STM JOURNALS

Journal of Artificial Intelligence Research & Advances
ISSN: 2395-6720 (Online)
Volume 4, Issue 3
www.stmjournals.com

Superpixel Segmentation using DBSCAN Clustering Algorithm

Partha Ghosh[1,*], Kalyani Mali[2], Sitansu K. Das[3]

[1]Department of Computer Science and Engineering, Government College of Engineering and Ceramic Technology, Kolkata, West Bengal, India

[2]Department of Computer Science and Engineering, University of Kalyani, Nadia, West Bengal, India

[3]Department of Computer Science, Chittaranjan College, Kolkata, West Bengal, India

Abstract

Superpixel division strategies are generally utilized as a pre-preparing venture to accelerate picture handling tasks. They assemble the pixels of a picture into homogeneous locales while attempting to regard existing shapes. In this paper, we propose a quick superpixels division calculation with contour adherence utilizing the density based spatial clustering of applications with noise (DBSCAN) algorithm. We implement a two-stage processing to decrease the computational costs of superpixel algorithms. First is clustering stage and the second is merging stage. At first, the DBSCAN algorithm with color similarity and geometric restrictions are used to cluster the pixels quickly, and then, small clusters are merged into superpixels by their neighborhood. It is done using a distance, which is defined by color and spatial features. We have defined a robust and simple distance function to obtain better superpixels. The experimental results show that our proposal outperforms the state-of-the-art superpixel segmentation methods in terms of both accuracy and efficiency.

Keywords: *Superpixels, DBSCAN, segmentation, under segmentation error, boundary recall*

****Author for Correspondence*** E-mail: parth_ghos@rediffmail.com

INTRODUCTION

Image segmentation is an essential tool to analyze the image content. The aim is to split the image into similar regions with respect to some priors (e.g., object, color or texture). To decrease the computational time and to improve the accuracy of unsupervised segmentation, superpixel segmentation methods have been proposed. Superpixel segmentation is popular image pre-processing technique used in many computer vision applications such as body model estimation [1], multi-class segmentation [2], depth estimation [3], object localization [4], optical flow [5], and tracking [6]. They provided an alternative to avoid the struggle with image semantics when using traditional segmentation algorithms [7, 8].

Superpixel provides more consistent representation of image by grouping pixels (which perceptually belongs together one) that adhere well to object boundaries and contains less redundancy. What differentiates superpixel algorithms from traditional segmentation algorithms is their ability to generate roughly equally-sized clusters of pixels.

Existing System

From different superpixel segmentation algorithms, we can understand that superpixel segmentation generally needed the following properties [8–12]:

(1) Superpixels should adhere well to the natural image boundaries and each superpixel should not overlap with multiple objects.
(2) As a preprocessing technique for improving efficiency of computer vision tasks, superpixel segmentation should be of low complexity itself.
(3) Global image information for human vision perception should be considered appropriately.

But considering global relationship among pixels generally lead to increases in computational complexity. A typical example is the eigen-based result to the normalized cuts

(Ncuts) based superpixel segmentation algorithm [9].

So, maximum superpixel segmentation algorithms are mainly based on the investigation of local image information only [10–12]. These algorithms offer no explicit control over the size and number of the superpixels and compactness is not considered. Superpixels thus produced are usually of irregular sizes and shapes and tend to overlap with multiple objects. So, these methods may fail to correctly segment image regions with high intensity variability [8].

Proposed System

To address this issue, we propose a superpixel segmentation algorithm by Density-Based Spatial Clustering of Applications with Noise (DBSCAN) to accomplish preferred execution over best in class strategies [12]. Among the superpixel algorithms, the SLIC algorithm ends up plainly well known, as it can create superpixels rapidly without losing a significant part of the division accuracy. Since the superpixels are utilized as a preprocessing venture in vision applications, the algorithm of top notch superpixels with less calculation is favored.

Our proposed superpixel method using DBSCAN clustering, inherits the advantages of existing superpixel algorithms (such as SLIC) and further carries forward beyond these advantages [12]. Our DBSCAN superpixel segmentation algorithm is not only more efficient but also more accurate than previous superpixel algorithms.

DBSCAN is a density-based clustering algorithm [12]. Since DBSCAN can find clusters of arbitrary shape, it has a good possibility to segment objects of complex and irregular shape. But it is necessary to introduce additional geometric boundaries on the DBSCAN clustering algorithm to yield regular superpixles. The regularity is also an important criterion for evaluating the performance of superpixels. The geometric boundaries obtain the regular shape by restricting the searching paths and searching range. The main two steps are:

Step 1: Our proposed superpixel algorithm is based on the DBSCAN clustering, with reduced computational cost. Here pixels are clustered using only color and geometric restrictions.

Step 2: Then we merge the small initial superpixels with their neighbor superpixels through a measurement of color and spatial information.

In step1, our distance function contains two items: the seed distance item and the neighbor distance item. The seed distance item ensures that the pixels contained in the same superpixel should be similar. The neighbor distance item has more influence in weak boundary and flat regions.

Our proposed method has good performance in adherence of boundaries, even for complex and irregular objects in images that state-of-the-art superpixel algorithms cannot handle (Figure 1).

