Computer Graphics Forum

Volume xx (200y), Number z, pp. 1–12

Superpixel Generation by Agglomerative Clustering with Quadratic Error Minimization

Xiao Dong[1,3], Zhonggui Chen[†2], Junfeng Yao[1], and Xiaohu Guo [‡3]

[1]Software School, Xiamen University, China
[2]Department of Computer Science, Xiamen University, China
[3]Department of Computer Science, University of Texas at Dallas, USA

Abstract

Superpixel segmentation is a popular image preprocessing technique in many computer vision applications. In this paper we present a novel superpixel generation algorithm by agglomerative clustering with quadratic error minimization. We use a quadratic error metric (QEM) to measure the difference of spatial compactness and color homogeneity between superpixels. Based on the quadratic function, we propose a bottom-up greedy clustering algorithm to obtain higher quality superpixel segmentation. There are two steps in our algorithm: merging and swapping. First, we calculate the merging cost of two superpixels and iteratively merge the pair with the minimum cost until the termination condition is satisfied. Then, we optimize the boundary of superpixels by swapping pixels according to their swapping cost to improve the compactness. Due to the quadratic nature of the energy function, each of these atomic operations has only $O(1)$ time complexity. We compare the new method with other state-of-the-art superpixel generation algorithms on two datasets, and our algorithm demonstrates superior performance.

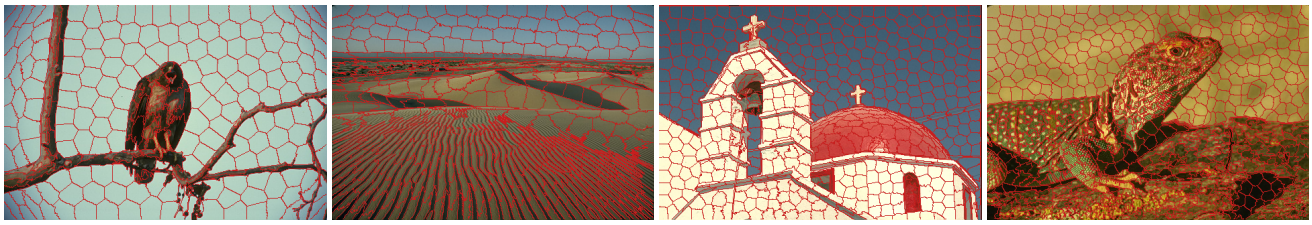
1. Introduction

Superpixel generation divides an image into several regions. Pixels in one region are connected and have similar color values, and each region is a perceptually meaningful small patch that adheres well to object boundaries. The concept of superpixels as a preprocessing step was first introduced by Ren and Malik [RM03] in 2003. This high level image representation can greatly reduce the computation complexity of many image processing tasks such as segmentation [PZZ11, LWC12, SPD*17, DSSVG16], saliency detection [PKPH12, WSS18], object tracking [WLYY11], modeling [HZR06, NC08], and 3D reconstruction [HEH05].

Many superpixel generation algorithms have been proposed in recent years. Some algorithms are designed for video superpixel generation [LSD*16]. The segmentation results of each algorithm have their own characteristics in terms of superpixel size and shape. Although the quality of image segmentation is a subjective judgment, high quality results generally meet the following criteria. First, the superpixel must be a connected region, meaning that all pixels within a superpixel can be traversed by adjacency. Second, the algorithm needs to take the spatial compactness of superpixels into account. For regions having no boundaries or features, it should exhibit regular and compact shape pattern. Third, the segmentation results should be adaptive to local image contents, i.e., pixels with similar color and intensity should be grouped into the same region. Last but not the least, one superpixel should not contain multiple objects, and it should adhere well to the object boundaries. These are important criteria for measuring the quality of superpixels.

A traditional type of superpixel segmentation methods is based on Lloyd's algorithm [Llo82], by iterating through the following two steps: (1) growing superpixel regions starting from some initially-selected seeds; (2) moving the seeds to better locations based on the current segmentation. Both steps are guided by some well-defined energy functions. Due to the non-convex nature of optimization, it usually results in local minima of their corresponding energy functions, and the results highly depend on the initial placement of seeds. In contrast to these optimization methods, there is no seed in our algorithm. Our optimization is based on agglomerative clustering with merging and swapping operations. The use of a quadratic energy function guarantees that each of these atomic operations is of $O(1)$ time complexity. Each step of the bottom-up clustering chooses a pair of clusters with the least energy increase after merging in a greedy fashion, thus it eliminates the dependency of optimization results on initial seed placement. This brings a significant advantage as can be seen from the experimental results in Sec. 4. Even though our quadratic energy function is similar to SLIC [ASS*12] and MSLIC [LYYH16], our optimization achieves higher quality of superpixel results, evaluated with commonly-used performance metrics on various image datasets.

† Corresponding author:chenzhonggui@xmu.edu.cn
‡ Corresponding author:xguo@utdallas.edu



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

Figure 1: Superpixel Segmentation using our method. (a) 300 superpixels; (b) 500 superpixels; (c) 900 superpixels; (d) 1500 superpixels.

2. Related Work

The difficulty of superpixel generation algorithms lies in the need not only to fit the boundary of objects well, but also to take the compactness of superpixels into consideration. The early methods used to segment images include watersheds [VS91], mean shift [CM02] and quick shift [VS08]. Quick shift arranges all of the data points into a tree where parents in the tree are the nearest neighbors in the feature space which increases the estimate of density. Mean shift defines a density function and attempts to maximize the function by moving the pixels within a window towards areas of higher density. In these methods, there is no consideration of spatial distance of pixels, so the shape is generally irregular. In recent years, newly proposed algorithms have taken into account the spatial compactness. The superpixel generation methods can be roughly classified into several families. In first category, the algorithms are based on the idea of graph theory and gradually add cuts to segment the graph. Usually, the pixels are taken as nodes of the graph with edges representing the similarities between pixels. In second category, algorithms generally use the gradient image, and grow clusters from selected seeds.

2.1. Graph-based Methods

Wu and Leahy [WL93] proposed a clustering method which globally minimizes a graph-based objective function to find the optimal partition. Based on this work, Shi and Malik introduced Normalized Cuts [SM00] which avoid favouring the cuts in small sets of nodes in the graph. Normalized Cuts is less effective when the number of superpixels grows, and some other methods were proposed to speed it up [EOK07, XLS09]. Felzenszwalb and Hutenlocher [FH04] proposed an agglomerative clustering algorithm, each superpixel is the minimum spanning tree of the constituent pixels. However, the algorithm only uses the color information to calculate the similarity between pixels, so the boundaries of superpixels are usually irregular. Moore et al. [MPW*08, MPW10] presented the optimal cuts by using pre-computed boundary maps, but the quality of such boundary maps affects the performance of algorithm. Liu et al. [LTRC11] introduced a method that maximizes the entropy rate of cuts in graph. There are other methods based on graph-cuts to improve the efficiency of algorithm [VBM10, ZHMB11].

2.2. Seed-based Methods

The second type of algorithms defines an energy function to evaluate the similarities between pixels and grows superpixels from assigned seeds. Turbopixels [LSK*09] is a geometric flow method that uses level-set, and the results are highly uniform. One obvious drawback of Turbopixels is that their lattice-like results present relatively low adherence to boundaries of objects. SLIC [ASS*12] uses iterative K-means clustering, and generates clusters by measuring the similarities between pixels and centers in the combined five-dimensional (color and coordinate) space. EWCVT [WJW09] and VCells [WW12] are essentially Centroidal Voronoi Tessellations (CVT) with compactness constraints. Peng et al. [PSYL16] adopted K-means clustering to get initial partition and applied high order energy function to refine the results. The main drawback of these methods is that they can not adhere the weak boundary well in the graph. Recently, Wang et al. [WZG*13] presented an image segmentation algorithm based on a geodesic distance metric. Based on this, Zhou et al. [ZPW*16, PZLZ17] proposed a Bilateral Geodesic Distance to improve the boundary adherence at weak boundaries, and we refer it as BGD. Shen et al. [SDWL14] presented the Lazy Random Walk(LRW) method to compute the probabilities of pixels and according to which the boundaries of image are obtained. The authors also proposed a real-time image superpixel segmentation method called DBSCAN [SHL*16] by using the density-based spatial clustering. Liu et al. [LYYH16] proposed the MSLIC method which extends the conventional SLIC algorithm. MSLIC algorithm lifts pixels to a 2-manifold M embedded in the 5-dimensional space and computes restricted centroidal Voronoi tessellations under the Euclidean metric. As an improvment of MSLIC, the authors proposed IMSLIC [LYLH18] method to compute a geodesic centroidal Voronoi tessellations on the image manifold M instead. Cai et al. [CG16] presented a Gaussian Generative Model for anisotropic superpixel generation based on Mahalanobis distance, and it is referred as GGM in this paper.

