Differential Evolutionary Superpixel Segmentation

Yue-Jiao Gong, *Member, IEEE*, and Yicong Zhou, *Senior Member, IEEE*

***Abstract*— Superpixel segmentation has been of increasing importance in many computer vision applications recently. To handle the problem, most state-of-the-art algorithms either adopt a local color variance model or a local optimization algorithm. This paper develops a new approach, named differential evolutionary superpixels, which is able to optimize the global properties of segmentation by means of a global optimizer. We design a comprehensive objective function aggregating within-superpixel error, boundary gradient, and a regularization term. Minimizing the within-superpixel error enforces the homogeneity of superpixels. In addition, the introduction of boundary gradient drives the superpixel boundaries to capture the natural image boundaries, so as to make each superpixel overlaps with a single object. The regularizer further encourages producing similarly sized superpixels that are friendly to human vision. The optimization is then accomplished by a powerful global optimizer—differential evolution. The algorithm constantly evolves the superpixels by mimicking the process of natural evolution, while using a linear complexity to the image size. Experimental results and comparisons with eleven state-of-the-art peer algorithms verify the promising performance of our algorithm.**

***Index Terms*— Superpixel segmentation, differential evolution, seeding, preprocessing, clustering.**

I. Introduction

SUPERPIXEL segmentation plays a crucial preprocessing role in many image processing and computer vision applications, such as salient object detection [1]–[3], image segmentation [4]–[10], video cosegmentation [11], dense matching [12], object tracking [13], [14], and image parsing [15], [16]. As illustrated in Fig. 1, the task of superpixel segmentation is to over-segment an image into a few compact regions that are aligned well with natural objects. Compared with the rigid pixel representation of images, a superpixel representation contains less redundancy and greatly reduces the number of image primitives. For example, for a 481×321 image, the number of units to be processed is $154,401$ by a pixel representation, while this number can be reduced to hundreds if superpixels are considered instead. Therefore, as a preprocessing step, superpixel segmentation is able provide a dimension reduction effect and a substantial speedup for the subsequent operation. In addition, the produced superpixels are perceptually meaningful regions that agree well with human perception.

Fig. 1. Illustration of superpixel segmentation. Given an input image (top), the proposed algorithm groups the pixels into a specific number of small atomic regions (bottom).

Since the concept of superpixel segmentation has been proposed [17], intensive studies have been devoted to this area, such as the bottom-up, data-driven methods in [18]–[23] and the top-down, task-driven methods in [24]–[28] (these algorithms are going to be compared in this paper). Each of these methods may favor an *ad hoc* application, but generally the following three properties are desired for a good superpixel algorithm.

1) The boundaries of superpixels should capture natural image boundaries such that each superpixel overlaps with a single natural object. The satisfaction rate of this property greatly influences the effectiveness of subsequent operation and hence acts as the primarily important goal in superpixel segmentation.
2) Since the superpixel algorithms are used as a preprocessing step, they are required to possess good time efficiency. Namely, in designing a superpixel algorithm, the computational complexity is a critical issue to be considered.
3) Superpixels with relatively regular shapes and similar sizes are commonly preferred, in order to make the resulting superpixels be friendly to human vision or the feature extraction procedure in applications.

To address these issues, this paper proposes a novel algorithm named Differential Evolutionary Superpixels (DES). Generally, many existing methods optimize the boundary

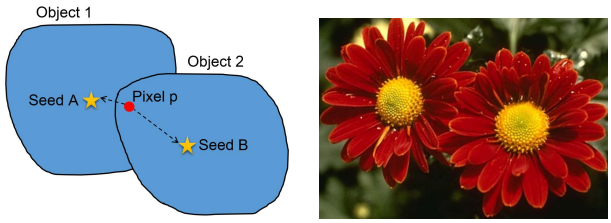
Manuscript received February 17, 2017; revised July 31, 2017, September 11, 2017, and October 11, 2017; accepted November 13, 2017. Date of publication November 29, 2017; date of current version December 27, 2017. This work was supported in part by the Macau Science and Technology Development Fund under Grant FDCT/016/2015/A1, in part by the Research Committee at University of Macau under Grant MYRG2016-00123-FST, and in part by the National Natural Science Foundation of China under Grant 61502542. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ling Shao. *(Corresponding author: Yicong Zhou.)*

Y.-J. Gong is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with the Department of Computer and Information Science, University of Macau, Macau 999078, China (e-mail: gongyuejiao@gmail.com).

Y. Zhou is with the Department of Computer and Information Science, University of Macau, Macau 999078, China (e-mail: yicongzhou@umac.mo).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2017.2778569



(a) (b)

Fig. 2. Illustration of the importance of using global segmentation properties. (a) Given two objects with identical (or similar) colors, it is very difficult to segment pixels near the boundary part (e.g., pixel p in the image) to the correct object (object 2 for p) by minimizing local color variance only. (b) A real-world case.

adherence of superpixels by minimizing local color variance, whereas DES optimizes this global property in a more straightforward way. The limitation of minimizing local variance only is illustrated in Fig. 2. When two objects in the image possess very similar colors, it becomes very difficult to segment them correctly, especially for the pixels near the boundary parts of the objects. In DES, we design a *Boundary Gradient* term to evaluate the adherence of superpixel boundaries to the high-gradient components in the image. The optimization of the boundary gradient term greatly relieves the above problem since the superpixel boundaries are now enforced to capture the natural object boundaries. For the purpose of producing similarly sized superpixels, we further introduce a *Regularizer* to measure the global variance of superpixel sizes. The two global terms, as well as the traditionally used local cost to enforce superpixel homogeneity (named *Within-Superpixel Error* in this paper), are aggregated in the objective function to optimize.

The optimization task is then accomplished by Differential Evolution (DE), a powerful stochastic global optimizer mimicking the nature evolution process [29]–[31]. In DES, the superpixel segmentations are encoded as individuals in the population and bred by the genetic operators of DE. New individuals with better objective values will replace old ones to realize the "evolution" of segmentations. Meanwhile, the algorithm maintains a kernel point set that interacts with the DE population in a form of *positive feedback* to improve the efficiency. Owing to the low complexity of DE, the proposed DES satisfies the computational restriction that it produces promising superpixels with a linear computational complexity to the image size.

In the experiments undertaken, DES is tested on the Berkeley segmentation benchmark with 500 images (BSDS 500) [32] and the PASCAL-S dataset [33]. Compared with eleven state-of-the-art superpixel segmentation algorithms, DES exhibits better or equally well performance in terms of standard performance metrics.

The major contributions of this work are summarized as follows. We accomplish the superpixel segmentation task by designing a global property model and performing a global optimizer on the model. First, in the model, we are the first to design and aggregate three components (within-superpixel error, boundary gradient, and regularizer) in the objective function. Optimizing the three components together enhances the local homogeneity, boundary adherence, and the regularity of superpixels. Second, considering the optimizer, we make the first attempt to use DE for superpixel segmentation, which not only provides accurate optimization results but also possesses low computational complexity. Third, extensive experiments and the comparisons with existing algorithms validate that DES provides a powerful and reliable tool for superpixel segmentation. Note that DES has a preliminary version published in a conference paper [28]. Later, we will compare the two versions of DES in detail.

II. Related Work

Generally, there are two categories of algorithms for superpixel segmentation: the seeding-based and graph cut-based methods. In this section, we review representative algorithms in the two categories. Besides, we present the methodology of DE at the end of this section.

A. Seeding-Based Superpixel Segmentation

Many algorithms produce superpixels by identifying a number of seeds (or centers) and then growing superpixels from the seeds. Therefore, the superpixels are data-driven and formed in a bottom-up manner.

