{bc}$ is the threshold for boundary clearness which is set to 3 in this paper. Refer to the Alg. 1 and Fig. 3 for more details.

**Algorithm 1** Attributes Extracting

**Input:** Input image $I$ after bilateral filtering and padding
**Output:** Sets of power-windows with different sizes: $set_0, set_1, ..., set_{k_{max}}$

Empty $set_0, set_1, ..., set_{k_{max}}$
Split $I$ into power-windows with power size $k_{max}$
Insert the power-windows into $set_{k_{max}}$
**for** $k = k_{max} \to 1$ **do**
    **for** each $pw_i \in set_k$ **do**
        **if** BoundaryClearness($pw_i$) $< \epsilon_{bc}$ **then**
            **Erase** $pw_i$ from $set_k$
            **Split** $pw_i$ into 4 smaller power-windows.
            **Insert** the 4 power-windows into $set_{k-1}$
        **end if**
    **end for**
**end for**
**return** $set_0, set_1, ..., set_{k_{max}}$

**Boundary Clearness** : To determine whether a power-window contains only one single object, it is necessary to calculate the boundary clearness of the image within the window. We present a graph-based definition of the concept of boundary clearness and propose a corresponding calculation method. Items in a power-window can be interpreted as a graph $G = (V, E)$, where four corners of each pixel in the window are the vertices and the boundaries between pixels are the edges (see Fig. 4). We define the weight of each edge as the Euclidean distance in [rgb] space of the two pixels with this edge as a dividing line. Our proposed algorithm for calculating boundary clearness can be roughly seen as a process of adding edges to the graph. First, we add to the graph all the outermost edges, which represent the boundaries of the power-window and form a large cycle. Next, we add the remaining edges one by one according to the descending order of their weights, until a new cycle appears in the graph. The weight of the last edge added to the graph is the expected boundary clearness. To explain further, if the weight is large enough, it means that the two regions inside and outside the new cycle have clear boundaries, which can be considered as two objects. Refer to the Alg. 2 and Fig. 4 for more details.

The resulting power-windows with different power sizes through this stage correspond to different regional attributes. The larger the resulting power-window is, the closer it is to the central region of a certain object, and vice versa. In the graph struture of the method calculating boundary clearness, the number of vertices and edges are both on the same order of magnitude as the number of pixels in the power-window. We use the DSU (disjoint-set-union) to determine if a new cycle is generated, so the time complexity of the method is linear. For the

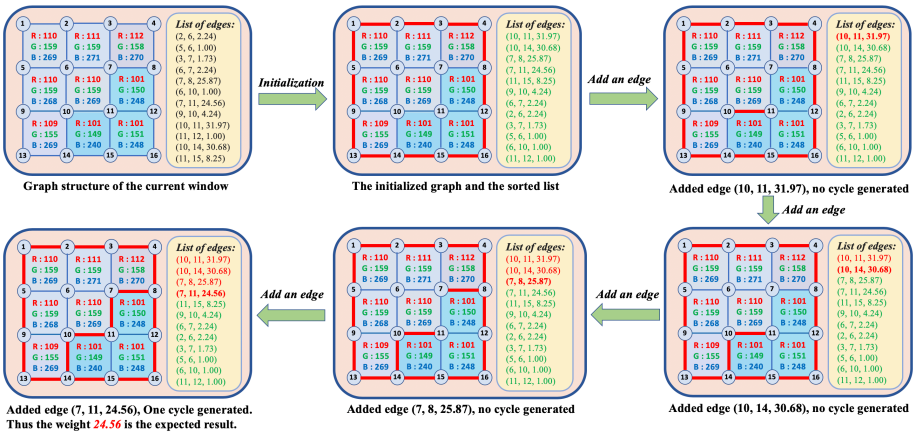


**Fig. 4. Computation flow of boundary clearness.** The example shows a portion of an image with nine pixels. A weighted graph is set up and the edge list is sorted at first. The edges are added one by one until a new cycle is generated. The cycle encloses three pixels in the right bottom corner that express another object in the image. The last edge weight 24.56 represents the boundary clearness of these two regions. The larger the boundary clearness value is, the less likely we think these two regions belong to the same object and we should try smaller power-windows, and vice versa.