The rest of this paper is organized as follows: Next part of the paper surveys the related background work i.e., review of existing methodologies for superpixel segmentation, followed by the proposed DBSCAN algorithm method. Experimental results are demonstrated after that and lastly we conclude our work.

Fig. 1: *Images Segmented into 250/450/950 Superpixels Using the Proposed DBSCAN Algorithm in the Regions Separated by Two White Parallel Lines [13].*

RELATED WORKS

There are two types of superpixel segmentation methods:

(1) The first class is based on clustering, including the normalized cuts [14], SLIC [15], LSC, SEEDS [16] and Turbo [17].
(2) The other class is based on optimization, such as graph cuts [18], lattice cut [19], entropy rate (ERS) [20], pseudo-boolean optimization (PB) [12] and lazy random walk (LRW) [2].

In previous work, algorithms designed for image segmentation were directly used for generating superpixels, such as efficient graph-based image segmentation [8], mean shift [10] and quick shift [10]. In the work by Felzenszwalb and Huttenlocher, each superpixel is denoted by a minimum spanning tree and two superpixels are combined if the maximum weight of edges inside the trees is larger than the minimum weight of edges connecting them [8]. Mean shift and quick shift are two mode seeking techniques attempting to maximize a density function by shifting pixels near areas of higher density. Pixels converging to the same mode mount a superpixel. These algorithms have no explicit control over the size and number of the superpixels and compactness is not considered. So, superpixels produced are generally of irregular sizes and shapes and have a tendency to overlap with multiple objects.

However, in normalized cuts formulation, the old eigen-based solution, of high computational complexity, which further grows when the number of eigen vectors to be computed increases [21]. Due to its heuristic nature, Ncuts is less effective compared to other approaches when the number of superpixels increases.

Aforementioned algorithms which do not consider the spatial compactness generally lead to under segmentation, particularly when there is poor contrast or shadow [11]. Among the algorithms stated above, Ncuts is the only one that indirectly takes compactness into consideration [9]. However, its application is restricted due to its high computational complexity.

To solve this problem, Turbopixel algorithm was proposed, which produces highly uniform lattice-like superpixels by iteratively stretching regularly distributed seeds [11]. However, due to the consistancy and efficiency issues of the level-set method, superpixels thus generated present relatively low adherence to boundaries and the algorithm is slow in practice.

Moore *et al.* proposed an algorithm for lattice cut-constructing superpixels using layer constraints that reserves the topology of a regular lattice in superpixel segmentation [22, 23].

Bergh *et al.* suggested SEEDS by introducing an energy function that inspires color homogeneity and shape regularity [14]. A hill-climbing algorithm is used for optimization. However, SEEDS also suffers from high shape irregularity and the superpixel number is difficult to control. Achanta *et al.* proposed a linear clustering based algorithm (SLIC) which produces superpixels by iteratively applying simple K-means clustering in the combined five-dimensional color and coordinate space [15]. In spite of its simplicity, SLIC has been proved to be effective in various computer vision applications [24]. Nevertheless, as a local feature based algorithm, the relationship between SLIC and global image properties is not clear.

Dhillon *et al.* proved that K-way normalized cuts in the original pixel space is indistinguishable to the weighted K-means clustering in a high dimensional feature space by modifying weighted k-means clustering as a trace maximization problem [25]. However, the high dimensional feature space is not clearly defined and the kernel trick has to be used [26]. The generated kernel matrix can be very large in practice. So, time and space complexity will be very high.

Li and Chen proposed LSC to obtain superpixels [26]. Due to the calculations of features, the LSC is slow as a preprocessing method. Shen *et al.* proposed a method using LRW to get superpixel segmentation results [27]. The LRW algorithm uses the probabilities of each pixel from the input image, and commute time to get initial

superpixels. Then their technique introduces an energy function to optimize iteratively the initial superpixels, which is correlated to the texture measurement and the commute time. However, the LRW superpixel algorithm is very expensive, and it usually costs several seconds to generate the final superpixels.

Liu *et al.* expressed the superpixel segmentation problem as an objective function that consists of the entropy rate (ERS) of a random walk on a graph and a balancing term which inspires the generation of superpixels with similar sizes [28]. The entropy rate can help to cluster the compact and homogeneous regions, which also favors the superpixels to overlap with a single object on the perceptual boundaries. However, the irregular shape of ERS superpixels may become a possible disadvantage in future application.

Ghosh *et al.* proposed NKSC algorithm to obtain superpixels [29]. The NKSC combines the advantages of normalized cut and k-means to get good quality superpixels.

Ghosh *et al.* used DBSCAN clustering algorithm for image segmentation [30]. Manavalan and Thangavel used DBSCAN clustering for transrectal ultrasound image segmentation [31]. The ultrasound images usually have poor image quality, such as low contrast, speckle noise, and weak boundaries, and it is very difficult to segment them by conventional clustering approaches. However, the DBSCAN algorithm is quite suitable to detect and segment most of these important regions.