1) Linear Clustering: Intuitively, the well-known K-means clustering algorithm can be utilized for superpixel segmentation to minimize local color variance. However, when clustering the numerous pixels of an image into superpixels, K-means endures high computational costs in calculating the distance between pixels and seeds. To solve this problem, a restricted K-means is widely used, such as in the Simple Linear Iterative Clustering (SLIC) [18] and the Linear Spectral Clustering (LSC) [19], [34]. Both SLIC and LSC restrict the pixel assignment into small windows to obtain linear complexity i.e., $O(N)$, where N is the image size. Compared with SLIC, LSC further makes an improvement by using kernel functions to realize a nomarlized cuts effect. Despite the K-means clustering, the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm is also adapted for superpixel segmentation. Shen *et al*. [20] develop a two-stage algorithm: in the first stage, the pixels are grouped into clusters using DBSCAN; and in the second stage, small clusters are merged into superpixels. The algorithm has $O(N)$ complexity and it exhibits real-time performance (the fastest algorithm in our experiment). Achanta and Susstrunk [35] propose a Simple Non-Iterative Clustering (SNIC) algorithm. As the name indicated, SNIC is non-iterative and efficient. In addition, based on SNIC and a polygonal partition method, the authors design a SNICPOLY algorithm for polygonal segmentation. Both SNIC and SNICPOLY achieve promising performance among their peer algorithms.

2) Mode Shifting: Mean Shift (MS) [36] and Quick Shift (QS) [21] are mode-seeking algorithms that iteratively shift the seeds towards maximum density areas in the image. The two algorithms are quite slow, whose computational complexity are $O(N^2)$. Besides, using MS and QS, the number of produced superpixels cannot be explicitly controlled. For this type of algorithms, in order to specify the superpixel

number, the parameters involved in the algorithms need to be tuned on the dataset. Shen *et al*. [22] propose a lazy random walk (LRW) algorithm. The algorithm initializes superpixel seeds by LRW, then iteratively shifts the seeds by energy optimization, and finally refines the seeds by executing LRW once again. As reported in [22], the computational complexity of LRW is $O(N^2)$. Apart from the superpixel segmentation in [22], the random walk method is also applied to generate video supervoxels [37].

3) Morphological Method: Another popular work is the Turbopixel (TP) method [23], which gradually dilates a set of regularly distributed seeds using geometric flows. TP poses strong constraints on the uniformness and compactness of superpixels. As a result, the algorithm can generate highly regular superpixels, but, meanwhile, the boundary adherence is relatively poor, as can be observed in Fig. 3. According to [23], the computational complexity of TP is $O(N)$. However, in real execution, the algorithm is much slower than that of the other $O(N)$ superpixel algorithms such as the SLIC [18] and LSC [19].

4) Watershed Transform: Due to the over-segmentation effect, the original Watershed Segmentation (WS) [38] algorithm is available for generating superpixels. The complexity of WS is $O(N \log N)$, but the algorithm cannot explicitly control the number of generated superpixels. Note that WS does not belong to the seeding-based superpixel category, but its variant, the Waterpixel (WP) [39], belongs to. WP first selects a number of markers and then perform watershed transformation based on the makers and the image gradient. The reported complexity is $O(N)$ [39].

B. Graph Cut-Based Superpixel Segmentation

Unlike the seeding-based methods, in this category, the superpixels are task-driven and produced in a top-down fashion. Particularly, the image is characterized by a graph of vertices and edges. Each vertex represents a pixel, while each edge connecting two vertices denotes the similarity between the two pixels. The superpixel segmentation is then performed on the graph to minimize the costs of cuts over the graph.

1) Normalized Cuts: Normalized Cuts (NC) [17], [40], [41] is the most classical algorithm in this category and is widely considered as a pioneer of superpixel segmentation. In NC, the pixel graph is partitioned based on the eigenvectors of the normalized Laplacian graph matrix, which globally minimizes the segmentation cost. However, the computational complexity of NC is $O(N^{1.5})$, while the reported real execution time is extraordinary long (a few minutes for a single image according to [19] and [26]). The large computational overhead limits the wide applicability of this algorithm.

2) Hierarchical Models: Felzenszwalb and Huttenlocher [42] propose a highly efficient graph-based over-segmentation algorithm (GS) by performing agglomerative clustering and building a minimum spanning tree. GS arrives at a good boundary adherence performance, but the resulting superpixels have irregular shapes and quite different sizes. The theoretical time complexity of GS is $O(N \log N)$. In the Hierarchical Edge-Weighted Centroidal Voronoi Tessellation (HEWCVT) [43], multiscale superpixels are produced by a multilevel segmentation process, from coarse grain to fine grain. As reported in [43], HEWCVT is $O(N)$ complex.

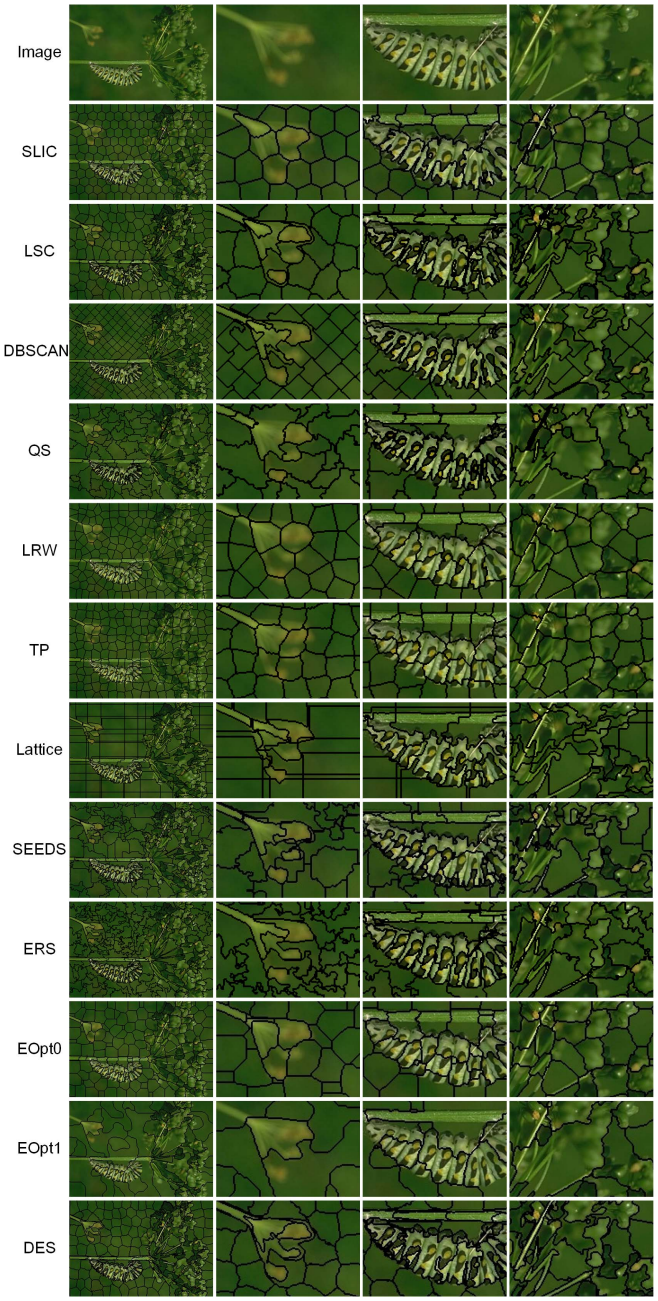


Fig. 3. Superpixels produced by different state-of-the-art algorithms. From up to bottom: image, SLIC, LSC, DBSCAN QS, LRW, TP, Lattice, SEEDS, ERS, EOpt0, EOpt1, and DES.