2.3. Other Methods

Bergh et al. proposed SEEDS [VdBBR*12] based on hill-climbing optimization that measures color homogeneity and shape regularity by histograms. It proposes new partitions at pixel-level and block-level to avoid excessive operations. However, SEEDS also suffers from high shape-irregularity. Li et al. [LC15] presented the LSC algorithm which combines the normalized cuts formulation and weighted K-means method, and still uses seed points to grow super-

pixels. This method can produce compact and uniform superpixels with low computational costs. Zhang et al. [ZMZZ17] proposed a superpixel generation algorithm by simplifying the 3D triangle mesh model based on the quadric error metric (QEM) [GH97] framework. Their main problem is that the edges of a triangle mesh can not fit the boundaries of objects very well. Our new method is inspired by the GGM method [CG16], but we integrate agglomerative clustering with the well known quadratic CVT energy function that measures color homogeneity and spatial compactness. We illustrate in Sec. 4 that our method is more sensitive to weak boundaries and has better boundary recall rates (Fig.6 and Fig.9) than GGM method. Fig.1 shows some segmentation results of our method. We can see that our new method generates compact superpixels in background and also preserves the boundaries of objects very well.

3. Agglomerative Clustering with Quadratic Error Minimization

In this section, we present our superpixel algorithm which not only produces superpixels with compact shape but also captures the boundaries of objects in image. The algorithm for generating superpixels is inspired by the idea of Quadric Error Metric (QEM) [GH97]. As we know, QEM is a surface simplification method for triangle meshes, and it is built upon edge contraction and quadric errors. We define quadratic functions to measure the color homogeneity and spatial compactness of the superpixels. Our algorithm includes two main steps: merging and swapping. The merging process is a greedy clustering method, similar to edge contraction in QEM, we merge superpixel pairs according to their quadratic merging cost. The use of quadratic energy function guarantees that all these atomic operations are of $O(1)$ time complexity, which ensures the high efficiency of our algorithm.

3.1. Definition of Objective Function

Given an input image I with its two-dimensional domain Ω, the set of superpixels C is a partition of I into k smaller regions. We denote the partition as $P = \{C_i\}_{i=1}^{k}$, which satisfies $C_i \cap C_j = \emptyset$ for $i \neq j$, and $\cup_i C_i = \Omega$. In this paper, we measure the similarity of pixels from two aspects: color homogeneity and spatial compactness. In some previous algorithms [WZG*13, ZPW*16], instead of using traditional Euclidean distance, they presented a measure called geodesic distance to improve the segmentation results. Some other algorithms [LC15] calculated the similarity in a ten-dimensional feature space combining colors and coordinates. From our perspectives, since the metrics of spatial distance and color distance are different, we introduce two energy terms to measure the spatial compactness and color homogeneity in our energy function $F(P)$, and combine them with a parameter λ, which can be automatically decided during the optimization process (described in Sec. 3.3). We find the optimal partition by minimizing the following energy function:

$$\begin{aligned} F(P) &= \lambda F_{spatial}(P) + F_{color}(P) \\ &= \sum_{i=1}^{k} (\lambda F_{spatial}(C_i) + F_{color}(C_i)), \end{aligned} \tag{1}$$

where $F_{spatial}$ measures spatial compactness and F_{color} measures color homogeneity.

Algorithm 1 Superpixel Algorithm Framework

Input:
An input image I, and user-specified number of superpixels k;
Output:
An optimal k-partition of the image I.
1: Merge pixels to k superpixels as initial segmentation using Algorithm 2;
2: Optimize the energy funtion by swapping pixels between neighboring superpixels using Algorithm 3;
3: Enforce connectivity.

Spatial Compactness: We denote a pixel of an image as $\mathbf{p} = (\mathbf{p}_s, \mathbf{p}_c)$, where $\mathbf{p}_s$ is a two-dimensional vector, representing its coordinates; $\mathbf{p}_c$ is a three-dimensional vector, representing its color. For a superpixel C containing several pixels, we expect C to have a regular shape in non-feature or non-boundary regions of an image. Similar to the CVT energy, the $F_{spatial}$ term is defined as follows:

$$F_{spatial}(C) = \sum_{\mathbf{p} \in C} \|\mathbf{p}_s - \bar{\mathbf{p}}_s(C)\|^2 . \tag{2}$$

Where $\bar{\mathbf{p}}_s(C)$ represents the barycenter of the cluster C. It is easy to convert the formula into a matrix form and represent it with the trace of a superpixel's covariance matrix:

$$\begin{aligned} F_{spatial}(C) &= \sum_{\mathbf{p} \in C} \|\mathbf{p}_s - \bar{\mathbf{p}}_s(C)\|^2 \\ &= \sum_{\mathbf{p} \in C} Tr(\mathbf{p}_s - \bar{\mathbf{p}}_s(C))^{\mathrm{T}} (\mathbf{p}_s - \bar{\mathbf{p}}_s(C))) \\ &= Tr(\sum_{\mathbf{p} \in C} (\mathbf{p}_s - \bar{\mathbf{p}}_s(C))(\mathbf{p}_s - \bar{\mathbf{p}}_s(C))^{\mathrm{T}}) \\ &= Tr(\mathbf{M}_s(C)). \end{aligned} \tag{3}$$

Here, $\mathbf{M}_s(C) = \sum_{\mathbf{p} \in C} (\mathbf{p}_s - \bar{\mathbf{p}}_s(C))(\mathbf{p}_s - \bar{\mathbf{p}}_s(C))^{\mathrm{T}}$ is the 2×2 covariance matrix for pixel coordinates in 2D.

Color Homogeneity: The color distribution of a superpixel C is expected to be uniform. We use the CIELAB color space, and use the same CVT formula to measure color homogeneity:

$$\begin{aligned} F_{color}(C) &= \sum_{\mathbf{p} \in C} \|\mathbf{p}_c - \bar{\mathbf{p}}_c(C)\|^2 \\ &= Tr(\mathbf{M}_c(C)). \end{aligned} \tag{4}$$

Similarly, $\mathbf{M}_c(C) = \sum_{\mathbf{p} \in C} (\mathbf{p}_c - \bar{\mathbf{p}}_c(C))(\mathbf{p}_c - \bar{\mathbf{p}}_c(C))^{\mathrm{T}}$ is the 3×3 covariance matrix for colors in 3D CIELAB space.

3.2. Agglomerative Clustering Algorithm

In this section, we show the framework of our agglomerative clustering algorithm. The algorithm contains two main parts: *merging* and *swapping*. The first step of our algorithm starts from treating each pixel in the image as an individual superpixel, and iteratively merging pairs of superpixels with the least energy increase in a greedy way. Superpixel number will decrease from the total pixel number n to the user specified k. In the seond step, taking the merging result as initial segementation, our energy function is optimized by swapping boundary pixels between neighboring superpixels. We then enforce the connectivity of suerpixels in the last step.

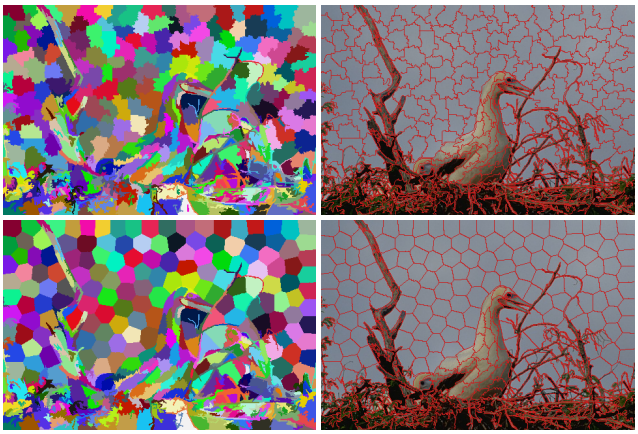


Figure 2: Image segmentation results with 500 superpixels. The partition in first row shows superpixels after merging, compactness $CO = 0.29$; the second row shows partition after swapping, compactness $CO = 0.43$.