PROPOSED DBSCAN SUPERPIXEL ALGORITHM

In this section, we will present the proposed DBSCAN superpixel segmentation algorithm. It will create superpixels with state of the art boundary adherence and also faster than existing well-known algorithms (LSC [26], SLIC [15] and SEED [14]).

The superpixel segmentation can be considered as a clustering problem where each superpixel contains a unique feature in color and shape. The aim of superpixels is to cluster pixels with a homogenous appearance from an image into small, compact regions. The proposed method includes two steps: a clustering step and a merging step. The DBSCAN clustering algorithm will improve the performance of segmentation algorithms by adding geometric restrictions.

Algorithm

In the clustering step, we define two sets as labeled set and candidate set, respectively, and then assign the top-left pixel a label for the first seed and add it into the labeled set. Now we have three types of pixels, i.e., the seed, labeled pixels and unlabeled ones. Firstly, we find all of the unlabeled four neighboring pixels of the labeled set, and then calculate the combination distance between each unlabeled pixel of its center pixel (the unlabeled pixel is generated by the center pixel) and the seed. Now if the distance is less than the threshold (threshold *S/N*, where *S* denotes the size of the input image, *N* stands for the number of superpixels that the user wants) we defined, we place it into the candidate set. Secondly, we update the labeled set through replacing it by the candidate set and provide them the same label with the seed. We repeat the two steps until the termination condition is satisfied.

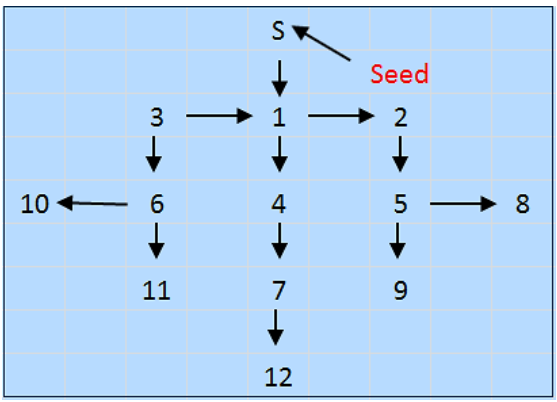

Fig. 2: *Pixel Generation.*

As shown in Figure 2, the termination conditions may arise in two ways. The first one is that the number of pixels in this cluster is more than a threshold *S/N*. This condition is used to regulate the size of the cluster. The second one is the labeled set that becomes an empty set. It implies that the neighboring pixels of the previously labeled set will locate at the borderline regions, and the algorithm will stop at the borderline for this condition.

Then we select a new seed from the unlabeled pixels and repeat the above technique until all the pixels are labeled. We choose the seeds from left to right and from top to bottom. Thus, initial superpixels are created. Here, pixels are similar in color through the distance constraint. We are able to reduce the computational complexity by modifying the searching strategy from original DBSCAN algorithm [12]. We restrict the searching range in the parallelogram neighbor region around the seed in our proposed algorithm. But the searching range of the conventional DBSCAN algorithm is the whole image as in Figure 3. By adapting this strategy in clustering stage, we are also able to make each superpixel with a uniform shape as far as possible.

L(p) is the initial superpixel which we get after clustering. In most of the cases, superpixel boundaries arrange the edges of objects very well. But in our method, the clustering stage generates rather small superpixels and fragments (very small initial superpixels) at some edges of objects. This is clear from Figure 4.

So, to remove the small fragments created in the clustering stage we merge the initial superpixels in merging stage. This is shown in Figure 4.

Fragments are generated due to the usage of distance between pixels, which is sensitive to the local color features.

To eliminate the fragments, we may adopt two policies:

- One is to add a small fragment into its neighbor to form a larger superpixel, and
- The other is to merge two neighbor fragments.

Here, we introduce a superpixel distance to regulate the merging strategy. This distance combines color distance and spatial distance.