3) Splitting-Path Finding: Lattice [24] generates superpixels by seeking the horizontal and vertical splitting-paths on the image. The segmentation task is hence formulated as an optimal path problem and solved by *s-t min-cut* and *dynamic programming*. The algorithm targets at generating quasi-regular grid, as depicted in Fig. 3. The complexity of Lattice is $O(N^{1.5} \log N)$. Following this concept, the Superpixels Extracted via Energy-Driven Sampling (SEEDS) [25], [44] algorithm finds optimal paths on the images through *hill climbing*. The algorithm possesses very promising computational

efficiency, i.e., $O(N)$, in that only pixels near the superpixel boundaries are considered in the optimization. Nevertheless, SEEDS poses a strict limitation on the number of superpixels to be generated, and it also suffers from shape irregularity. The Entropy Rate Superpixels (ERS) [26] designs a new objective, namely, the entropy rate of the graph, and optimizes it together with a balancing term. The objective function has submodular and monotonic properties, while ERS uses a greedy algorithm to obtain approximate solutions. ERS is $O(N^2 \log N)$ complex. Besides, as can be seen in Fig. 3, the superpixels produced by ERS are very irregular.

4) Energy Optimization: Veksler *et al*. [27] formulate the superpixel segmentation problem in an energy optimization framework and develop two algorithms (EOpt0 and EOpt1) to generate compact and constant-intensity superpixels, respectively. The two algorithms provide good shape regularity at the cost of $O(N^3/K^2)$. Peng *et al*. [45] develop a higher order energy function that uses the segmentation results of MS [36] as prior. The algorithm conducts in a two-stage manner: first, the K-means clustering is applied to generate initial superpixels; and then, the superpixels are refined by optimizing the higher order energy function. Involving MS as a pre-segmentation step, the computational complexity of this algorithm is $O(N^2)$.

C. Background of Differential Evolution

DE was first developed by Storn and Price in 1997 [46] and undergone intensive studies recently [47]–[49]. The algorithm is a powerful stochastic optimizer specialized in solving nondeterministic polynomial-time (NP) hard problems. Given an optimization problem, DE mimics the process of nature evolution to search the global optimum. The basic idea is described as follows. The algorithm maintains a population that consists of P individuals. Each individual is a D-dimensional parameter vector to represent a candidate solution to the problem, where D is the dimensionality of the problem space. The objective function F of the problem acts as the role of environment to evaluate the fitness (i.e., quality) of individuals. In each generation of DE, the genetic reproduction operators such as mutation and crossover are used to breed new and possibly more competitive offspring individuals. After evaluating the fitness, the selection operator updates the population based on a rule named "Survival of Fitness (SoF)". Particularly, the individuals possessing high fitness values survive to the next generation, whereas the inferior individuals are discarded. This way, the population is going to have better and better fitness values with the increase of generation number. Namely, the solutions are improved during the evolution process. After a number of generations, the population converges to the global optimum of the problem. Finally, the algorithm outputs the optimal solution, which is represented by the individual possessing the best fitness.

As an evolutionary optimization algorithm, DE has the following strengths [29]–[31]. 1) Conceptual simplicity: the idea and procedures of the algorithm follow the natural evolution process, which are easy to understand and implement. 2) Global optimization ability: compared with some local optimizers, DE can produce globally optimal solutions of the problems. 3) Flexibility: the operators of DE do not rely on the objective formulation, so that, with small adaptation, the algorithm can be utilized to solve various practical optimization problems. 4) Efficiency: the algorithm is able to approach optimal or near-optimal solutions with low computational overhead. 5) Robustness: the algorithm is persistent under uncertain conditions. In addition, compared with the other evolutionary optimization methods (such as the genetic algorithms), DE has distinction on its mutation scheme, which automatically adapts the step length to fit the objective landscape. As reported in various competitions of optimization methods organized by the evolutionary computation community, DE and its variants always ranked the first or very front places among all the evolutionary algorithms. Therefore, DE is recognized as the most competitive evolutionary algorithm, when considering its comprehensive performance in the competitions of global optimization, multimodal optimization, multiobjective optimization, large-scale optimization, etc [29], [30].

Owing to the above benefits, DE has been widely applied to solve complex optimization problems in various fields like economic dispatch [50], network localization [51], and seismic inversion [52], etc. In the context of image processing and computer vision, DE has seen applications to image segmentation [53], [54], classification [55], [56], and human motion tracking [57]. In this paper, we make the first attempt to apply the algorithm to superpixel segmentation.

III. Differential Evolutionary Superpixels

This paper develops a novel superpixel segmentation algorithm, the DES, which formulates a global model and then adopts a global optimizer to solve the model. DES is a seeding-based algorithm, namely, given an input image I and the specified superpixel number K, the algorithm targets at finding an optimal set of K seeds to determine the superpixel segmention. Fig. 4 depicts the segmentation pipeline of DES, which is composed of of four major modules:

1) *Feature Representation:* Represent each pixel of the image by a feature vector consists of color, spatial, and contour information (Subsection A).
2) *Initialization:* According to the image size and the requested superpixel number, initialize a kernel point set and P individuals of DE, where P is the population size (Subsection B).
3) *Superpixel Iteration:* Aggregate the kernel point set and the DE individuals to obtain P candidate seed sets, use nearest-neighbor assignment to obtain the corresponding superpixel label matrices (Subsection C), evaluate the quality of the label matrices (Subsection D), and update the kernel point set (Subsection E) and DE population (Subsection F) accordingly.
4) *Postprocessing:* Enforce the connectivity of superpixels (Subsection G).

A. Feature Representation

Each pixel $\boldsymbol{p}$ in the input image is represented by a six-dimensional feature vector $[l, \alpha, \beta, x, y, \delta]^{\mathrm{T}}$, where l, α,

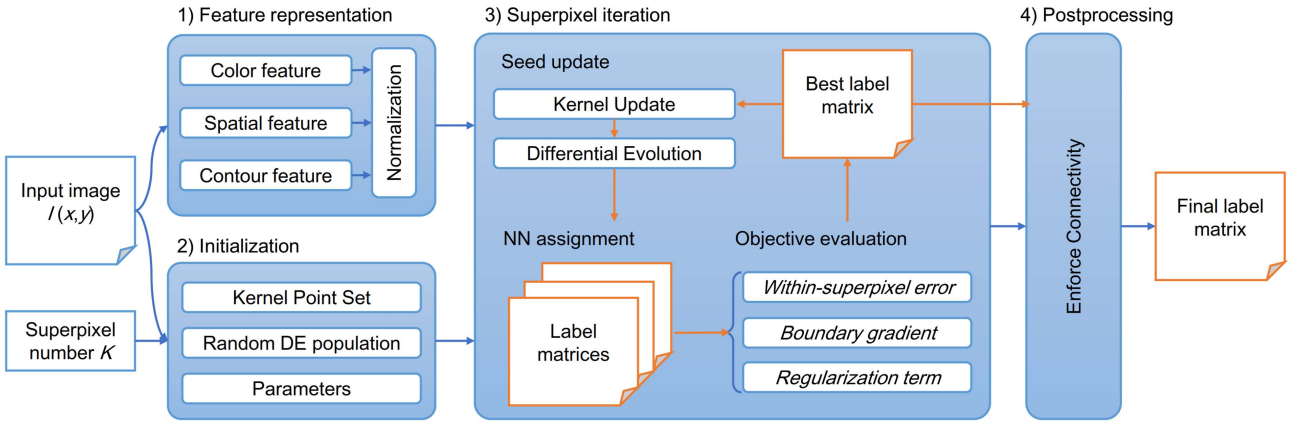


Fig. 4. The DES segmentation pipeline is composed of four processing modules: feature representation, system initialization, superpixel iteration, and postprocessing.

and β stand for light and color components in CIELAB color space (color feature), x and y denote the pixel coordinates (spatial feature), and δ denotes the pixel gradient (contour feature). Due to the critical time limitation of superpixel segmentation, only simple and low-level features are used. Then, without loss of generality, the six components are linearly normalized into range $[0, 1]$. For clarity, for the coordinates x and y, the normalized values are denoted as μ and ν in the following statement.