Merging: As shown in Eqs. (3) and (4), the objective function can be computed from the two covariance matrices $\mathbf{M}_s(C)$ and $\mathbf{M}_c(C)$. Start from treating each pixel as a superpixel, the merging process iteratively merges two superpixels with least energy increase until the number of superpixels is reduced to the specified value k. During merging operation, it is necessary to calculate the merging cost of two superpixels. Suppose we have a candidate pair of superpixels C_i and C_j that will be merged into C_k. We need to calculate the merging cost ΔF and place the merging pair (C_i, C_j) with ΔF in a min-heap. We denote the merging operation as $C_i + C_j \rightarrow C_k$. As $F_{spatial}$ and F_{color} are defined in a similar fashion, we ignore the subscript s, c and denote the average values of clusters C_i, C_j and C_k with symbols $\bar{\mathbf{p}}(C_i)$, $\bar{\mathbf{p}}(C_j)$ and $\bar{\mathbf{p}}(C_k)$, respectively. To calculate the merging cost, we update the average value $\bar{\mathbf{p}}(C_k)$ of new cluster C_k by using the following equation:

$$\bar{\mathbf{p}}(C_k) = \frac{|C_i|\bar{\mathbf{p}}(C_i) + |C_j|\bar{\mathbf{p}}(C_j)}{|C_i| + |C_j|}, \tag{5}$$

where $|C_i|$ and $|C_j|$ are the numbers of pixels in C_i and C_j. The covariance matrix can be updated by:

$$\begin{aligned}\mathbf{M}(C_k) =& \mathbf{M}(C_i) + |C_i|(\bar{\mathbf{p}}(C_i) - \bar{\mathbf{p}}(C_k))(\bar{\mathbf{p}}(C_i) - \bar{\mathbf{p}}(C_k))^{\mathrm{T}} \\ &+ \mathbf{M}(C_j) + |C_j|(\bar{\mathbf{p}}(C_j) - \bar{\mathbf{p}}(C_k))(\bar{\mathbf{p}}(C_j) - \bar{\mathbf{p}}(C_k))^{\mathrm{T}}.\end{aligned} \tag{6}$$

Note that in the above updates of $\bar{\mathbf{p}}(C_k)$ and $\mathbf{M}(C_k)$, we do not need to sum over all pixels in these superpixels. Each superpixel C can be fully represented by $\sigma_C = \{\mathbf{M}(C), \bar{\mathbf{p}}(C), |C|\}$. Due to the quadratic nature, the objective function of the newly merged cluster C_k can be simply computed from the representatives σ_{C_i} and σ_{C_j} of the two clusters C_i and C_j. This is an operation of $O(1)$ time complexity, which does not depend on the number of pixels in each cluster. The merging cost is positive, and defined as $\Delta F = F(C_k) - F(C_i) - F(C_j)$. As shown in Alg. 2, the pair of superpixels with the minimum merging cost will be first processed.

Swapping: The segmentation results need to be further optimized. In Fig.2 we can see the shape of superpixels is irregular

Algorithm 2 Merging

Input:
An input image I, and number of superpixels k;
Output:
Initial Segmentation of image I.
initialize each superpixel C_i to contain only one pixel $\mathbf{p}_i$;
initialize N to be the number of all pixels;
for each pixel $\mathbf{p}_i$ **do**
  compute the merging cost ΔF of $\mathbf{p}_i$ with its four neighboring pixels;
  place the valid merging pair (C_i, C_j) with ΔF into min-heap H_{min}, ordered by ΔF;
end for
while $N > k$ **do**
  remove a merging pair (C_i, C_j) from the top of heap H_{min};
  if C_i is not empty and C_j is not empty **then**
    merge two superpixels C_i, C_j into C_k;
    minus one from N;
  end if
  delete all merging pairs involving C_i and C_j from H_{min};
  compute new merging pairs of C_k with its neighboring superpixels, as well as their merging cost ΔF;
  place new merging pairs into H_{min}.
end while

Algorithm 3 Swapping

Input:
Initial Segmentation after merging;
Output:
Optimized segmentation.
initialize set S_s to conatin all superpixels
calculate the adjacency between superpixels
while termination condition not satisfied **do**
  compute the set of boundary pixels B of superpixels in S_s;
  for each superpixel C_i **do**
    for each boundary pixel $\mathbf{p} \in B_i$ **do**
      for each neighboring pixel $\mathbf{p}_n$ of $\mathbf{p}$ that are not in C_i **do**
        get the superpixel C_j which $\mathbf{p}_n$ belongs to;
        get the cost of swapping $\mathbf{p}$ from C_i to C_j;
        store the minimum cost as ΔF_{min};
      end for
      if $\Delta F_{min} < 0$ **then**
        store the swapping item $(C_i, C_j, \mathbf{p}, \Delta F_{min})$ in the heap P_{swap} with ΔF_{min} as the key;
      end if
    end for
  end for
  clear S_s
  for each swapping item $(C_i, C_j, \mathbf{p}, \Delta F_{min})$ in P_{swap} **do**
    swap $\mathbf{p}$ from C_i to C_j;
    add C_i, C_j and their neighboring superpixels to S_s;
  end for
  update the adjacency between superpixels and clear P_{swap}
end while

after merging especially in the background area. Taking the merging result as an initial segmentation, the swapping algorithm minimizes the energy in Eq. (1) by swapping the boundary pixels between superpixels. Alg. 3 demonstrates how to optimize the partition with swapping operation. It is an iterative process, and we can set the maximum number of iterations or terminate the iteration when the energy reduction ratio is below a given threshold. First we will detect boundary pixels for a superpixel C_i. We treat a pixel $\mathbf{p}$ in C_i as the boundary pixel if one of its four neighbors does not belong to C_i. For a boundary pixel $\mathbf{p}$, it may have more than one neighboring superpixels. The swapping method calculates the cost of swapping $\mathbf{p}$ to its neighboring superpixels and chooses the optimal swapping strategy, which means swapping $\mathbf{p}$ to the superpixel with the minimun swapping cost. Suppose we consider swapping a boundary pixel $\mathbf{p}$ from C_i to C_j, i.e., $(C_i, C_j) \rightarrow (C_i^{'}, C_j^{'})$, and $C_i^{'} = C_i - \mathbf{p}$, $C_j^{'} = C_j + \mathbf{p}$. We calculate the energy change by formula $\Delta F_o = F(C_i^{'}) + F(C_j^{'}) - F(C_i) - F(C_j)$. If the value of ΔF_o is negative, then the energy will be reduced and the pixel can be swapped. The formulas to compute $F(C_i^{'})$ and $F(C_j^{'})$ are also straightforward. If we treat the pixel $\mathbf{p}$ as a superpixel, the generation of $C_j^{'}$ is a merging operation between C_j and $\mathbf{p}$. Thus we can use the above mentioned merging formulation. The generation of $C_i^{'}$ is actually a subtraction: $C_i - \mathbf{p} \rightarrow C_i^{'}$. In this case, we need to calculate the average value and covariance matrix of superpixel $C_i^{'}$ by:

$$\bar{\mathbf{p}}(C_i^{'}) = \frac{|C_i|\bar{\mathbf{p}}(C_i) - \mathbf{p}}{|C_i| - 1}, \tag{7}$$

and

$$\begin{aligned}\mathbf{M}(C_i^{'}) =& \mathbf{M}(C_i) - |C_i^{'}|(\bar{\mathbf{p}}(C_i^{'}) - \bar{\mathbf{p}}(C_i))(\bar{\mathbf{p}}(C_i^{'}) - \bar{\mathbf{p}}(C_i))^{\mathrm{T}} \\ & - (\mathbf{p} - \bar{\mathbf{p}}(C_i))(\mathbf{p} - \bar{\mathbf{p}}(C_i))^{\mathrm{T}}.\end{aligned} \tag{8}$$

Our swapping method does not ensure superpixel connectivity, i.e., during the swapping of pixels a superpixel may be divided into several disconnected sub-clusters. We keep the sub-cluster with largest number of pixels, calculate the merging cost of other small sub-clusters with their neighboring superpixels, and assign these sub-clusters to the least-cost neighboring superpixel. After such post-processing the superpixels are guaranteed to be connected.

Fig. 2 illustrates results of merging and swapping. It is clear to see that the swapping method significantly improves the spatial compactness and boundary smoothness. To quantitatively demonstrate the improvement of compactness, we use the compactness metric CO [SFS14]:

$$CO = \sum_{c \in C} Q_c \cdot \frac{|c|}{|I|}, \tag{9}$$

where $Q_c = \frac{4\pi A_c}{L_c^2}$. A_c and L_c are the area and perimeter of superpixel, $|c|$ and $|I|$ are the number of pixels in superpixel c and image I. $Q_c = 1$ if the shape of superpixel is circular, $Q_c (< 1)$ increases for more compact shape, so does the CO value. In Fig. 2 the CO value of merging result is 0.29, and it increases to 0.43 after swapping.