**Algorithm 2** Boundary Clearness

**Input:** Input power-window $P$
**Output:** The Boundary Clearness of $P$

Initialize set $V$
$V$ contains corner nodes of pixels in $P$.
Initialize set $E$
$E$ contains edges represented as triples $(u, v, w)$, which $u, v \in V$ and $w$ is defined as the Euclidean distance in [rgb] space of the two pixels which divided by the edge.
Initialize set $E'$ as a subset of $E$ containing the outermost edges of $P$.
Initialize $G = (V, E')$
List $E \setminus E'$ into $L$ as $[edge_1, edge_2, ...]$
Sort $L$ by $edge.w$ in **descending** order.
**for** each $edge_i \in L$ **do**
    Add $edge_i$ to $G$
    **if** a new cycle in $G$ generated by $edge_i$ **then**
        $LastEdge \leftarrow edge_i$
        **break**
    **end if**
**end for**
**return** $LastEdge.w$

entire extracting method, the number of iterations is at most $K_{max}$, which is on the order of logarithm. Therefore, the time complexity of extracting method is $O(NlogN)$, where $N$ is the number of pixels in the input image. The proposed method has the following outstanding features:

1) Simple definition of power-windows. Each window is a square structure which could be defined simply by the upper left coordinate $(x, y)$ and the power size $k$.
2) The square structure with a side length of $2^k$ can be easily cut into four squares with the same side length of $2^{k-1}$, which can be seen as an efficient half-interation process.
3) The proposed definition and solution method of boundary clearness can itelligently ignore the color gradation in the same object and be sensitive to the real object boundaries.

### 3.2 Merging

Using the resulting power-windows extracted in last stage, we propose a graph-based methods for merging them into superpixels. Since the image pixels in each power-window have the same regional attribute, each power-window is integrally treated as an independent vertex in the graph, and the adjacencies between power-windows are the edges. The two types of power-windows with the largest power size (i.e. $k_{max}$, $k_{max}-1$) are first merged through BFS-based (breath-first search) algorithm. For the smaller windows, we enumerate in descending order of power size values and merge them through the proposed Attractive Competive method. Each of the superpixel we ultimately generate is mapped to a label, called **spColor**. The output of the merging stage is a map called $ColorMap$, which records the spColor of each vertices in the graph structure. The initial spColor of all vertices is 0, and the final spColor of them are integer numbers between 1 and the total number of superpixels generated. Our final superpixels are generated according to $ColorMap$ by mapping the spColor of the vertices to the corresponding pixel area in the original image. Image pixels with the same spColor belong to the same superpixel.

**Graph Structure:** To merge power-windows into superpixels, the proposed method represents the image as a graph where vertices are power-windows and the edge weight are the similarities between the adjacent windows (i.e. sharing a boundary). In this work, the [rgb] color of a power-window is defined as the mean color of the pixels inside it and the similarity of two adjacent power-windows is defined as the Euclidean distance in [rgb] space of their colors. It is clear that the number of vertices and edges are both of the same order as the number of pixels of the image, expressed by $N$. The structure is more clearly shown in Fig. 5.

**BFS-based Merging:** The power-windows with the two largest power size $k_{max}, k_{max}-1$ represent the central regions, which should be merged into connected regions first. The algorithm consists of two iterations, while each power-windows with $k_{max}$ and $k_{max}-1$ power size are traversed respectively to expand into some connected regions. The expansion of vertices has three limitations.