B. System Initialization

We need to identify a set of seeds to partition the image into K superpixels that arrive at the optimal segmentation performance. As described in Section II-C, DE is a population-based optimizer that maintains and evolves P individuals. Each individual i is a parameter vector X_i, which can be decoded to a candidate solution. For the superpixel segmention problem, each individual i should be decoded to a candidate seed set $\boldsymbol{s}_i = [\boldsymbol{s}_{i,1}, \boldsymbol{s}_{i,2}, \cdots, \boldsymbol{s}_{i,K}]$. Meanwhile, DES maintains a kernel point set $\boldsymbol{c} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_K]$ that interacts with the individuals. The interaction is bidirectional: on the one hand, each individual i retrieves the candidate seed set $\boldsymbol{s}_i$ based on the combination of its own parameter vector X_i and the kernel point set $\boldsymbol{c}$, as will be detailed in the subsection C; and on the other hand, the kernel point set $\boldsymbol{c}$ is updated based on the segmentation result of the individual achieving the best objective evaluation value, as will be detailed in the subsection E.

The kernel set $\boldsymbol{c}$ consists of K points, each is characterized by a feature vector $[l_k, \alpha_k, \beta_k, \mu_k, \nu_k, \delta_k]^{\mathrm{T}}$ and the coordinates (x_k, y_k) in the image plane ($k = 1, 2, \cdots, K$). In the initialization, the kernel points are uniformly sampled from the image. Particularly, the image is partitioned into K uniform squares of size $S \times S$, where $S = \sqrt{N/K}$. The kth initial point is sampled from the pixel located in the center (x_k, y_k) of the kth square. To reduce the risk of sampling a kernel point on an edge or noisy pixel in the image, (x_k, y_k) is moved to the position with the lowest gradient in its 3×3 neighborhood [18]. Then, retrieve the feature vector at (x_k, y_k) for $\boldsymbol{c}_k$.

The DE population consists of P individuals: X_1, $X_2, \cdots, X_P$. Each individual is characterized by a D-dimensional parameter vector:

$$X_i = [p_{i,1}, p_{i,2}, \cdots, p_{i,D}], \quad p_{i,j} \in [-S/2, S/2], \tag{1}$$

where $D = 2K$. For an integer $k \in \{1, 2, \cdots, K\}$, the pair of $(p_{i,2k-1}, p_{i,2k})$ in X_i denotes the differential lengths made to the coordinates of the kth kernel point along the vertical and horizontal directions, respectively, in the image plane. By combining the individuals and the kernel points, various candidate seed sets can be obtained, which will be described later in the next subsection. Here, in the initialization, the population of DE is randomly generated, with each $p_{i,j}$ being a random number in the range.

$$p_{i,j} = -S/2 + \mathrm{rand}_{i,j}(0, 1) \cdot S, \tag{2}$$

where $i = 1, 2, \cdots, P$, $j = 1, 2, \cdots, D$, and $\mathrm{rand}_{i,j}$ is a uniformly random number lying between 0 and 1. Note that the uniformly random initialization is commonly used in the DE literature, for the purpose of increasing the diversity of initial population [29]–[31].

C. Seeding and Nearest-Neighbor Assignment

In the seeding procedure, the P individuals are decoded to P candidate seed sets. Then, the nearest-neighbor (NN) assignment is performed on each of these candidate seed sets to obtain P candidate label matrices respectively. The procedure is described as follows.

Given an individual X_i, we can retrieve a candidate set of seeds in the following way. For each $(p_{i,2k-1}, p_{i,2k})$ in X_i, a new position in the image plane is obtained by adding the pair to the coordinates of the kth kernel points (x_k, y_k):

$$(\hat{x}_{i,k}, \hat{y}_{i,k}) = (x_k + \lfloor p_{i,2k-1} \rfloor, y_k + \lfloor p_{i,2k} \rfloor), \tag{3}$$

where $i = 1, 2, \cdots, P$ and $k = 1, 2, \cdots, K$. Then, using each $(\hat{x}_{i,k}, \hat{y}_{i,k})$ to retrieve a feature vector $\hat{\boldsymbol{c}}_{i,k} = [\hat{l}_{i,k}, \hat{\alpha}_{i,k}, \hat{\beta}_{i,k}, \hat{\mu}_{i,k}, \hat{\nu}_{i,k}, \hat{\delta}_{i,k}]^{\mathrm{T}}$, the kth candidate seed of individual i is calculated as

$$\boldsymbol{s}_{i,k} = (1 - q) \cdot \boldsymbol{c}_i \oplus q \cdot \hat{\boldsymbol{c}}_{i,k}, \tag{4}$$

where "$\oplus$" is component-wise addition, $q = 0.1$ is a constant denoting the modification rate of kernel points.

After obtaining $\boldsymbol{s}_i = [\boldsymbol{s}_{i,1}, \boldsymbol{s}_{i,2}, \cdots, \boldsymbol{s}_{i,K}]$, we can derive a superpixel label matrix L_i through the NN assignment of pixels. Namely, each pixel of the image is assigned with the index of its nearest seed in $\boldsymbol{s}_i$. The distance measure used in the proposed algorithm is defined as follows. Given two points $\boldsymbol{a} = [l_{\boldsymbol{a}}, \alpha_{\boldsymbol{a}}, \beta_{\boldsymbol{a}}, \mu_{\boldsymbol{a}}, \nu_{\boldsymbol{a}}, \delta_{\boldsymbol{a}}]^{\mathrm{T}}$ and $\boldsymbol{b} = [l_{\boldsymbol{b}}, \alpha_{\boldsymbol{b}}, \beta_{\boldsymbol{b}}, \mu_{\boldsymbol{b}}, \nu_{\boldsymbol{b}}, \delta_{\boldsymbol{b}}]^{\mathrm{T}}$ in the feature space, the distance between the two points is

$$d(\boldsymbol{a}, \boldsymbol{b}) = \sqrt{d_{\mathrm{c}}^2(\boldsymbol{a}, \boldsymbol{b}) + w_{\mathrm{s}} \cdot d_{\mathrm{s}}^2(\boldsymbol{a}, \boldsymbol{b}) + w_{\mathrm{g}} \cdot d_{\mathrm{g}}^2(\boldsymbol{a}, \boldsymbol{b})}, \quad (5)$$

where d_{c}, d_{s}, and d_{g} are the color, spatial, and contour distance components calculated as

$$\begin{aligned} d_{\mathrm{c}}(\boldsymbol{a}, \boldsymbol{b}) &= \sqrt{(l_{\boldsymbol{a}} - l_{\boldsymbol{b}})^2 + (\alpha_{\boldsymbol{a}} - \alpha_{\boldsymbol{b}})^2 + (\beta_{\boldsymbol{a}} - \beta_{\boldsymbol{b}})^2}, \\ d_{\mathrm{s}}(\boldsymbol{a}, \boldsymbol{b}) &= \sqrt{(\mu_{\boldsymbol{a}} - \mu_{\boldsymbol{b}})^2 + (\nu_{\boldsymbol{a}} - \nu_{\boldsymbol{b}})^2}, \\ d_{\mathrm{g}}(\boldsymbol{a}, \boldsymbol{b}) &= \sqrt{(\delta_{\boldsymbol{a}} - \delta_{\boldsymbol{b}})^2} \end{aligned} \quad (6)$$

In Eq. (5), the coefficients w_{s} and w_{g} determine the relative importance of the three distance, which are empirically set to 0.005 and 0.3, respectively.

Following [18], in the assignment step, we associate each seed with a limited window, i.e., only pixels located in the small window can be assigned to the seed. This restriction significantly reduces the computational overhead for distance calculations and also facilitates producing compact superpixels. The window length is set to $3S$, where S is the grid length of the uniform partition of image.