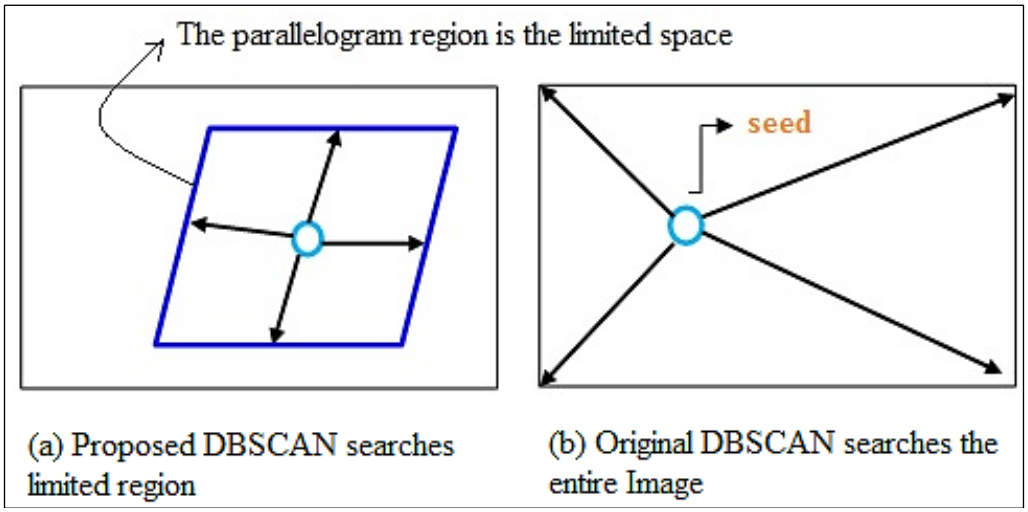

Fig. 3: *Clustering Stages.*

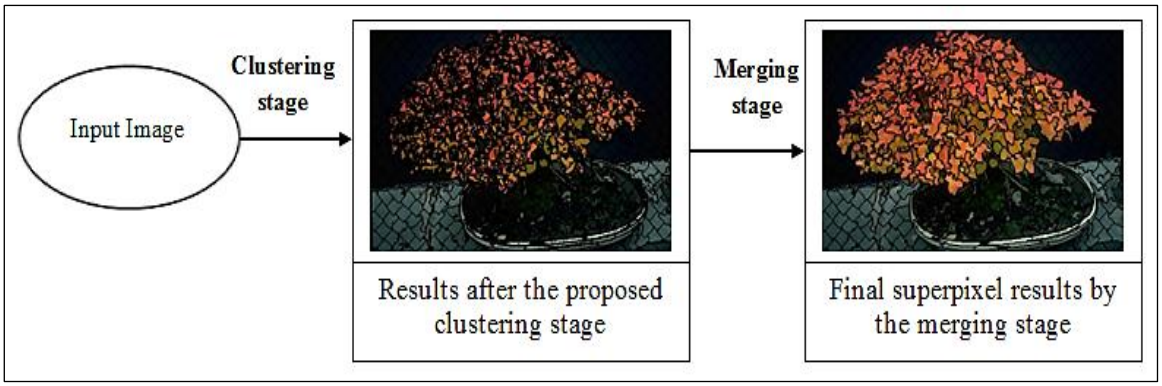

Fig. 4: *Result of Clustering Stages.*

In our DBSCAN, superpixel segmentation clustering is done by comparing the distance between pixels in the RGB color space. The distance is an important parameter; we define two distance metric functions in clustering and merging stages. In the clustering stage, we generate a linear combination function $D_1^k(i,j)$and in the merging stage, the distance function $D_2$is applied to calculate the distance between the initial superpixels.

Clustering Stage

$D_1^k(i,j)$is calculated by integrating two Euclidean metric functions $d_n(i,j)$and$d_s^k(i,k)$, where$d_n(i,j)$ is the distance between two adjacent pixels in RGB color space and $d_s^k(i,k)$is the distance between an unlabeled pixel i and a seed k. As $d_n(i,j)$is simple but not robust enough especially in the weak boundaries, we improve this boundary discrimination ability by adding $d_s^k(i,k)$and enhancing the superpixel internal consistency. The distance$D_1^k(i,j)$is defined as:

$$d_n(i,j) = \sqrt{(R_i - R_j)^2 + (G_i - G_j)^2 + (B_i - B_j)^2}$$

$$d_s^k(i,k) = \sqrt{(R_i - R_k)^2 + (G_i - G_k)^2 + (B_i - B_k)^2}$$

$$D_1^k(i,j) = \alpha_1 d_s^k(i,k) + \alpha_2 d_n(i,j) \qquad (1)$$

Where, R_i, G_i, $B_i$represent the RGB color features of pixel i, α_1, α_2 are the weight parameters, and$\alpha_1 + \alpha_2 = 1$. Now to produce the initial superpixels during the clustering stage we will use Eq. (2):

$$L(p) = \{i \in I \mid D_1^p(i,j) < \mu, i \sim j, j \in p\} \cup L(p) \quad (2)$$

Where, p is a superpixel, $\mu$is a threshold (distance constraint), $i \sim j$indicates pixel i around pixel j, j is in labeled set L(p), and$D_1^p(i,j)$is the distance between pixels i and j.

Merging Stage

Function D_2 combines color distance and spatial distance. In our method, the color difference is used to determine whether two initial superpixels should be merged into one or not, and the spatial distance assures the regularity of final superpixels. Thus, the merging distance $D_2$is defined as:

$$d_c(l,p) = \sqrt{(\widetilde{R_l} - \widetilde{R_p})^2 + (\widetilde{G_l} - \widetilde{G_p})^2 + (\widetilde{B_l} - \widetilde{B_p})^2}$$

$$d_a(l,p) = \sqrt{(\widetilde{x_l} - \widetilde{x_p})^2 + (\widetilde{y_l} - \widetilde{y_p})^2}$$

$$D_2(l,p) = d_c(l,p) + \alpha_3 d_a(l,p) \qquad (3)$$

Where,$d_c(l,p)$represents the color distance between two initial superpixels l and p, and ensures a high degree of similarity, $d_a(l,p)$is the spatial distance that has a large influence on the shape of final superpixel results. $\widetilde{R_p}$, $\widetilde{G_p}$, and $\widetilde{B_p}$are the average RGB color values for initial superpixel p, $\widetilde{x_p}$and $\widetilde{y_p}$are the coordinates of the centroid in superpixel p. The parameter $\alpha_3$in Eq. (3) is a positive coefficient for balancing the relative influence between $d_c(l,p)$and $d_a(l,p)$. We use Eq. (4) in the merging stage, where$D_2(p,i)$ is the distance between initial superpixels p and i.