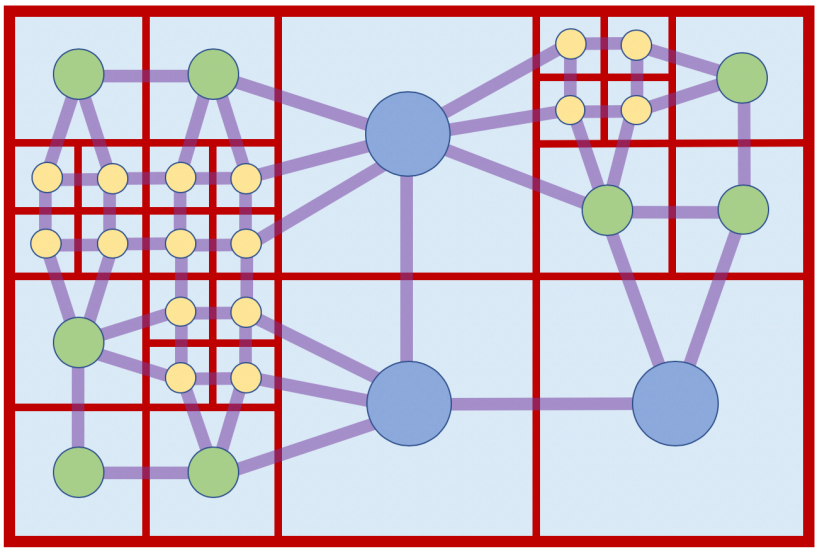

**Fig. 5. Graph structure:** Three types of power-windows separated by red lines with different sizes are represented in the shown graph by three different types of vertices. The purple lines in the figure are the edges connecting adjacent vertices.

**Algorithm 3** BFS-based Merging

**Input:** Graph $G = (V, E)$ created on power-windows
**Output:** $ColorMap$

**function** BfsExtension($v, ColorMap, spColor$)
  Initialize an empty $Queue$
  Push $v$ into $Queue$
  $ColorMap[v] \leftarrow spColor$
  **while** $Queue$ is not empty **do**
    Pop a vertex $v$ from $Queue$
    **for** each uncolored $v'$ adjacent to $v$ **do**
      **if** Distance($v.color, v'.color$) $\leq \epsilon_{bfs}$ **then**
        $ColorMap[v'] \leftarrow spColor$
        Push $v'$ into $Queue$
      **end if**
    **end for**
  **end while**
**end function**

**for** each uncolored $v \in V$ **do**
  **if** $v.power\_size \geq k_{max} - 1$ **then**
    $spColor \leftarrow spColor + 1$
    BfsExtension($v, ColorMap, spColor$)
  **end if**
**end for**
**return** $ColorMap$

1) Only adjacent vertices can be expanded, just as in general BFS algorithm.
2) Only vertices with power size $\geq k_{max} - 1$ can be expanded.
3) A vertex only expands the vertices which the weight of the edge between them $< \epsilon_{bfs}$. The threshold $\epsilon_{bfs}$ is set to 25 in this work.

The limitations are intended to form better initial subject regions, laying the groundwork for the subsequent Attractive Competive algorithm. Refer to the Alg. 3 and Fig. 2 for more details.

**Attraction Competition**: In this section, We define the concept of attraction between two vertices as the length of the shortest path between them. Further more, we propose an efficient competition algorithm based on Dijkstra Alg. [3]. The proposed algorithm goes through $k_{max} - 1$ iterations (i.e. from power size $(k_{max} - 2)$ downto 0). For each iteration, we compute the shortest path from the colored vertices (spColor of which is not 0) to the uncolored ones with Dijkstra algorithm. When one vertex $u$ relaxes another vertex $v$ in the process, we assign the spColor of $u$ to $v$ temporarily. As $v$ may be relaxed again soon by another vertex $u'$ and the spColor of $v$ should change to the the spColor of $u'$, the process is just like an attraction competition between the vertices. We do iterations in order of power size from $(k_{max} - 2)$ to 0, so in each iteration only the vertices with the current power size are colored. For the verices that are far away from any colored vertices, we take a clustering approach to merge them and form new color regions, similar to SLIC [1]. This is done in the five-dimensional $[rgbxy]$ space, where $[rgb]$ is the color of the power-window represented by a vertex, and $[xy]$ is the middle pixel position of the power-window. The initial cluster centers are selected randomly and the distance measure $D$ is defined as

$$D = d_{rgb} + \alpha d_{xy} \tag{2}$$

where $d_{rgb}$ and $d_{xy}$ is the Euclidean norm in $[rgb]$ space and $[xy]$ space respectively. $\alpha$ is set to 0.02 in this paper. A **binary search** method is taken to minimize the number of new colors to added under the condition that the maximum color difference in one cluster is $\leq \epsilon_{cl}$. The threshold $\epsilon_{cl}$ is the only parameter in this work allowing us to control the degree of detail and the number of superpixels. The smaller $\epsilon_{cl}$ is set, the finer the superpixels are and the more the superpixel numbers, vice versa. In particular, for vertices with power size = 0, we only let them be attracted to the colored regions but cannot cluster by themselves as they express the details of the image boundaries. Refer to the Alg. 4 and Fig. 2 for more details of Attraction Competition method.