To summarize, for each individual i in DES, X_i is the *genotype*, while the corresponding seed set $\boldsymbol{s}_i$ and label matrix L_i represent the *phenotype*. The objective evaluation is performed on phenotype to calculate the superpixel segmentation costs (Subsection D). Based on the costs, the evolution operators, including mutation, crossover, and selection, are performed on the genotype to seek the best solution (Subsection F).

D. Objective Evaluation

Given a seed set $\boldsymbol{s}_i$ and the label matrix L_i, the objective function to be optimized is defined as

$$F(\boldsymbol{s}_i, L_i) = f_{\mathrm{wse}}(\boldsymbol{s}_i, L_i) + \lambda \cdot (-1) \cdot f_{\mathrm{bg}}(L_i) + \gamma \cdot f_{\mathrm{r}}(L_i), \quad (7)$$

where f_{wse}, f_{bg}, and f_{r} stand for the within-superpixel error, boundary gradient, and regularization terms, respectively. Since we want to minimize f_{wse} and f_{r} while maximizing f_{bg}, f_{bg} is prefixed with a negative sign in the aggregation. Parameters λ and γ control the relative influence of the three terms, which are fixed to constant values. In the experiments of this paper, we empirically use $\lambda = 8$ and $\gamma = 2$.

1) Within-Superpixel Error: The within-superpixel error is defined as the mean squared error (MSE) of assigning each pixel to its nearest seed:

$$f_{\mathrm{wse}}(\boldsymbol{s}_i, L_i) = \frac{1}{N} \sum_{j=1}^{N} \left[d^2(\boldsymbol{p}_j, \boldsymbol{s}_{i,l}) \middle| l = L_i(j) \right], \quad (8)$$

where $\boldsymbol{p}_j$ is the feature vector of the jth pixel, $l = L_i(j)$ is the pixel label, $\boldsymbol{s}_{i,l}$ is the feature of seed, and d is the distance measure defined in Eq. (5). This term evaluates the local color homogeneity within superpixels.

2) Boundary Gradient: Improving the boundary adherence is a primary goal of superpixel segmentation. However, we could not know the natural image boundaries in prior, so that the boundary gradient is used instead, since it provides a signal of the boundary strength.

First, based on the label matrix L_i, we calculate the boundary map of superpixels in the following way. For each position in L_i, there are eight neighbors located in its 3×3 neighborhood. If some neighboring labels are different from the label of the current position, this position belongs to the superpixel boundaries. The operation can be implemented as follows. Given L_i, first, we shift the matrix one unit along all eight directions to generate eight neighboring matrices: $L_i^{(1)}, L_i^{(2)}, \cdots, L_i^{(8)}$. Then, the truth value of whether the jth position in L_i belongs to the superpixel boundaries can be calculated as

$$B_{L_i}(j) = \mathbb{I}\left(\sum_{n=1}^{8} \mathbb{I}\left(L_i(j) \neq L_i^{(n)}(j) \right) > 1 \right), \quad (9)$$

where $\mathbb{I}$ is a truth indicator function. In this way, we obtain the superpixel boundary map of individual i.

As stated in Subsection A, before starting the algorithm, the gradient δ of each pixel is calculated and normalized. Here, the gradient map of the entire image I is denoted as ΔI. Then, the boundary gradient term in the objective function is fomulated as

$$f_{\mathrm{bg}}(L_i) = \frac{\sum_{j=1}^{N} \Delta I(j) \cdot B_{L_i}(j)}{\sum_{j=1}^{N} B_{L_i}(j)} \quad (10)$$

Therefore, f_{bg} is the average gradient value of superpixel boundaries, which indicates the strength of boundary adherence of individual i. Note that we use the average but rather than the summarization of boundary gradient to avoid the algorithm to favor long boundaries.

3) Regularizer: Since we partition the image with N pixels into K superpixels, the expected number of pixels assigned to each superpixel is $c_{\mathrm{m}} = N/K$. The regularizer punishes segmenting superpixels with sizes far away from c_{m}, which is defined as

$$f_{\mathrm{r}}(L_i) = \frac{1}{K} \cdot \frac{\sum_{k=1}^{K} (c_{L_i,k} - c_{\mathrm{m}})^2}{c_{\mathrm{m}}^2} \quad (11)$$

where $c_{L_i} = [c_{L_i,1}, c_{L_i,2}, \cdots, c_{L_i,K}]$ denotes the number of pixels assigned to each seed, K and c_{m}^2 are divided for the purpose of normalization.

E. Kernel Update

After evaluating P individuals, we can find the best-so-far individual in the population that has the best evaluation score. The evaluation score is named "fitness" in the context of DE. For minimization problems, a lower cost score corresponds to a better fitness of the individual to the environment. Using bsf to denote the index of the best-so-far individual, the kernel

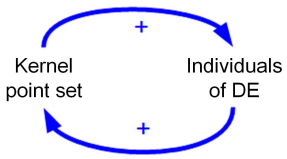


Fig. 5. The positive feedback loop in DES.

points $[l_k, \alpha_k, \beta_k, \mu_k, \nu_k, \delta_k]^{\mathrm{T}}$ $(k = 1, 2, \cdots, K)$ are updated based on the label matrix of bsf, L_{bsf}. Particularly, for each $k \in \{1, 2, \cdots, K\}$, retrieve all the pixels that are labelled with k in L_{bsf}, then set the kernel point to the mean feature vector of these pixels.

Introducing the kernel point set has two advantages. First, the optimal superpixel seeds are somewhat, though not exactly, associated with the centering values of the current segmentation [18]. However, the raw DE component explores the problem space without using this prior knowledge. Thus, we introduce the kernel point set to make use of the mean feature vectors of a partition. It can be considered as a kind of "heuristic information", which provides additional knowledge to assist the algorithm in searching the optimal superpixel seeds. Second, note that DE is a population-based search algorithm that can produce various candidate partitions during the search, but only the partition of the best-so-far individual is adopted to update the kernel points. The mechanism exploits the strengths of the historically best information, which is an elitism strategy. To summarize, the proposed algorithm forms a circle of positive feedback shown in Fig. 5: on the one hand, the kernel points are adopted as heuristic information to improve the phenotype of individuals, and, on the other hand, the improved phenotype L_{bsf} is used to update the kernels. The positive feedback loop constantly improves the best label matrix during the iterations. In the next subsection, we are going to describe another component that plays an important role in the positive loop, the operations of DE.

F. Differential Evolution

In DES, each individual X_i is encoded by a vector representing the differential lengths made to the coordinates of kernel points, by which we can derive a candidate seed set. In the initialization, the P individuals are randomly generated in the search space, with fitness evaluated according to Eq. (7). Then, at each iteration step (named "generation" in the context of DE), as illustrated in Fig. 6, three genetic operators, namely, the mutation, crossover, and selection, are performed one by one to update the individuals.

1) Mutation: In evolutionary biology, "mutation" stands for a random and sudden perturbation on the genotype of individuals, which hereafter alters the product of the genes. Following this concept, in the mutation of DES, for each individual i, a *mutant vector* $V_i = [v_{i,1}, v_{i,2}, \cdots, v_{i,D}]$ is created as :

$$v_{i,j} = p_{bsf,j} + 0.5 \cdot (p_{r_1,j} - p_{r_2,j}), \quad r_1 \neq r_2 \neq bsf, \tag{12}$$

where $X_{bsf} = [p_{bsf,1}, p_{bsf,2}, \cdots, p_{bsf,D}]$ is the best fitted individual, r_1 and r_2 are two randomly generated indices,

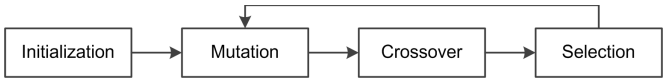


Fig. 6. The flowchart of DE.

the coefficient 0.5 is a scalar factor, $i = 1, 2, \cdots, P$, and $j = 1, 2, \cdots, D$. Note that the three individuals joining the mutation are kept distinct.