We quantitatively demonstrate the performance of the swapping algorithm in Fig. 3. The explanation of boundary recall, achievable segmentation accuracy, and under-segmentation error metrics

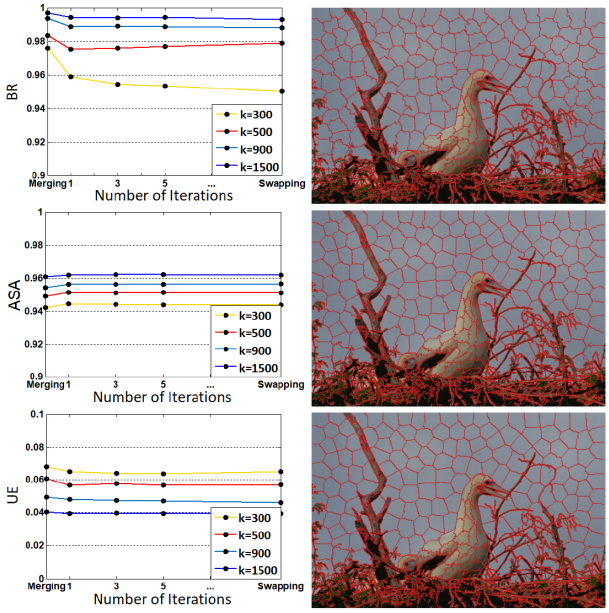


Figure 3: Performance of swapping iterations. The first column shows performance of BR, ASA and UE of results after merging, 1 iteration , 3 iterations, 5 iterations of swapping and the final result after swapping. The second column shows results after 1 ,3 and 5 iterations of swapping with k=500.

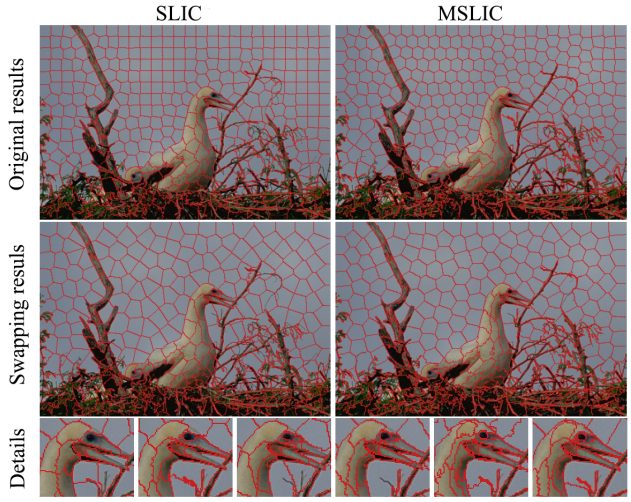


Figure 4: Taking SLIC and MSLIC results as input for the swapping method to optimize. The last row shows details of original and swapping results of SLIC, MSLIC and ours, respectively. The results after merging and swapping of our method are shown in Fig.2. We show the performance of (BR, UE, ASA): SLIC:(0.903,0.085,0.926),(0.963,0.076,0.941); MSLIC:(0.942,0.059,0.936),(0.974,0.059,0.949); ours:(0.984,0.060,0.949),(0.975,0.058,0.950).

is given in Sec. 4.1. We can see that the BR performance may decrease slightly during iterations, as the superpixels becoming more

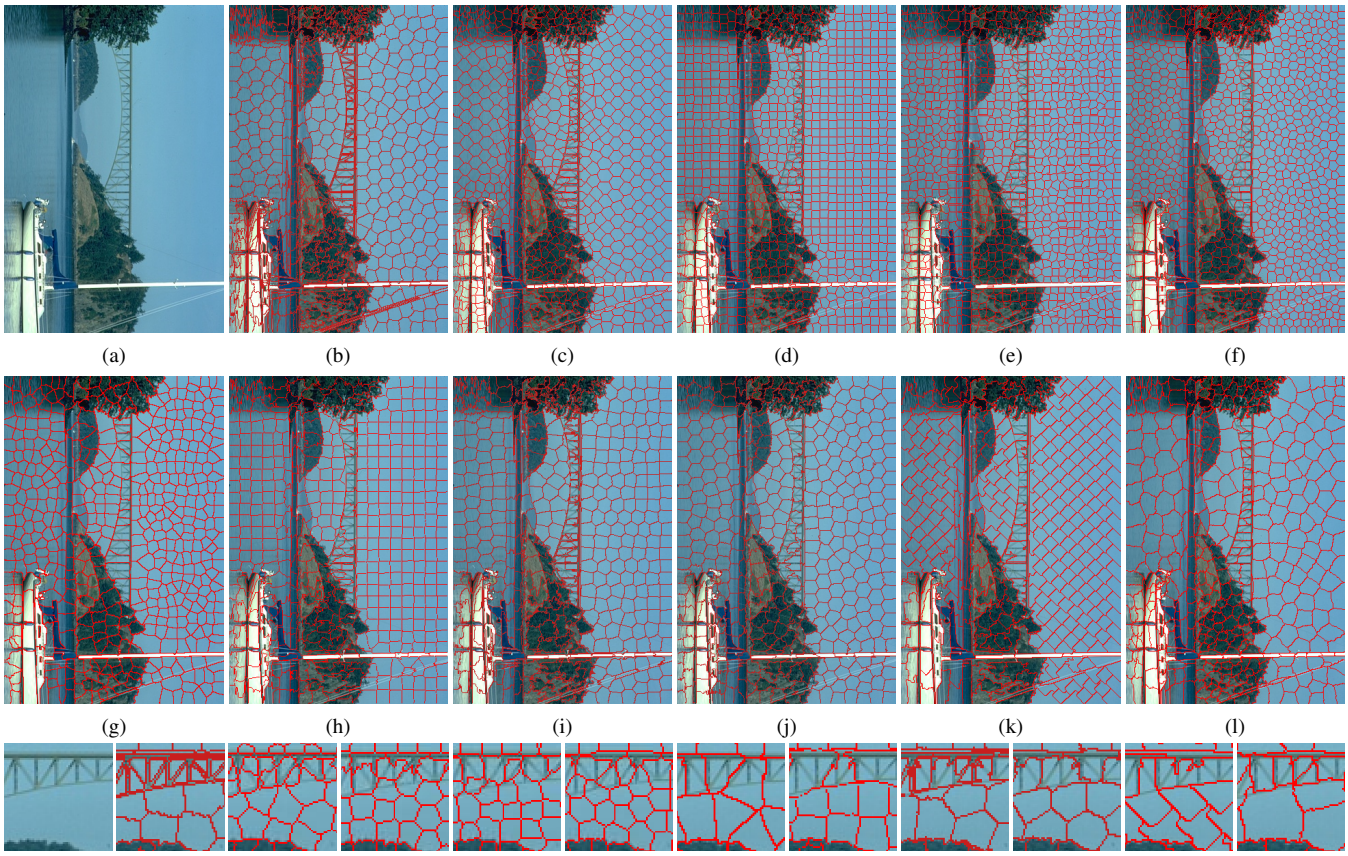


Figure 5: Superpixel generated by different methods with 500 clusters. (a) Original image; (b) our method; (c) BGD [ZPW*16]; (d) SLIC [ASS*12]; (e) Turbopixels [LSK*09]; (f) VCells [WW12]; (g) LRW [SDWL14]; (h) MSLIC [LYYH16]; (i) LSC [LC15]; (j) GGM [CG16]; (k) DBSCAN [SHL*16]; (l) IMSLIC [LYLH18]. The last row shows the magnified details in sequence.

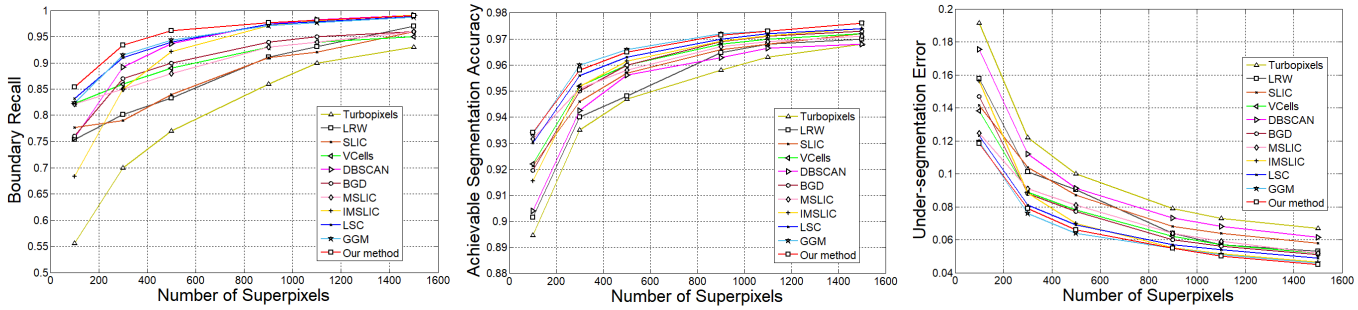
compact it allows more color variation in a region, and the boundary adherence may decrease slightly. As for ASA and UE, the performance does not change much. In addition, the more number of superpixels achieves the better performance.