As the number of vertices and edges in our proposed graph structure are both of the same order as the number of the pixels of the image, expressed by $N$, the time complexity of BFS-based Merging is $O(N)$. For the subsequent Attraction Competition method, $k_{max} - 1$ iterations are executed which is on the order of logarithm. Further more, the complexity of Dijkstra algorithm with priority queue is $O(NlogN)$ and the complexity of the Cluster Method with few constant number of iterations is $O(N)$. So the total complexity of this algorithm is $O(Nlog^2N)$.

**Algorithm 4** Attraction Competition

**Input:** Graph $G = (V, E)$ created on power-windows
**Output:** $ColorMap$

**for** $k = k_{max} - 2 \to 0$ **do**
  $Set_{ori} \leftarrow$ colored vertices
  $Set_{ter} \leftarrow$ vertices with power size $k$
  DIJKSTRA($Set_{ori}, Set_{ter}, ColorMap$)
  **Update** $ColorMap$
  **if** $k > 0$ **then**
    $Set' \leftarrow$ uncolored vertices in $Set_{ter}$
    $L \leftarrow 1, R \leftarrow Set'.size$
    **while** $L < R$ **do**
      $Mid = \lfloor \frac{L+R}{2} \rfloor$
      $Dif_{max} \leftarrow$ CLUSTER($Set', Mid$)
      **if** $Dif_{max} > \epsilon_{cl}$ **then**
        $R \leftarrow Mid$
        **Update** $ColorMap$
      **else**
        $L \leftarrow Mid$
      **end if**
    **end while**
  **end if**
**end for**
**return** $ColorMap$

## 4 Experiments

In this section, the proposed method is compared with some unsupervised state-of-the-art superpixel segmentation methods, including two representative clustering-based algorithms (SLIC [1] and LSC [20]), a graph-based algorithm (LRW [13]) and an algorithm based on energy optimization (SEEDS [2]) on **BSDS500**[5] dataset. Furthermore, we also compared our method with two latest deep learning based superpixel segmentation methods (SEAL [16] and SSN[6]) to show the great advantage on **EV** [11], which quantifies the variation of the image explained by the superpixels. We also show that our algorithm can retain more details in the visual comparison, and it is consistent with EV score.

### 4.1 Datasets and Performance metrics

For evaluation of the proposed method, we use the BSDS500 dataset which containing 500 images with size $321 \times 481$. Achievable Segmentation Accuracy (ASA) [9] is the most widely adopted criteria for measuring superpixel segmentation quality. Moreover, in order to better measure how well the data in the original pixels is represented by superpixels we use Explained Variation (EV) [11]. The detailed definitions of these metrics are as follows:

[5] https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping

**Achievable Segmentation Accuracy (ASA)**: ASA [9] quantifies the achievable accuracy for segmentation using superpixels as pre-processing step. The performance of subsequent processing is expected to be unaffected. Then ASA is defined as:

$$ASA(G,S) = \frac{\sum_i \max_j |S_i \cap G_j|}{\sum_j |G_j|} \tag{3}$$

ASA computes the highest achievable accuracy by labeling each superpixel with the label of ground truth segmentation that has the biggest overlap area, i.e. higher is better.

**Explained Variation (EV)**: EV [11] evaluates the superpixel segmentation quality independent of annotated ground truth. As image boundaries tend to exhibit strong change in color and structure, EV assesses boundary adherence independent of human annotions. EV is defined as:

$$EV(S) = \frac{\sum_{S_j} |S_j|(\mu(S_j) - \mu(I))^2}{\sum_{x_n} (I(x_n) - \mu(I))^2} \tag{4}$$

where $\mu(S_j)$ and $\mu(I)$ are the mean color of superpixel $S_j$ and the image $I$, respectively. As result, EV quantifies the variation of the image explained by the superpixels, i.e. higher is better.