Eq. (12) is a typical mutation form in DE, which is named "DE/best/1" [31]. There are two requirements for the mutant vector: diversity and possibly good quality. To this end, random perturbation is performed on the parameter vector of the best fitted individual. It can be observed from Eq. (12) that the perturbation of individuals is derived from scaled population difference, which has the following benefits. In the initial stage, since the population is randomly distributed, the perturbation length is long, which helps the algorithm to explore the problem space. After a few generations, with the convergence of the population, the perturbation length decreases to a very small value, which helps local exploitation. Therefore, the mechanism balances the global exploration and local exploitation of the algorithm. In addition, if the fitness landscape is compact on one dimension and diverse on another, the population difference on the former dimension will become smaller than that of the latter. This mechanism brings an automatic contour matching effect in the fitness landscape. To conclude, rather than setting a step size manually, DE adapts the step size automatically during the optimization process.

2) Crossover: Crossover is used to combine the current individual and the mutant vector so as to reproduce an offspring named *trial vector*. In DES, the binomial crossover [31] is used, by which $X_i = [p_{i,1}, p_{i,2}, \cdots, p_{i,D}]$ randomly exchanges some components from the mutant vector V_i with 90% probability. Denote the trial vector as $U_i = [u_{i,1}, u_{i,2}, \cdots, u_{i,D}]$, it is calculated as

$$u_{i,j} = \begin{cases} v_{i,j}, & \text{if } \mathrm{rand}_{i,j}(0,1) < 0.9 \text{ or } j = j_{\mathrm{rnd}} \\ p_{i,j}, & \text{otherwise} \end{cases} \tag{13}$$

where $\mathrm{rand}_{i,j}(0, 1)$ is a uniform random number generator and j_{rnd} is a random dimension index to keep the trial vector has at least one dimension different from the individual X_i. In this way, the offspring inherits the genotypes of both the current individual and the mutant vector, which has good diversity and may lead to promising phenotype and competitive objective evaluation score.

3) Selection: Selection is performed to keep good genotype of the problem while discarding the inferior ones to the next generation. In this step, first, the fitness of the trial vector generated in Eq. (13) is evaluated according to the procedures presented in Subsections C and D. Then, the individual X_i is updated as

$$X_i = \begin{cases} U_i, & \text{if } fitness(U_i) \text{ is\_better\_than } fitness(X_i) \\ X_i, & \text{otherwise} \end{cases} \tag{14}$$

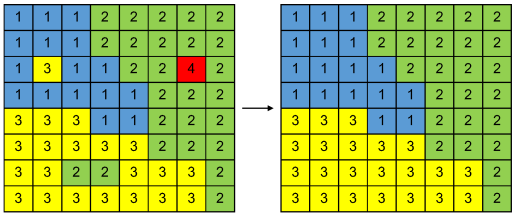


Fig. 7. Illustration of the postprocessing step.

where $fitness$ is calculated according to the objective function we want to minimize. The population will preserve either the individual or the trial vector according to which one is more fitted to the environment. The number of individuals in the population is hence kept as a constant. In this way, the population evolves with respect to the minimization of objective, and it never deteriorates.

Note that the convergence and stability analysis of DE are reported in the literature [58]. The above operators iterate for T generations in order to output a satisfactory solution. The setting of T balances the segmentation performance and computational overhead of DES. We found that $T = 10$ is enough for producing promising superpixel segmentation, and, at the same time, the required processing time is relatively short. Meanwhile, we use a population size $P = 5$.

G. Postprocessing

At the end of optimization, the best-so-far label matrix, L_{bsf}, is outputted as the superpixel segmentation results. However, in the above procedures, we do not explicitly enforce the connectivity of superpixels. Therefore, there may possibly exist some "orphaned" pixels that are assigned with labels different from the surrounding pixels. Following the other algorithms [18], [42], we perform a postprocessing step named "enforce connectivity" on the label matrix to correct the lables of isolated pixels and obtain the final segmentation results. As illustrated in Fig. 7, the isolated pixels are merged into their surrounding superpixels in the postprocessing step.

H. Complexity Analysis

The pseudo code of DES is presented **Algorithm 1**. Given an image of size N, the complexity of feature extraction and initial seeding is $O(N)$. By restricting the covering range of seeds in a limited window, the complexity of label assignment is reduced to $O(N)$, as reported in [18]. The cost in calculating the objective value (i.e., the fitness of individuals in DE), as well as the kernel update, is $O(N)$. The DE operation has a linear complexity to the number of seeds and it is thus $O(K)$ complex. The seeding process iterates for a constant number of times, which is independent with N and K. Finally, the postprocessing procedure costs $O(N)$ time according to [18] and [19]. This way, DES is $O(N + K)$ complex. Since $K \ll N$, the $O(K)$ term can be omitted. To conclude, the proposed DES algorithm has a linear complexity $O(N)$ to the image size.

Algorithm 1 DES

/* Initialization */
Extract and normalize features $[l, \alpha, \beta, \mu, \nu, \delta]^{\mathrm{T}}$ for each pixel of the image;
Initialize kernel points $\boldsymbol{c} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_K]$ from the center of uniform partitions of the image (move them to their lowest-gradient neighbors);
Initialize the variable range of DE and sampling P individuals randomly within the search range;
for $i = 1$ **to** P **do**
    Derive candidate seeds $\boldsymbol{s}_i = [\boldsymbol{s}_{i,1}, \boldsymbol{s}_{i,2}, \cdots, \boldsymbol{s}_{i,K}]$ according to Eqs. (3) and (4);
    Calculate label matrix L_i by NN assignment using the distance measure defined in Eq. (5);
    Evaluate fitness $F(\boldsymbol{s}_i, L_i)$ by the objective function defined in Eq. (7);
end for
/* Main Loop */
repeat
    Identify the best-so-far label matrix L_{bsf};
    Update the kernel points $\boldsymbol{c} = [\boldsymbol{c}_1, \boldsymbol{c}_2, \cdots, \boldsymbol{c}_K]$ to the mean feature vector of each superpixel defined by L_{bsf};
    for $i = 1$ **to** P **do**
        Generate mutant vector V_i by mutation in Eq. (12);
        Generate trial vector U_i by crossover in Eq. (13);
        Seeding, NN assignment, and fitness evaluation based on trial vector U_i;
        Update individual i by selection in Eq. (14);
    end for
until Maximum generations T is arrived;
Enforce connectivity of L_{bsf};
return Superpixel label matrix L_{bsf}.

I. Discussion

Compared with the algorithms reviewed in Section II, the proposed DES algorithm has the following characteristics/advantages:

1) The previous seeding-based algorithms perform superpixel segmentation by minimizing local color variance, which encounter difficulties in segmenting multiple objects with similar color features (see discussion in Fig. 2). In comparison, DES embeds and optimizes the global segmentation properties in the objective function straightforwardly, which is able to improve the boundary adherence of superpixels.
2) The information propagation between the kernel points and the DE population forms a positive feedback loop that enhances the quality of these two components interchangeably. DES is capable of self-adaptively searching in the fitness landscape without external guidance and finding the global optimum.
3) The existing global optimization-based algorithms endure high computational cost and long execution time [17], [27], [40], [41], which are not suitable for preprocessing the images in some time-critical applications. In contrast, although adopting a global optimization

model, the computational cost of DES is linear to the image size.

4) Different from some algorithms that produce irregular superpixels [21], [25], [26], [42], in the design of DES, we also pay attention to the regularity issue.