In our optimization framework, the merging method generates a partition with k superpixels. Based on this initial partition, swapping method optimizes energy function by swapping boundary pixels between superpixels. We can acctually replace merging result with other segmentation and do the swapping to minimize the energy, such as taking SLIC or MSLIC results as initial segmentation. Obviously, a good initial segmentation is vital to the final result, and we believe that our merging method can provide better segmentation than other methods. Since for seed-based methods, the results highly depend on the initial placement of seeds and easily get stuck in local minima. While in the proposed merging method, each step of bottom-up clustering choose the pairs of clusters with leaset energy increase in a greedy pattern, thus eliminates the dependency of optimization results on initial seeds position. In Fig. 2, we demonstrate the partition of merging method and swapping method with k=500. In Fig. 4, the left column shows partition generated by SLIC method and MSLIC method. We take SLIC and MSLIC results as initial partition for swapping, and show the corresponding optimized partition in right column. The last row shows the magnified details of initial partition and optimized partition of SLIC, MSLIC and our method, respectively. The results show that our optimization algorithm with simply defined CVT energy outperforms others.

3.3. Implementation Details

As mentioned in Eq. (1), the parameter λ balances the relative importance of color homogeneity and spatial compactness. In the merging step, we start with setting λ to be an empirical value $\lambda = 0.1$. Later in the swapping progress, λ will be updated automatically by setting the two objectives equally important, i.e., $\lambda F_{spatial}(P) = F_{color}(P)$. If we want to obtain more compact superpixels, we can emphasize the importance of spatial compactness by setting λ to a value bigger than the computed value. The swapping method is an iterative process. In practice, it generates a good partition of an image by setting the maximum iteration number to be 100 or setting the energy reduction ratio to be 0.01% as the termination condition.

In Sec. 4.2 we extend our algorithm to RGB-D images of NYU-



(a) (b) (c)

Figure 6: Comparisons with Turbopixels [LSK*09], LRW [SDWL14], SLIC [ASS*12], VCells [WW12], DBSCAN [SHL*16], BGD [ZPW*16], MSLIC [LYYH16], IMSLIC [LYLH18], LSC [LC15] and GGM [CG16] on the BSDS500 benchmark. (a) Boundary recall; (b) achievable segmentation accuracy; and (c) under-segmentation error.

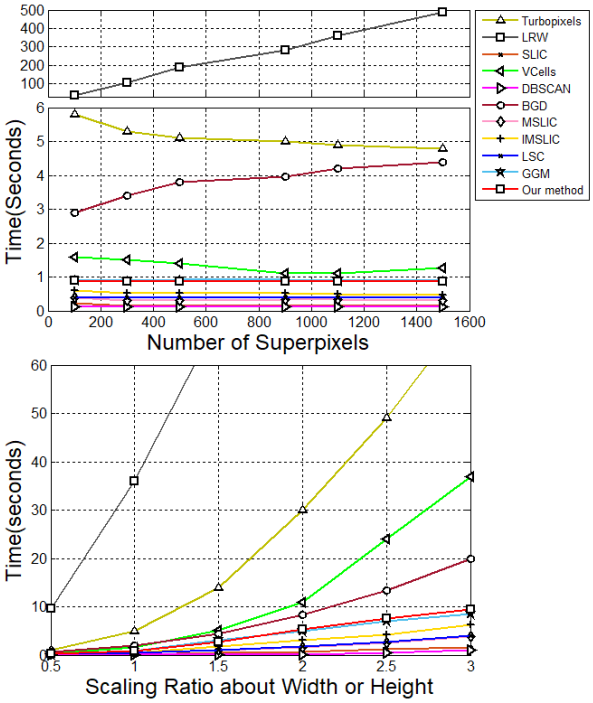


Figure 7: Running time comparison with different algorithms on BSDS500. The first shows runtime with respect to the superpixel numbers; The second shows runtime with respect to the scaling ratio.

V2 dataset, which includes extra depth information. In practice, by incorporating depth information into the spatial coordinates, we extend the dimension of the spatial covariant matrix $\mathbf{M}_s(C)$ to 3×3. Experiments show that our algorithm can detect the boundaries of objects in indoor scenes very well by using the depth information.

4. Experiments

We implemented our algorithm in C++ and tested it on a PC with an Intel i7- 4790 CPU (3.60GHz) and 16GB RAM. Experiments were performed on two datasets: the Berkeley Segmentation Dataset (BSDS500) [MFTM01] and the NYU Depth Dataset (NYUV2) [SHKF12]. BSDS500 dataset consists of 500 images, human segmented ground truth and benchmark to test performance. The resolution of the image is 481×321 (321×481). NYUV2 dataset consists of 1449 indoor images with resolution of 640×480. For each image, a depth image is provided. Since the raw images of NYUV2 contain white frames, we further crop the images to a resolution of 608×448.

4.1. BSDS500 Results

In this section, we compare our algorithm with other superpixel generation methods on the BSDS500 dataset. The performance of superpixels is commonly measured by three standard metrics: boundary recall (BR), under-segmentation error (UE) and achievable segmentation accuracy (ASA). For UE, the lower the better, and for BR and ASA, the higher the better. Among these standard metrics, the BR [LSK*09] measures the fraction of the ground truth boundaries correctly overlapped by the superpixel boundaries. A boundary pixel is regarded to be detected if it falls within 2 pixels from one point of superpixel boundaries. A high value of BR means good performance of boundary adherence. The UE [LSK*09] measures the percentage of pixels exceeding the ground truth boundaries when the superpixels are mapped on. It is actually a penalization strategy when the superpixels overlap with multiple objects. The ASA [WZG*13] is a performance upper-bound measure defined as the highest achievable object segmentation accuracy. The superpixels are labeled with the ground truth segments of the largest overlapping area.

In Fig. 5 and Fig. 6, we show the comparison result of various algorithms including BGD [ZPW*16], SLIC [ASS*12], Turbopixels [LSK*09], VCells [WW12], LRW [SDWL14], MSLIC [LYYH16], LSC [LC15], GGM [CG16], DBSCAN [SHL*16],and IMSLIC [LYLH18]. We can see that our method is better than most algorithms in terms of boundary adherence. Many algorithms are

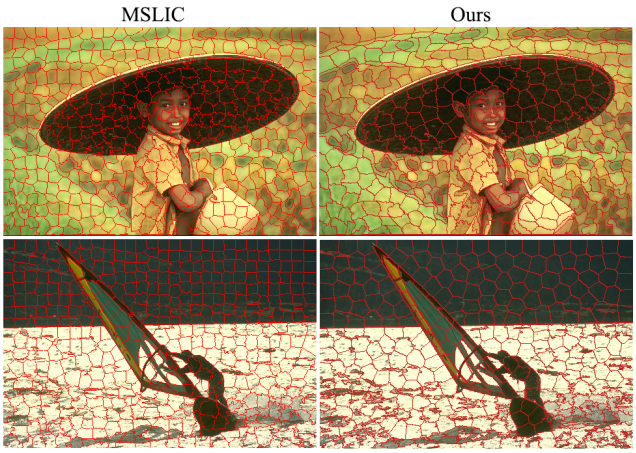


Figure 8: Comparison with the MSLIC method [LYYH16]. We show the performance of (k, BR, UE, ASA): first row, MSLIC: (505,0.910, 0.046, 0.931), Ours: (505,0.958, 0.028, 0.955); second row, MSLIC: (486,0.975, 0.019, 0.973), Ours: (486,0.987, 0.013, 0.980).

not able to detect boundaries that have pixels of similar color on opposite sides. In Fig. 6 we illustrate the performance of BR, ASA and UE, our method has the best performance on BR. It is very sensitive to capture the pixel color changes, and in the vast majority of cases it is able to detect the object boundaries. Except for our new method, the LSC and GGM methods exhibit high performance: the BR value of our method is 2 percentage higher than GGM method. In terms of UE, our method is comparable to the GGM method with the increase of the number of superpixels. For ASA, our method achieves a little bit lower value than the GGM method, it is about 0.2 percentage lower when the number of superpixels is 300 or 500. The GGM algorithm has an excellent performance on three metrics – it adopts a Gaussian Generative Model to assure the color homogeneity and the results of this method present good quality on both spatial and color aspects. The SLIC algorithm is a special case of K-means adapted to the task of generating superpixels, which localizes pixel search to a limited area. The DBSCAN algorithm also limites the pixel search in a local region, using the density-based spatial clustering of applications with noise algorithm instead. Both these algorithms are super efficient. The boundary adherence of SLIC is less competitive. For runtime performance showed in Fig. 7, our algorithm is comparable to other algorithms. For the scaling ratio, a ratio of 2 means that the image size is 4 times of the original.