### 4.2 Compare with State-of-the-art

In all comparative experiments in this section, instead of directly controlling the number of superpixels manually, we use the cluster diameter threshold $\epsilon_{cl}$ to control the degree of detail of segmentation since it is difficult for the human eye to determine the appropriate number of superpixels of an image but the degree of detail. We obtained the average number of superpixels by adjusting the value of $\epsilon_{cl}$ on the BSDS500 dataset, as shown in Table. 1.

| Number of superpixels | 100 | 200 | 300 | 400 | 500 |
|---|---|---|---|---|---|
| $\epsilon_{cl}$ | 30 | 20 | 15 | 8 | 5 |

**Table 1.** The correspondence between $\epsilon_{cl}$ and the average number of superpixels of EAM in the experiment on the BSDS500 dataset.

We compare EAM with four unsupervised state-of-the-art superpixel algorithms, including SLIC [1], LSC [20], LRW [13] and SEEDS [2] on the BSDS500 dataset. For all four algorithms, the implementations are based on publicly available codes. The experiments are performed on the BSDS500 dataset. EAM has achieved higher ASA than other methods as shown in the **leftmost** graph of Fig. 6. It shows that our method can better express the object boundaries.

In order to test the performance of our method in more depth, we also performed comparison experiments with two latest deep learning based methods

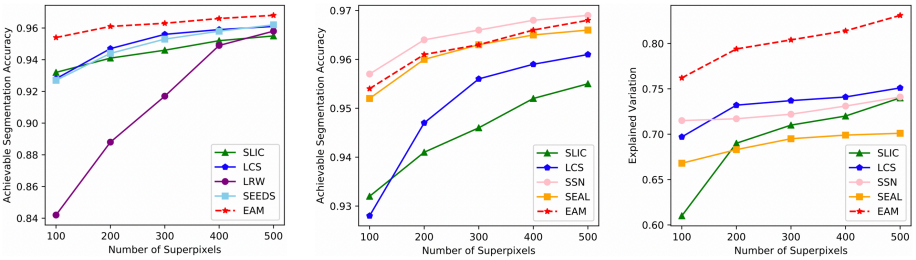


**Fig. 6.** Comparing state-of-the-art methods for a range of 5 different numbers of superpixels on the BSDS500 datasets, ASA on the left and middle, EV on the right. The leftmost graph shows the performance comparison of the unsupervised methods and the middle one shows the deep learning based methods. For each such value, all methods were initialized, and ended with about the same number of superpixels.