The initial results of the work have been accepted to publish in a conference paper [28] (*ICME*'16). In this journal extension, not only new experimental results and analysis are presented, but also the algorithm design is improved. To facilitate the comparisons between the two versions of DES, the preliminary version of DES in *ICME*'16 is denoted as "DESp". Generally, DES and DESp both apply DE to tackle the superpixel segmentation in a global optimization way, and their frameworks are similar. Therefore, DES and DESp have shared properties such as bottom-up segmentation manner, explicit control of superpixel number, regularity of superpixels, and linear complexity to the image size N. The major difference between DES and DESp lies in the individual representation of DE. In DESp, each individual represents the coordinates of superpixel seeds directly. In comparison, we now use the individuals to represent the differential parameter vectors made to the kernel points and then to obtain the superpixel seeds. The kernel points are the mean feature vectors of superpixels represented by the best-so-far individual. Basically, this new mechanism has two advantages as described in Subsection E. First, the use of mean feature vectors can be considered as a kind of "heuristic information," which provides additional knowledge for algorithm to search in the problem space. Second, note that the partition by the best-so-far individual is utilized. It is an elitism strategy to exploit the strengths of the historical search information so as to improve the performance of DE. Besides, DESp utilizes only color and spatial features, and DES further incorporated the contour feature in the feature representation vector. The reason of this improvement is intuitive: for some images that endure low color contrast, the color and spatial features may not be sufficient to provide a good discriminability for the pixels. Thus, DES further exploits the contour feature to relieve this problem.

IV. Experiments and Comparisons

Experiments are carried on the Berkeley Segmentation Data Set (BSDS 500) [32] and the PASCAL-S dataset [33]. BSDS is the most commonly used dataset to examine the performance of superpixel segmentation algorithms. As an improved version of BSDS 300, the BSDS 500 contains 500 natural images. In addition, the PASCAL-S dataset contains 850 images of various object classes and sizes. Both datasets are assigned with human-annotated ground-truth labels.

The proposed DES algorithm is compared with eleven state-of-the-art segmentation algorithms: SLIC [18], LSC [19], DBSCAN [20], QS [21], LRW [22], TP [23], Lattice [24], SEEDS [25], ERS [26], EOpt0, and EOpt1 [27]. Table I summarizes the characteristics of the algorithms, including the segmentation manner, control of superpixel number, regularity of superpixels, and computational complexity. Although these algorithms are based on different concepts, they are all unsupervised methods that input a single image and output the label matrix of superpixels for the image. In addition, from the original references of these algorithms, it can be seen that the algorithms are evaluated using the same criteria, namely, undersegmentation error, boundary recall, etc. Therefore, it is reasonable to make comparison of these algorithms. All the algorithms are implemented based on the codes provided by the authors and executed on the same platform. Note that a single CPU is used throughout the execution.

TABLE I

Characteristics of the Superpixel Segmentation Algorithms Performed in the Experiments

Algorithm	DES	SLIC [18]	LSC [19]	DBSCAN [20]
Segmentation manner	bottom-up	bottom-up	bottom-up	bottom-up
Control of K	explict	explict	explict	imexplict
Shape of superpixels	regular	regular	regular	regular
Complexity	$O(N)$	$O(N)$	$O(N)$	$O(N)$
Algorithm	QS [21]	LRW [22]	TP [23]	Lattice [24]
Segmentation manner	bottom-up	bottom-up	bottom-up	top-down
Control of K	implicit	explict	explict	explict
Shape of superpixels	inregular	regular	regular	quasi-grid
Complexity	$O(N^2)$	$O(N^2)$	$O(N)$	$O(N^{1.5}\log N)$
Algorithm	SEEDS [25]	ERS [26]	EOpt0 [27]	EOpt1 [27]
Segmentation manner	top-down	top-down	top-down	top-down
Control of K	implicit	explict	explict	implicit
Shape of superpixels	inregular	inregular	regular	inregular
Complexity	$O(N)$	$O(N^2\log N)$	$O(\frac{N^3}{K^2})$	$O(\frac{N^3}{K^2})$

A. Performance Metrics

To evaluate the quality of superpixels produced by different algorithms, standard performance metrics are used, namely, the Undersegmentation Error (UE) [18], Boundary Recall (BR) [18], and Achievable Segmentation Accuracy (ASA) [26]. The three metrics are formulated as follows. Let $\mathcal{SP} = \{sp_1, sp_2, \cdots, sp_K\}$ denote the superpixel set, $\mathcal{GT} = \{gt_1, g_2, \cdots, gt_{N_g}\}$ denote the ground-truth segmentations, where N_g stands for the number of ground-truth segmentations. Then, $|sp_i|$ and $|gt_j|$ represent the sizes of superpixels and segmentations, which satisfy $\sum_{i=1}^{K}|sp_i| = \sum_{j=1}^{N_g}|gt_j| = N$.

Undersegmentation Error (UE) measures the area of superpixels that slops over the ground-truth segmentation borders.

$$UE_{\mathcal{GT}}(\mathcal{SP}) = \frac{1}{N}\left[\sum_{i=1}^{K}\sum_{j=1}^{N_g}|sp_i|\cdot H(sp_i, gt_j) - N\right],$$
$$H(sp_i, gt_j) = \begin{cases}1, & \text{if } |sp_i \cap gt_j| > 5\%\cdot|sp_i| \\ 0, & \text{otherwise}\end{cases} \quad (15)$$

For each pair of superpixel sp_i and ground-truth segmentation gt_j, if the number of intersecting points of sp_i and gt_j exceeds five percent of $|sp_i|$, the two elements are considered as overlapped. If a superpixel overlaps with more than one ground-truth segmentations, UE increases.

Boundary Recall (BR) measures the proportion of grand-truth boundaries that is recalled by the superpixel boundaries. Define $B_{\mathcal{SP}}$ and $B_{\mathcal{GT}}$ as the boundaries of $\mathcal{SP}$

TABLE II
PERFORMANCE METRICS OBTAINED BY DIFFERENT ALGORITHMS ON THE BSDS 500 ($K = 400$)

	GRID	SLIC	LSC	DBSCAN	QS	LRW	TP	Lattice	SEEDS	ERS	EOpt0	EOpt1	DES
UE	*0.532*	0.222	0.197	0.225	0.227	0.212	0.239	0.304	0.200	0.196	0.231	0.306	**0.185**
BR	*0.330*	0.833	0.948	0.916	0.914	0.833	0.803	0.898	0.938	0.942	0.800	0.811	**0.953**
ASA	*0.919*	0.953	0.961	0.953	0.955	0.956	0.948	0.932	0.959	0.959	0.950	0.936	**0.962**
Time	*0.000*	0.097	0.375	**0.031**	2.048	94.735	4.144	0.247	0.058	0.815	1.818	4.469	0.264

and $\mathcal{GT}$, respectively. BR is calculated as

$$BR_{\mathcal{GT}}(\mathcal{SP}) = \frac{1}{|B_{\mathcal{GT}}|} \sum_{p \in B_{\mathcal{GT}}} \mathbb{I}(\min_{q \in B_{\mathcal{SP}}} ||p, q|| < \epsilon), \quad (16)$$

where $\mathbb{I}$ is an indicator function shows whether the nearest neighbor q in $B_{\mathcal{SP}}$ is within the ϵ-distance of pixel p in $B_{\mathcal{GT}}$. Commonly, ϵ is set to 2.

Achievable Segmentation Accuracy (ASA) provides the accuracy upperbound when using superpixels as basic units for image segmentation.

$$ASA_{\mathcal{GT}}(\mathcal{SP}) = \frac{1}{N} \sum_{i=1}^{K} \left[\max_{j \in \{1, \cdots, N_g\}} |sp_i \cap gt_j| \right]. \quad (17)$$

Assigning each superpixel with the label of the ground-truth segment to which the overlap area is the largest, the proportion of correctly identified labels is the ASA.

Besides, the time efficiency plays an important role in examining the superpixel segmentation algorithms. We record and compare the average execution time of different algorithms for processing a single image in the experiments.