In order to further demonstrate the differences between our algorithm and the other methods, we compare our method with three algorithms: MSLIC, GGM and LSC. For each image in these comparisons, we give the values of BR, UE, and ASA. As shown in Fig. 8, the MSLIC method cannot produce the exact number of superpixels specified by the user. It is clear that our algorithm outperforms MSLIC in terms of boundary adherence. In Fig. 9 we illustrate the results of the GGM method in the first column. The GGM algorithm can maintain regular shapes and uniform sizes of superpixels, and it does not allow relatively small or elongated shapes

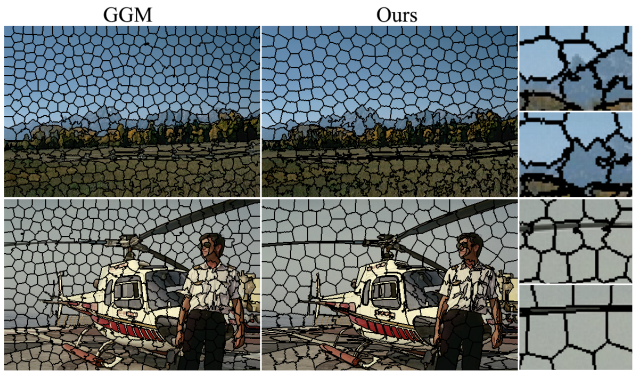


Figure 9: Comparison with the GGM method [CG16], k=500. The performance of (BR, UE, ASA): first row, GGM: (0.934, 0.057, 0.959), ours: (0.954, 0.053, 0.957); second row, GGM: (0.964, 0.059, 0.946), ours: (0.991, 0.056, 0.958).

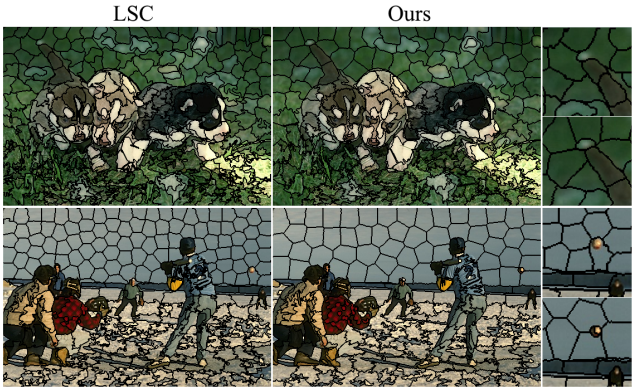


Figure 10: Comparison with the LSC method [LC15], k=300. The performance of (BR, UE, ASA): first row, LSC: (0.957, 0.035, 0.940), ours: (0.957, 0.034, 0.940); second row, LSC: (0.968, 0.076, 0.949), ours: (0.980, 0.055, 0.961).

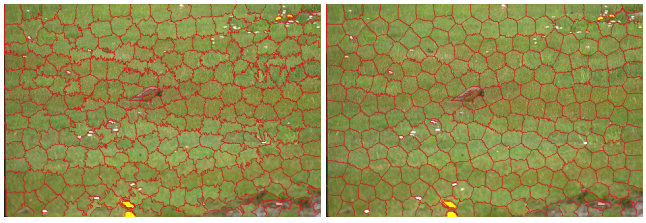


Figure 11: A bad case of our method, k=300. The left column shows zigzag boundaries with a automatically computed λ in swapping; the right column shows smooth boundaries with λ 4 times larger than the automatically computed value.

to appear. Thus in the second row, in the wing area of the plane, boundaries can not be detected well. Besides, GGM does not detect the weak boundaries well. Compared with the LSC method [LC15], which is a combined method of weighted K-means and normalized cuts by introducing a ten dimensional feature space, Fig. 10 shows the results. According to the experiments, these two algorithms generate similar results. The LSC algorithm performs well

on boundary adherence. However in some cases, the LSC has difficulty in detecting small objects, even when the color is obviously different with its adjacent regions.

Our algorithm can achieve better boundary adherence than other algorithms in most cases. One drawback of our algorithm is that it is very sensitive to color changes, generating irregular shapes and zigzag boundaries at some cases. The parameter λ which balances the importance of spatial compactness and color homogeneity, is automaticlly computed during swapping iterations and performs well in most cases. For an image with severe color changes, the value of λ needs to be larger to improve the compactness. Like the result shown in the right column of Fig. 11, we set λ to be 4 times larger than the automatically computed value.

4.2. NYUV2 Results

In Fig. 13, we test on an indoor image and its depth image from the NYUV2 dataset. Combining the depth information with the coordinates of the 2-dimensional image, the matrix $\mathbf{M}_s(C)$ becomes a 3×3 covariance matrix. We implement some existing superpixel algorithms by combining the depth information with the spatial coordinates as well, such as SLIC, MSLIC and GGM. As shown in Fig. 12 and Fig. 13, the other algorithms only operate with coordinate infomation. In Fig. 12, for three performance measurements our algorithm outperforms the others. Fig. 13 demonstrates details of superpixel results. It is clear that our algorithm makes good use of the depth information of the image, showing higher performance in boundary recall.

4.3. Application

Superpixel segmentation can be used as preprocessing step for many computer vision tasks. In this paper, we apply superpixel segmentation in saliency detection [WSS15, WSYP18]. A recent method for saliency optimization from robust background detection [ZLWS14] uses superpixel segmentation as input to generate salient object detection results. We randomly selected 500 images from the MSRA [LYS*11] dataset to test the performance. Fig. 14 shows results of saliency maps using different superpixel results. The greater gray value indicates more important area. The mean absolute error (MAE) means the average per-pixel difference between a saliency map and the ground truth, normalized to [0,1]. We compute the standard precision recall (PR) curves in Fig. 15. They are computed by comparing the saliency map to the ground truth. The high recall value means that more salient regions labeled by the ground truth are considered, and the high precision value means less pixels on the background are detected. In order to see the differences more clearly, we show the magnified details. In most cases, our method shows high precision.

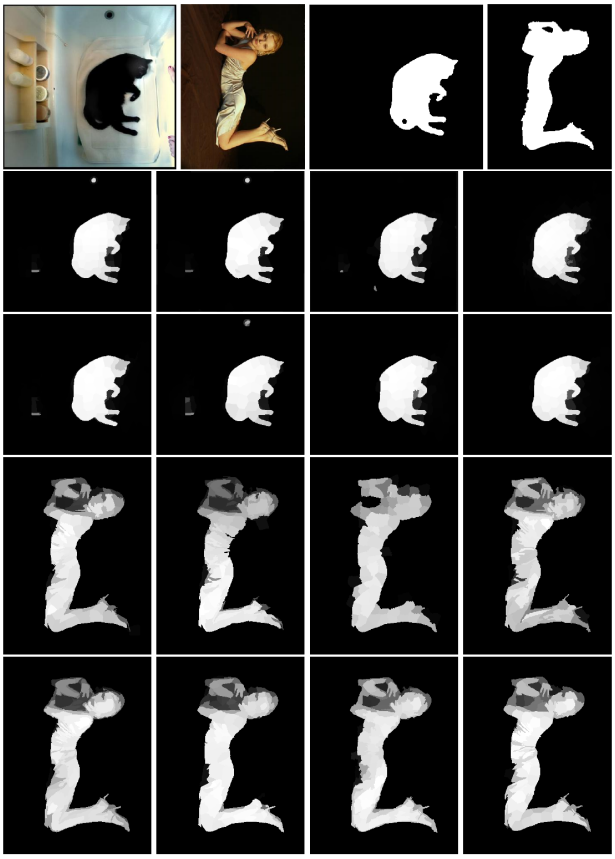
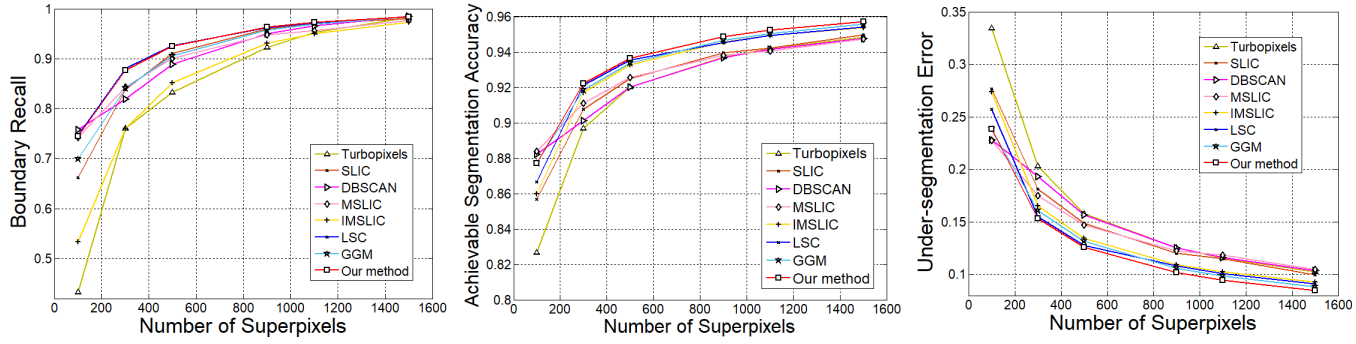