$$L(i) = argmin\, D_2(p,i), i \sim p$$

$$L_R(p) = L(p) \cup L(i) \ldots\ldots\ldots\ldots \qquad (4)$$

In the merging stage, all the initial superpixels are processed from left to right and from top to bottom. If the number of pixels in one initial superpixel is less than a threshold, we will merge another initial superpixel from its shortest distance neighbor into this one. After the merging stage, we can get the final refined superpixels with the regular shapes. The entire procedure is précised in Algorithm 1 and Algorithm 2.

Algorithm 1: DBSCAN Superpixel Segmentation

Input: Image I, superpixel p. threshold μ, labeled set L_{set} and candidate set C_{set};
Output: Initial superpixel label L(p);

/* Initialization */
Step 1: Set initial pixel label is 0 for each image in I;
/* Assignment */
Step 2: while each pixel have a new label do
Step 3: find a seed k;
Step 4: set $k \in L_{set}$;
Step 5: while L_{set} is empty or the number of pixels in p is larger than the threshold $\mu = S/N$do

Step 6: for each pixel j in L_{set}do
Step 7: for each pixel i around pixel j do
Step 8: Compute the clustering distance $D_1^k(i,j)$with seed k and j.
Step 9: if$D_1^k(i,j) < \mu$ then
Step 10: set $i \in C_{set}$;
Step 11: end if
Step 12: end for
Step 13: end for
Step 14: set $L_{set} = C_{set}$;
Step 15: end while
Step 16: end while

Algorithm 2: Fine-tuning by Merging Optimization
Input: Initial superpixel label L(p);
Output: Refined superpixel label L_R(p);

Step 1: Set distance d(p)=100000 for each superpixel p.
Step 2: if the number of pixels in p is less than S/N then
Step 3: for each superpixel *l* around *p* do
Step 4: compute the merging distance $D_2(l,p)$ between *p* and *l*;
Step 5: if $D_2(l,p) < d(p)$ then
Step 6: set $d(p) = D_2(l,p)$;
Step 7: set label *j*=*l*;
Step 8: end if
Step 9: end for
Step 10: merge two superpixels between j and p;
Step 11: end if

EXPERIMENTS: COMPARISON WITH STATE-OF-THE-ART

We evaluate the performance of the proposed DBSCAN algorithm superpixel algorithm by comparing with state-of-the-art algorithms, including SLIC [15], PB [17], Ncuts [21], Tubopixel [11], ERS [28], SEEDS [14], LRW [27] and LSC [26]. All the experiments are performed on the Berkeley Segmentation Database (BSD) [19], which consists of five hundred 321×481 images, together with human-annotated ground truth segmentations. We will demonstrate the effectiveness of the proposed method by first giving the visual comparison results of those methods and then providing a detailed quantitative comparison. To obtain a stable good performance we set the default parameters: $\alpha_1 = 0.6, \alpha_2 = 0.4, \alpha_3 = 1.0\ and\ \mu = 30$.

Visual Comparison
Figure 5 gives the representative visual superpixel results generated by SLIC [15], PB [17], LSC [26], ERS [28] and our proposed DBSCAN method. Some detail segmentation results are emphasized to facilitate close visual inspection. It is obvious that our method obtains a better performance of image edges adherence than the other methods. Our method has achieved the most perceptually satisfactory segmentation results for different kinds of images. It happens due to the fact that our algorithm uses color information to identify the image boundaries more precisely than other algorithms.

Our superpixel algorithm also produces compact and regular superpixels through the proposed parallelogram geometric shape constraints.

Quantitative Comparison
One of the most important requirements for superpixels is to maintain its adherence of object boundaries. Therefore, we adopt three commonly used evaluation metrics in superpixel segmentation for measuring the quality of boundary adherence: under-segmentation error (UE), boundary recall (BR) and achievable segmentation accuracy (ASA). Boundary recall (BR) is an important metric for measuring the performance of adherence of boundaries in superpixel algorithms. It measures what fraction of the ground truth edges falls within at least two pixels of a superpixel boundary. A high BR means that very few true boundaries are missed. Achievable segmentation accuracy (ASA) computes the highest achievable accuracy of labeling each superpixel with the label of ground truth that has the biggest overlap area. ASA is calculated as the fraction of labeled pixels that are not leaking from the ground truth boundaries. A high ASA means that the superpixels obey well with objects in the image.

Under-segmentation error (UE) measures the percentage of pixels that leak from the ground truth boundaries. A good superpixel algorithm should try to avoid the under segmentation areas in the segmentation results. In other words, we need to protect that a superpixel only overlaps with one object. A lower UE indicates that fewer superpixels cross multiple objects.