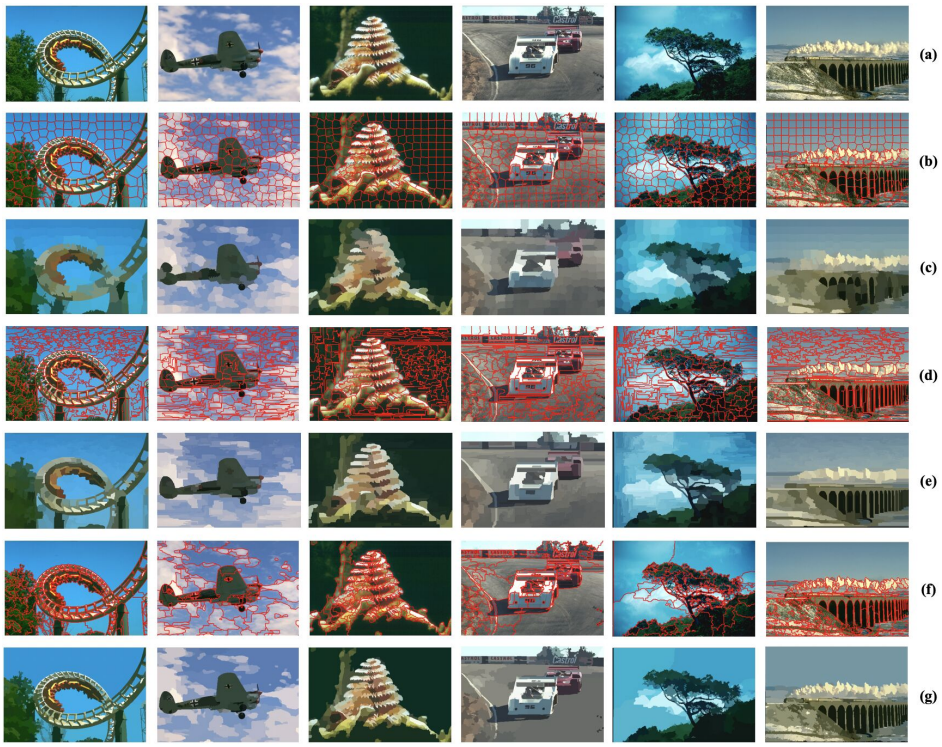


**Fig. 7.** Visual comparison of superpixel segmentation results, the superpixel boundaries and mean colors. (a) Origional. (b) SLIC (boundaries). (c) SLIC (mean color). (d) SEAL (boundaries). (e) SEAL (mean color). (f) EAM (boundaries). (g) EAM (mean color). The average superpixel numbers in all images is roughly 200. All methods is initialized and ended with about the same number of superpixels. The images with mean color show huge advantage of EAM compared to other methods in term of detail retention, as demonstrated by EV score.

(SEAL [16] and SSN [6]) on 200 images of the BSDS500 dataset (300 images for training). The results are also presented on Fig. 6. The experimental results of two kinds of unsupervised state-of-the-art methods (SLIC [1], LSC [20]) are also added to the line graph, in order to better form the performance comparison between deep learning based methods and unsupervised methods (including the proposed EAM).

In terms of ASA, compared with unsupervised methods, deep learning based methods has obvious advantages. The performance of EAM ranks second, not as good as the algorithms SSN [6], but EAM is the only one unsupervised methods that is comparable to the deep learning based methods. Refer to the **middle** graph of Fig. 6 for more details.

As the explained variation (EV) metric quantifies the quality of superpixel segmentation **without relying on ground truth**, it is more objective for comparison experiments with deep learning based methods. In terms of EV, the proposed method performs the best and shows the huge advantage against the deep learning based methods (SSN [6] and SEAL [16]) and unsupervised methods (LSC [20] and SLIC [1]). Refer to the **rightmost** graph of Fig. 6 for more details.

Deep learning methods rely too much on the ground truth from training sets thus they score highly on ASA but not very well on EV. The proposed method can achieve performance comparable to deep learning based methods without the need of extensive sample data for model training and has strong universality, which demonstrates the advantage and effectiveness of this approach.

To make more intuitive, we do visual experiment on the BSDS500 dataset, as shown in Fig. 7. Consistent with the EV score in Fig. 6, EAM retains much more details than other methods under the same superpixel numbers.

For time efficiency, EAM has no training overhead and the total complexity of the method is $O(Nlog^2N)$. Compared with most unsupervised methods such as ERS [10] $O(N^2logN)$ and NC [7] $O(N^{\frac{3}{2}})$, EAM has higher computational efficiency, but with some efficient algorithms such as SLIC [1] $O(N)$, we do need to strengthen.

## 5 Conclusion

In this paper, we focus on the regional attribute of image pixels for the first time in the field of superpixel segmentation. A novel superpixel method called EAM is proposed to generate finer superpixels. Compared with the direct merging of independent pixels, the pre-extraction process can obtain useful prior knowledge (the regional attributes). This allows EAM to smartly divide the detail portion with more superpixels and not waste on the bulk portion.

## Acknowledgments

This work is supported by the NSFC (under Grant 61876130, 61932009).