Figure 14: Saliency detection using different superpixels. The first row shows images and ground truth. From left to right, the saliency results are obtained based on the superpixel results generated by: SLIC; MSLIC; Turbopixels; DBSCAN; LSC; IMSLIC; GGM; ours. In sequence, the MAE values of the first image: 0.011, 0.011, 0.014, 0.014, 0.011, 0.013, 0.012, 0.010; the MAE values of the second: 0.066, 0.067, 0.067, 0.071, 0.066, 0.055, 0.057, 0.055.

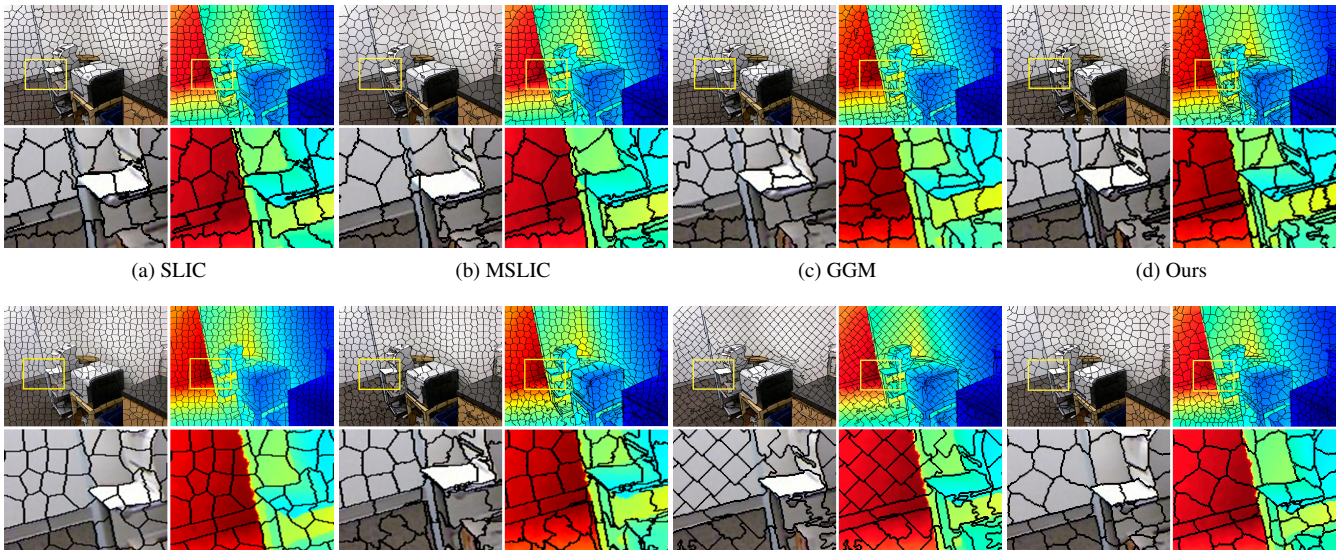
5. Discussion

In this paper, we present an agglomerative clustering algorithm based on a quadric error metric which produces compact superpixels and automatically adapts to local image contents. In contrast to seed-based methods [ASS*12, WW12, LYYH16], our method does not need a good initial seed distribution, thus avoids bad local minima to which the previous methods may get stuck. This is achieved by removing the dependency of the objective function on seed points, and using a greedy merging operation which is capable of generating sufficiently good results for subsequent optimization. The optimization scheme in this paper follows the method proposed by GGM [CG16], but for a different objective function. In GGM [CG16], they measure the spatial compactness and color homogeneity by using Mahalanobis distance, and the resultant superpixels show anisotropy in both shape and color. In other words, their method allows slightly larger color variation in a superpixel. In contrast, our algorithm is more sensitive to the changes of color and can capture small color variation. Our method tends to generate rather small and irregular shaped superpixels in complex regions with boundaries and features, leading to better boundary adherence. In the future, we consider utilizing deep learning features [WS18] as prior knowledge to regularize superpixel generation, which is also likely to get better results.



(a) (b) (c)

Figure 12: Comparisons with Turbopixels [LSK*09], SLIC [ASS*12], DBSCAN [SHL*16], MSLIC [LYYH16], IMSLIC [LYLH18], LSC [LC15] and GGM [CG16] on the NYUV2 dataset. (a) Boundary recall; (b) achievable segmentation accuracy; and (c) under-segmentation error.



(a) SLIC (b) MSLIC (c) GGM (d) Ours

(e) Turbopixels (f) LSC (g) DBSCAN (h) IMSLIC

Figure 13: Superpixel results of eight algorithms on RGBD images. Our algorithm shows high performance on boundary recall.

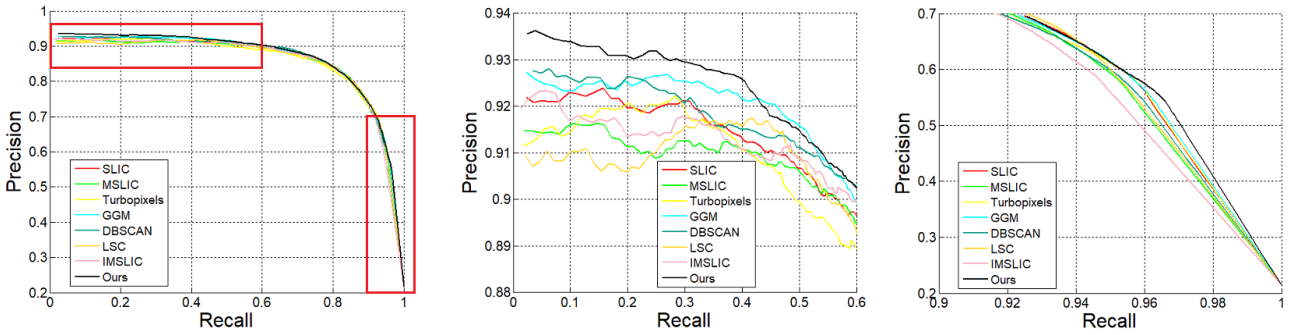


Figure 15: Precision recall curves of saliency detection using different superpixel algorithms. The middle and right figures show magnified details of the regions outlined in red in the left figure.