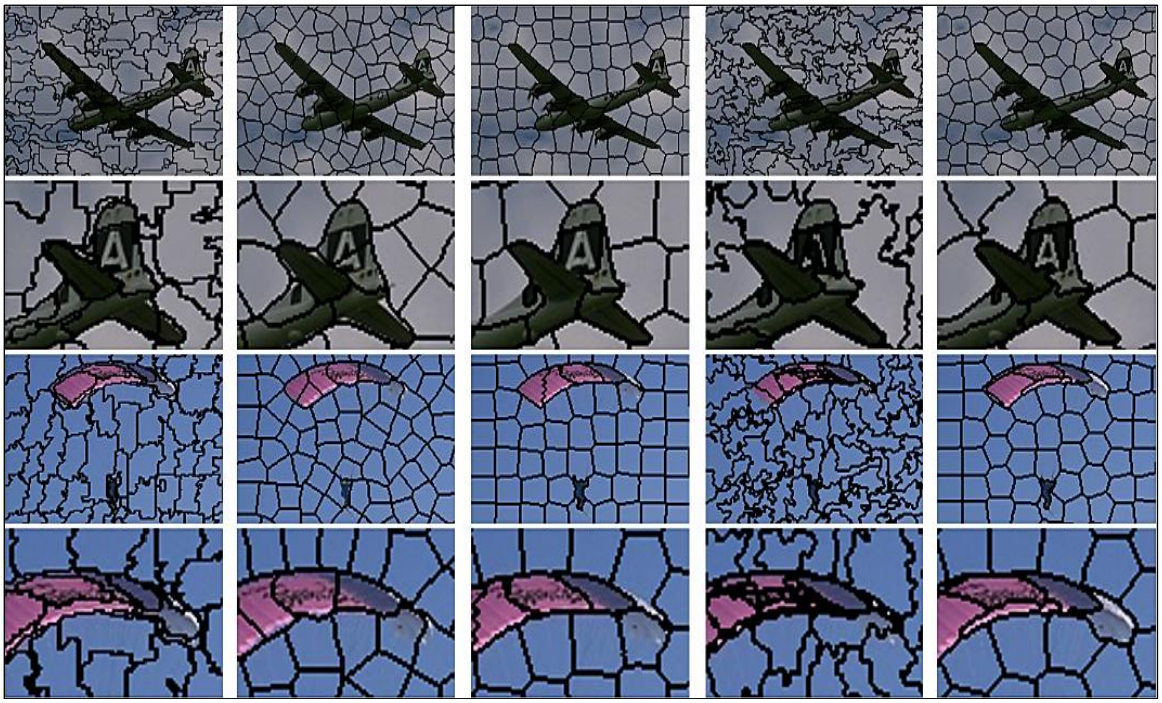

Fig. 5: *Visual Comparison of Superpixel Segmentation Results (From Left to Right) by PB, SLIC, LSC, NKSC and Our Proposed DBSCAN Method. The Number of Superpixels is about 500. In Each Group, the Top Row is the Superpixel Results Generated by Each Algorithm and the Bottom Row is the Magnified Regions.*

As shown in Figure 6(b), it is apparent that our superpixel DBSCAN algorithm outperforms the other nine algorithms from the lower superpixel densities to the higher superpixel densities for the BR measurement. Turbopixel gives the worst performance compared with other algorithms, and LSC and our previous NKSC method produce similar performance as our present DBSCAN method.

Figure 6(c) plots the average ASA result values of 500 images against the number of superpixels, and our method outperforms the other ten algorithms.

As shown in Figure 6(a), the UE curves are the average values for all 500 images in BSD. With the increase of superpixel numbers, our DBSCAN method demonstrates better performance.

As shown in Figure 6, Tubopixel shows the worst performance in terms of boundary adherence among all tested algorithms. ERS and SEEDS have similar boundary adherence performance with NKSC by sacrificing the regularity and perceptual satisfaction, and SEEDS was the fastest prevailing superpixel segmentation algorithm. Now, our proposed DBSCAN algorithm is the fastest superpixel segmentation algorithm.

ANALYSIS OF COMPUTATIONAL COMPLEXITY AND DISCUSSIONS

The computational efficiency is also an important factor for evaluating the performance of superpixel segmentation algorithms. We will analyze the algorithm complexity and compare the computational costs. We perform all the experiments on a desktop PC equipped with an Intel 8-core 3.4 GHz processor with 4 GB RAM. In our experiment, we calculate the average running time for different algorithms and the results are shown in Table 1 and Figure 6(d). The time taken by the Ncuts algorithm is much higher than that of the other methods.

The complexity of our DBSCAN superpixel method is *O(N)* without an iteration process, which is faster than all the state-of-the-art superpixel algorithms. Though the complexities of SLIC, LSC and our previous NKSC algorithms are also *O(N)*, their core algorithm requires many iterations.