## References


1. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S.: Slic superpixels compared to state-of-the-art superpixel methods. IEEE Transactions on Pattern Analysis and Machine Intelligence **34**(11), 2274–2282 (Nov 2012). https://doi.org/10.1109/TPAMI.2012.120
2. den Bergh, M.V., Boix, X., Roig, G., Gool, L.V.: SEEDS: superpixels extracted via energy-driven sampling. CoRR **abs/1309.3848** (2013), http://arxiv.org/abs/1309.3848
3. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik **1**(1), 269–271 (Dec 1959). https://doi.org/10.1007/BF01386390, https://doi.org/10.1007/BF01386390
4. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. International Journal of Computer Vision **59**(2), 167–181 (Sep 2004). https://doi.org/10.1023/B:VISI.0000022288.19776.77, https://doi.org/10.1023/B:VISI.0000022288.19776.77
5. Gadde, R., Jampani, V., Kiefel, M., Gehler, P.V.: Superpixel convolutional networks using bilateral inceptions. CoRR **abs/1511.06739** (2015), http://arxiv.org/abs/1511.06739
6. Jampani, V., Sun, D., Liu, M., Yang, M., Kautz, J.: Superpixel sampling networks. CoRR **abs/1807.10174** (2018), http://arxiv.org/abs/1807.10174
7. Jianbo Shi, Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **22**(8), 888–905 (Aug 2000). https://doi.org/10.1109/34.868688
8. Kwak, S., Hong, S., Han, B.: Weakly supervised semantic segmentation using superpixel pooling network. In: AAAI (2017)
9. Liu, M., Tuzel, O., Ramalingam, S., Chellappa, R.: Entropy rate superpixel segmentation. In: CVPR 2011. pp. 2097–2104 (June 2011). https://doi.org/10.1109/CVPR.2011.5995323
10. Liu, M.Y., Tuzel, O., Ramalingam, S., Chellappa, R.: Entropy rate superpixel segmentation. CVPR 2011 pp. 2097–2104 (2011)
11. Moore, A.P., Prince, S.J.D., Warrell, J., Mohammed, U., Jones, G.: Superpixel lattices. In: 2008 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–8 (June 2008). https://doi.org/10.1109/CVPR.2008.4587471
12. Sharma, A., Tuzel, O., Liu, M.Y.: Recursive context propagation network for semantic scene labeling. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 27, pp. 2447–2455. Curran Associates, Inc. (2014), http://papers.nips.cc/paper/5282-recursive-context-propagation-network-for-semantic-scene-labeling.pdf
13. Shen, J., Du, Y., Wang, W., Li, X.: Lazy random walks for superpixel segmentation. IEEE Transactions on Image Processing **23**(4), 1451–1462 (April 2014). https://doi.org/10.1109/TIP.2014.2302892
14. Shu, G., Dehghan, A., Shah, M.: Improving an object detector and extracting regions using superpixels. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition. pp. 3721–3727 (June 2013). https://doi.org/10.1109/CVPR.2013.477
15. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271). pp. 839–846 (Jan 1998). https://doi.org/10.1109/ICCV.1998.710815
16. Tu, W.C., Liu, M.Y., Jampani, V., Sun, D., Chien, S.Y., Yang, M.H., Kautz, J.: Learning superpixels with segmentation-aware affinity loss. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)

17. Yan, J., Yu, Y., Zhu, X., Lei, Z., Li, S.Z.: Object detection by labeling superpixels. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5107–5116 (June 2015). https://doi.org/10.1109/CVPR.2015.7299146
18. Yang, C., Zhang, L., Lu, H., Ruan, X., Yang, M.H.: Saliency detection via graph-based manifold ranking. 2013 IEEE Conference on Computer Vision and Pattern Recognition pp. 3166–3173 (2013)
19. Yang, F., Lu, H., Yang, M.H.: Robust superpixel tracking. Trans. Img. Proc. **23**(4), 1639–1651 (Apr 2014). https://doi.org/10.1109/TIP.2014.2300823, https://doi.org/10.1109/TIP.2014.2300823
20. Zhengqin Li, Jiansheng Chen: Superpixel segmentation using linear spectral clustering. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1356–1363 (June 2015). https://doi.org/10.1109/CVPR.2015.7298741
21. Zhu, W., Liang, S., Wei, Y., Sun, J.: Saliency optimization from robust background detection. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. pp. 2814–2821 (June 2014). https://doi.org/10.1109/CVPR.2014.360