References

[ASS*12] 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 (2012), 2274–2282. 1, 2, 6, 7, 9, 10

[CG16] CAI Y., GUO X.: Anisotropic superpixel generation based on mahalanobis distance. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 199–207. 2, 3, 6, 7, 8, 9, 10

[CM02] COMANICIU D., MEER P.: Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence 24*, 5 (2002), 603–619. 2

[DSSVG16] DONG X., SHEN J., SHAO L., VAN GOOL L.: Sub-markov random walk for image segmentation. *IEEE Transactions on Image Processing 25*, 2 (2016), 516–527. 1

[EOK07] ERIKSSON A. P., OLSSON C., KAHL F.: Normalized cuts revisited: A reformulation for segmentation with linear grouping constraints. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on* (2007), IEEE, pp. 1–8. 2

[FH04] FELZENSZWALB P. F., HUTTENLOCHER D. P.: Efficient graph-based image segmentation. *International journal of computer vision 59*, 2 (2004), 167–181. 2

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), ACM Press/Addison-Wesley Publishing Co., pp. 209–216. 3

[HEH05] HOIEM D., EFROS A. A., HEBERT M.: Automatic photo pop-up. *ACM transactions on graphics (TOG) 24*, 3 (2005), 577–584. 1

[HZR06] HE X., ZEMEL R. S., RAY D.: Learning and incorporating top-down cues in image segmentation. In *European conference on computer vision* (2006), Springer, pp. 338–351. 1

[LC15] LI Z., CHEN J.: Superpixel segmentation using linear spectral clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 1356–1363. 2, 3, 6, 7, 8, 10

[Llo82] LLOYD S.: Least squares quantization in pcm. *IEEE Transactions on Information Theory 28* (1982), 129–137. 1

[LSD*16] LIANG Y., SHEN J., DONG X., SUN H., LI X.: Video supervoxels using partially absorbing random walks. *IEEE Transactions on Circuits and Systems for Video Technology 26*, 5 (2016), 928–938. 1

[LSK*09] LEVINSHTEIN A., STERE A., KUTULAKOS K. N., FLEET D. J., DICKINSON S. J., SIDDIQI K.: Turbopixels: Fast superpixels using geometric flows. *IEEE transactions on pattern analysis and machine intelligence 31*, 12 (2009), 2290–2297. 2, 6, 7, 10

[LTRC11] LIU M.-Y., TUZEL O., RAMALINGAM S., CHELLAPPA R.: Entropy rate superpixel segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on* (2011), IEEE, pp. 2097–2104. 2

[LWC12] LI Z., WU X.-M., CHANG S.-F.: Segmentation using superpixels: A bipartite graph partitioning approach. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (2012), IEEE, pp. 789–796. 1

[LYLH18] LIU Y.-J., YU M., LI B.-J., HE Y.: Intrinsic manifold slic: a simple and efficient method for computing content-sensitive superpixels. *IEEE transactions on pattern analysis and machine intelligence 40*, 3 (2018), 653–666. 2, 6, 7, 10

[LYS*11] LIU T., YUAN Z., SUN J., WANG J., ZHENG N., TANG X., SHUM H.-Y.: Learning to detect a salient object. *IEEE Transactions on Pattern analysis and machine intelligence 33*, 2 (2011), 353–367. 9

[LYYH16] LIU Y.-J., YU C.-C., YU M.-J., HE Y.: Manifold slic: A fast method to compute content-sensitive superpixels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 651–659. 1, 2, 6, 7, 8, 9, 10

[MFTM01] MARTIN D., FOWLKES C., TAL D., MALIK J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on* (2001), vol. 2, IEEE, pp. 416–423. 7

[MPW*08] MOORE A. P., PRINCE S. J., WARRELL J., MOHAMMED U., JONES G.: Superpixel lattices. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (2008), IEEE, pp. 1–8. 2

[MPW10] MOORE A. P., PRINCE S. J., WARRELL J.: âĂIJlattice cutâĂİ-constructing superpixels using layer constraints. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (2010), IEEE, pp. 2117–2124. 2

[NC08] NWOGU I., CORSO J. J.: 2: Beyond pairwise belief propagation labeling by approximating kikuchi free energies. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (2008), IEEE, pp. 1–8. 1

[PKPH12] PERAZZI F., KRÄHENBÜHL P., PRITCH Y., HORNUNG A.: Saliency filters: Contrast based filtering for salient region detection. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (2012), IEEE, pp. 733–740. 1

[PSYL16] PENG J., SHEN J., YAO A., LI X.: Superpixel optimization using higher order energy. *IEEE Transactions on Circuits and Systems for Video Technology 26*, 5 (2016), 917–927. 2

[PZLZ17] PAN X., ZHOU Y., LI F., ZHANG C.: Superpixels of rgbd images for indoor scenes based on weighted geodesic driven metric. *IEEE transactions on visualization and computer graphics 23*, 10 (2017), 2342–2356. 2

[PZZ11] PENG B., ZHANG L., ZHANG D.: Automatic image segmentation by dynamic region merging. *IEEE Transactions on image processing 20*, 12 (2011), 3592–3605. 1

[RM03] REN X., MALIK J.: Learning a classification model for segmentation. In *ICCV* (2003), vol. 1, pp. 10–17. 1

[SDWL14] SHEN J., DU Y., WANG W., LI X.: Lazy random walks for superpixel segmentation. *IEEE Transactions on Image Processing 23*, 4 (2014), 1451–1462. 2, 6, 7

[SFS14] SCHICK A., FISCHER M., STIEFELHAGEN R.: An evaluation of the compactness of superpixels. *Pattern Recognition Letters 43* (2014), 71–80. 5

[SHKF12] SILBERMAN N., HOIEM D., KOHLI P., FERGUS R.: Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision* (2012), Springer, pp. 746–760. 7

[SHL*16] SHEN J., HAO X., LIANG Z., LIU Y., WANG W., SHAO L.: Real-time superpixel segmentation by dbscan clustering algorithm. *IEEE Transactions on Image Processing 25*, 12 (2016), 5933–5942. 2, 6, 7, 10

[SM00] SHI J., MALIK J.: Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence 22*, 8 (2000), 888–905. 2

[SPD*17] SHEN J., PENG J., DONG X., SHAO L., PORIKLI F.: Higher order energies for image segmentation. *IEEE Transactions on Image Processing 26*, 10 (2017), 4911–4922. 1

[VBM10] VEKSLER O., BOYKOV Y., MEHRANI P.: Superpixels and supervoxels in an energy optimization framework. In *European conference on Computer vision* (2010), Springer, pp. 211–224. 2

[VdBBR*12] VAN DEN BERGH M., BOIX X., ROIG G., DE CAPITANI B., VAN GOOL L.: Seeds: Superpixels extracted via energy-driven sampling. In *European conference on computer vision* (2012), Springer, pp. 13–26. 2

[VS91] VINCENT L., SOILLE P.: Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE transactions on pattern analysis and machine intelligence 13*, 6 (1991), 583–598. 2

[VS08] VEDALDI A., SOATTO S.: Quick shift and kernel methods for mode seeking. In *European Conference on Computer Vision* (2008), Springer, pp. 705–718. 2

[WJW09] WANG J., JU L., WANG X.: An edge-weighted centroidal voronoi tessellation model for image segmentation. *IEEE Transactions on Image Processing 18*, 8 (2009), 1844–1858. 2

[WL93] WU Z., LEAHY R.: An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE transactions on pattern analysis and machine intelligence 15*, 11 (1993), 1101–1113. 2

[WLYY11] WANG S., LU H., YANG F., YANG M.-H.: Superpixel tracking. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (2011), IEEE, pp. 1323–1330. 1

[WS18] WANG W., SHEN J.: Deep visual attention prediction. *IEEE Trans. Image Process 27*, 5 (2018), 2368–2378. 9

[WSS15] WANG W., SHEN J., SHAO L.: Consistent video saliency using local gradient flow optimization and global refinement. *IEEE Transactions on Image Processing 24*, 11 (2015), 4185–4196. 9

[WSS18] WANG W., SHEN J., SHAO L.: Video salient object detection via fully convolutional networks. *IEEE Transactions on Image Processing 27*, 1 (2018), 38–49. 1

[WSYP18] WANG W., SHEN J., YANG R., PORIKLI F.: Saliency-aware video object segmentation. *IEEE transactions on pattern analysis and machine intelligence 40*, 1 (2018), 20–33. 9

[WW12] WANG J., WANG X.: Vcells: Simple and efficient superpixels using edge-weighted centroidal voronoi tessellations. *IEEE Transactions on pattern analysis and machine intelligence 34*, 6 (2012), 1241–1247. 2, 6, 7, 9

[WZG*13] WANG P., ZENG G., GAN R., WANG J., ZHA H.: Structure-sensitive superpixels via geodesic distance. *International journal of computer vision 103*, 1 (2013), 1–21. 2, 3, 7

[XLS09] XU L., LI W., SCHUURMANS D.: Fast normalized cut with linear constraints. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (2009), IEEE, pp. 2866–2873. 2

[ZHMB11] ZHANG Y., HARTLEY R., MASHFORD J., BURN S.: Superpixels via pseudo-boolean optimization. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (2011), IEEE, pp. 1387–1394. 2

[ZLWS14] ZHU W., LIANG S., WEI Y., SUN J.: Saliency optimization from robust background detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014), pp. 2814–2821. 9

[ZMZZ17] ZHANG Y., MA L., ZHOU Y., ZHANG C.: Automatic superpixel generation algorithm based on a quadric error metric in 3d space. *Signal, Image and Video Processing 11*, 3 (2017), 471–478. 3

[ZPW*16] ZHOU Y., PAN X., WANG W., YIN Y., ZHANG C.: Superpixels by bilateral geodesic distance. *IEEE Transactions on Circuits and Systems for Video Technology* (2016). 2, 3, 6, 7