Table 1: *Performance Metrics of Superpixel Segmentation Algorithms as the Superpixel Number is 400.*

	SLIC	PB	SEEDS	ERS	LSC	NKSC	Our Proposed DBSCAN
Adherence to Boundaries							
Under segmentation error (UE)	0.21	0.654	0.197	0.198	0.191	0.190	**0.189**
Boundary recall (BR)	0.83	0.724	0.918	0.920	0.925	**0.926**	0.924
Achievable segmentation accuracy (ASA)	0.956	0.927	0.960	0.959	0.962	0.961	**0.964**
Segmentation Speed							
Computational complexity	O(N)	O(N)	O(N)	$O(N^2 \log N)$	O(N)	O(N)	O(N)
Average time per image (sec)	0.314	0.50	**0.213**	1.88	0.919	0.918	**0.019**

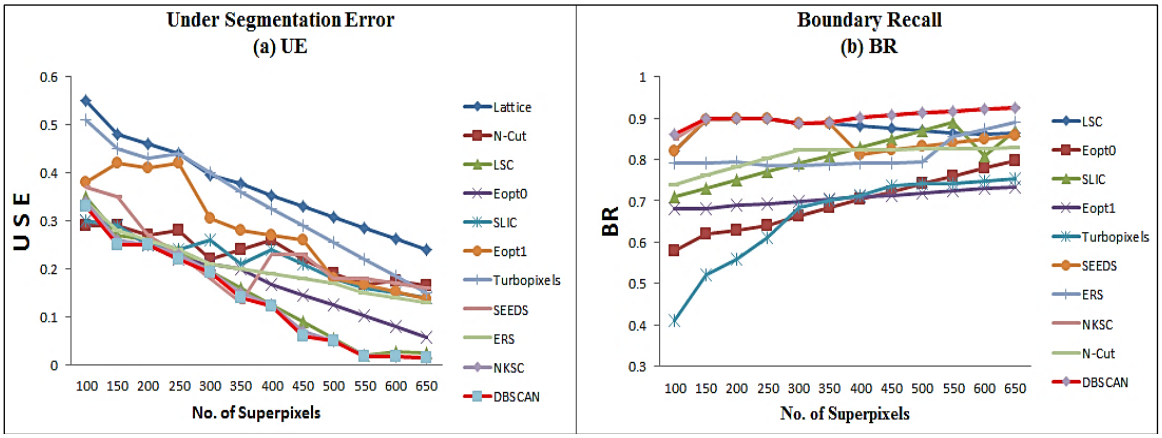

Fig. 6: (a) *Under Segmentation Error,* ***(b)*** *Boundary Recall.*

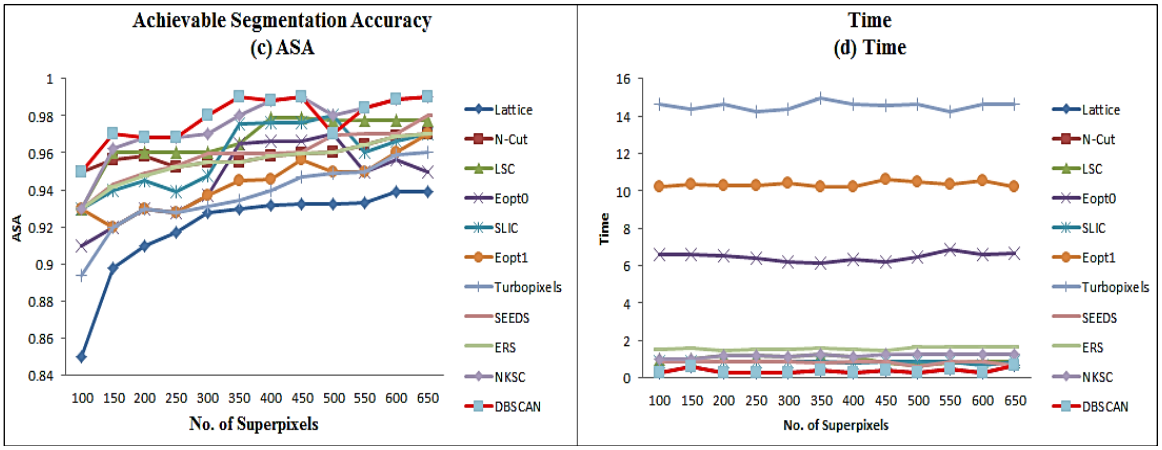

Fig. 6: (c) *Achievable Segmentation Accuracy,* ***(d)*** *Time.*

CONCLUSIONS

We present in this paper a novel superpixel segmentation algorithm, DBSCAN, which produces compact and regular shaped superpixels with linear time complexity. Our algorithm achieves the state-of-the-art performance at a significantly smaller computation cost, and considerably outperforms the algorithms that require more computational costs for the images with complex objects or complex texture regions.

REFERENCES

1. Mori G. Guiding Model Search Using Segmentation. In *IEEE International Conference on Computer Vision (ICCV)*. 2005.
2. Gould S, Rodgers J, Cohen D, *et al.* Multi-Class Segmentation with Relative Location Prior. *International Journal of Computer Vision (IJCV)*. 2008; 80(3): 300–316p.

3. Zitnick CL, Kang SB. Stereo for Image-Based Rendering Using Image Over-Segmentation. *International Journal of Computer Vision (IJCV)*. Oct 2007; 75: 49–65p.
4. Fulkerson B, Vedaldi A, Soatto S. Class Segmentation and Object Localization with Superpixel Neighborhoods. In *International Conference on Computer Vision (ICCV)*. 2009.
5. Menze M, Geiger A. Object Scene Flow for Autonomous Vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.
6. Wang S, Lu H, Yang F, *et al*. Superpixel Tracking. In *IEEE International Conference on Computer Vision (ICCV)*. Nov 2011.
7. Comaniciu D, Meer P. Mean Shift: A Robust Approach toward Feature Space Analysis. *IEEE Trans Pattern Anal Mach Intell*. May 2002; 24(5): 603–619p.
8. Felzenszwalb P, Huttenlocher D. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision (IJCV)*. Sep 2004; 59(2): 167–181p.
9. Ren X, Malik J. Learning a Classification Model for Segmentation. *Proc of ICCV*. 2003; 1: 10–17p.
10. Veldadi A, Soatto S. Quick Shift and Kernel Methods for Mode Seeking. *Proc of ECCV*. 2008; 705–718p.
11. Levinshtein A, Stere A, Kutulakos K, *et al*. Turbopixel: Fast Supepixels Using Geometric Flow. *IEEE Trans PAMI*. 2009; 31(12): 2209–2297p.
12. Ester M, Kriegel H-P, Sander J, *et al.* A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proc 2nd Int Conf Knowl Discovery Data Mining (KDD)*. 1996; 226–231p.
13. Martin D, Fowlkes C, Tal D, *et al.* A Database of Human Segmented Natural Images and Its Application to Evaluate Segmentation Algorithms and Measuring Ecological Statistics. *Proc of ICCV*. 2001; 2: 416–423p.
14. Bergh M, Boix X, Roig G, *et al*. Seeds: Superpixels Extracted via Energy-Driven Sampling. *Proc of ECCV*. 2012; 7578: 13–26p.
15. Achanta R, Shaji A, Smith K, *et al*. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Trans Pattern Anal Mach Intell.* Nov 2012; 34(11): 2274–2282p.
16. Ng A, Jordan M, Weiss Y. On Spectral Clustering: Analysis and an Algorithm. *Proc of NIPS*. 2001; 849–856p.
17. Zhang Y, Hartley R, Mashford J, *et al.* Superpixels via Pseudo-Boolean Optimization. In *Proc IEEE Int Conf Comput Vis*. Nov 2011; 1387–1394p.
18. Yu S, Shi J. Multiclass Spectral Clustering. *Proc of ICCV*. 2003; 1: 313–319p.
19. Martin D, Fowlkes C, Tal D, *et al.* A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In *Proc IEEE Int Conf Comput Vis*. Jul 2001; 2: 416–423p.
20. Cristianini N, Taylor J. *An Introduction to Support Vector Machines and other Kernel-Based Learning Methods*. New York, NY, USA: Cambridge University Press; 2000.
21. Shi J, Malik J. Normalized Cuts and Image Segmentation. *IEEE Trans PAMI*. 2000; 22(8): 888–905p.
22. Moore A, Prince S, Warrell J. Lattice Cut Constructing Superpixels Using Layer Constraints. *Proc of CVPR*. 2010; 2117–2124p.
23. Moore A, Prince S, Warrell J, *et al.* Superpixel Lattices. *Proc of CVPR*. 2008; 1–8p.
24. Wang S, Lu H, Yang F, *et al.* Superpixel Tracking. *Proc of ICCV*. 2011; 1: 1323–1330p.
25. Dhillon I, Guan Y, Kulis B. Weighted Graph Cuts without Eigen Vectors: A Multilevel Approach. *IEEE Trans PAMI.* 2007; 29(11): 1944–1957p.
26. Li Z, Chen J. Superpixel Segmentation Using Linear Spectral Clustering. In *Proc IEEE Conf Comput Vis Pattern Recognit (CVPR).* Jun 2015; 1356–1363p.
27. Shen J, Du Y, Wang W, *et al.* Lazy Random Walks for Superpixel Segmentation. *IEEE Trans Image Process.* Apr 2014; 23(4): 1451–1462p.

28. Liu M-Y, Tuzel O, Ramalingam S, *et al.* Entropy Rate Superpixel Segmentation. In *Proc IEEE Conf Comput Vis Pattern Recognit*. Jun 2011; 2097–2104p.
29. Ghosh P, Mali K, Das SK. Superpixel Segmentation with Contour Adherence using Spectral Clustering, Combined with Normalized cuts (N-Cuts) in an Iterative k-Means Clustering Framework (NKSC). *IJEFT*. 2017; 14(3): 23–37p.
30. Ghosh P, Mali K. Image Segmentation by Grouping Pixels in Color and Image Space Simultaneously using DBSCAN Clustering Algorithm. *JoRSG*. 2013; 4(3): 52–60p.
31. Manavalan R, Thangavel K. TRUS Image Segmentation Using Morphological Operators and Dbscan Clustering. In *Proc World Congr Inf Commun Technol*. Dec 2011; 898–903p.

Cite this Article
Partha Ghosh, Kalyani Mali, Sitansu K. Das Superpixel Segmentation using DBSCAN Clustering Algorithm. *Journal of Artificial Intelligence Research & Advances*. 2017; 4(3): 26–36p.