B. Comparisons of Superpixel Results

1) Results on BSDS 500: Tested on the BSDS 500, the quantitative results obtained by the algorithms for segmenting 100-600 superpixels are compared in Fig. 8. Specifically, Table II presents the numerical results when $K = 400$, where the results of the GRID [59] algorithm that uniformly segments the image is also presented as a baseline. First, considering the results of GRID, it can be noticed from Table II that the algorithm achieves a good ASA value: 0.9185. On the one hand, this result verifies the applicability of using superpixel segmentation as a preprocessing step for image segmentation, because even using such a naive method, the segmentation results will not decrease too much. On the other hand, the result indicates that the discriminability of the ASA metric for evaluating superpixel quality is not as good as those of UE and BR.

Generally speaking, as can be observed from Fig. 8 and Table II, three algorithms performs very well on the test suite, namely, DES, LSC, and ERS. DES outperforms the others in terms of all the performance metrics. Meanwhile, since DES has linear complexity, the execution is fast. Besides, as illustrated in Fig. 1 and Fig. 3, DES produces relatively regular superpixels. Therefore, to summarize, DES satisfies the common criteria of superpixel segmentation.

Considering the compared algorithms, SLIC is very fast, and its generated superpixels are very regular. The algorithm

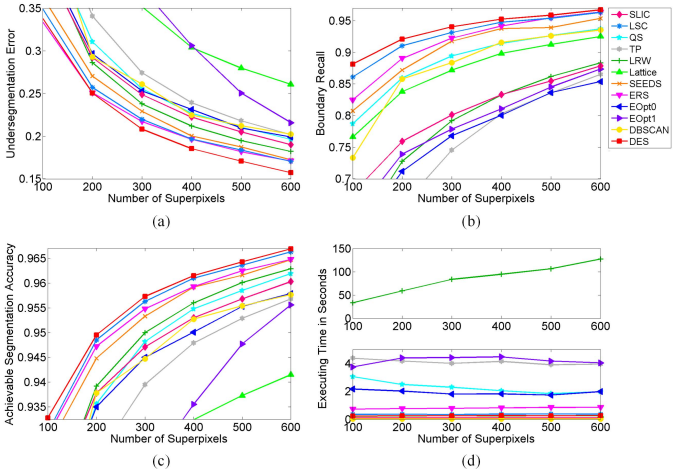


Fig. 8. Performance comparison curves of different superpixel algorithms on the BSDS 500. (a) Undersegmentation error. (b) Boundary recall. (c) Achievable segmentation accuracy. (d) Executing time per image.

is hence suitable for preprocessing some time-critical applications. The overall performance of LSC ranks the second in our experiments. LSC achieves very competitive BR and ASA values, while its execution is fast. As for the DBSCAN algorithm, the most prominent character lies in the execution efficiency. The algorithm requires only 0.03 seconds for processing an image (which is the fastest among all the compared algorithms), and it is hence suitable for real-time superpixel applications. QS performs better than a few algorithms, but it requires more than 2 seconds for handling an image and generates irregular results. LRW performs well in considering the UE and ASA, while its computational cost is high. Using morphological operation, TP produces very regular superpixels. However, the algorithm endures long execution time and poor boundary adherence. Lattice obtains promising BR values, but its performance on UE and ASA is unsatisfactory. Although the theoretical complexity Lattice is $O(N^{1.5} \log N)$, our experiments shows that the algorithm is faster than many $O(N)$ algorithms in practical execution. ERS and SEEDS rank the third and the forth in the experiments. SEEDS is the second fastest among the compared algorithms, whereas ERS is much slower. A drawback of the two algorithms is that they both produce extremely irregular superpixels. Besides, EOpt0 and EOpt1 achieves similar BR performance, but the UE and ASA of EOpt0 are much better than those of EOpt1. Both algorithms suffer from long computational overhead.

2) Results on PASCAL-S: Further, the results of different algorithms on the PASCAL-S dataset are shown in Fig. 9. It can be observed that, on this dataset, DES also performs the

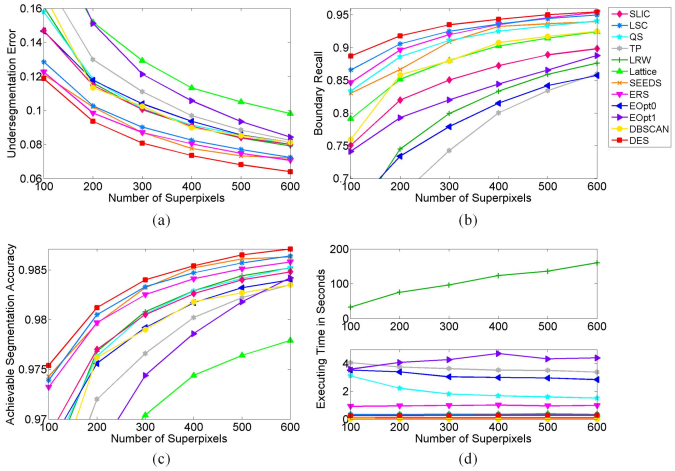


Fig. 9. Performance comparison curves of different superpixel algorithms on the PASCAL-S. (a) Undersegmentation error. (b) Boundary recall. (c) Achievable segmentation accuracy. (d) Executing time per image.

Fig. 10. Visual superpixel results. From left to right: input image, ground-truth segmentation, and results based on SLIC, LSC, QS, LRW, SEEDS, ERS, and the proposed DES.

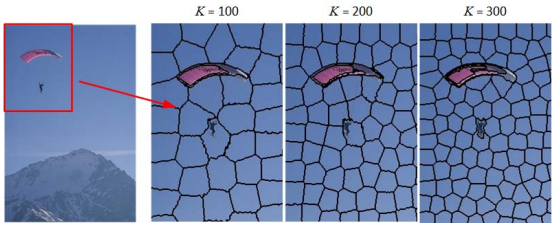


Fig. 11. A failure case of DES. The "parachuter" is not successfully detected when $K = 100$ and 200.

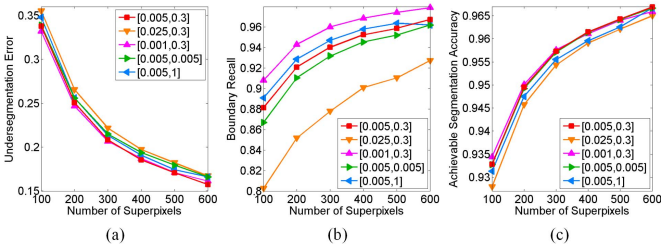


Fig. 12. Investigation of parameters $[w_s, w_g]$. (a) Undersegmentation error. (b) Boundary recall. (c) Achievable segmentation accuracy.

best in terms of UE, BR, and ASA. Another three algorithms, namely, SEEDS, ERS, and LSC, can be assigned to the secondary best category. For the UE metric, the results of SEEDS and ERS are better than those of LSC, whereas LSC outperforms on the BR metric. Considering the execution time, we can use a threshold, 0.5 seconds, to separate the algorithms into two classes. The first class consists of DBSCAN, SEEDS, SLIC, DES, Lattice, and LSC, which are efficient. In comparison, ERS, QS, EOPT0, TP, EOPT1, and LRW belong to the second class that requires much more execution time.

To summarize, the proposed DES outperforms most state-of-the-art methods, which is an effective, efficient, and reliable algorithm for superpixel segmention. In addition to Fig. 1 and Fig. 3, Fig. 10 shows more visual results, where DES is compared with SLIC, LSC, QS, LRW, SEEDS, and ERS (the six mostly competitive algorithms in the above quantitative comparison). In this qualitative comparison, DES also exhibits promising performance. Besides, we show a failure case of DES in Fig. 11, where the "parachuter" is not successfully detected when generating small numbers of superpixels ($K = 100$ and 200). The proposed algorithm generate regular and similarly-sized superpixels and thus it encounters a difficulty in detecting the object which is much smaller than the average superpixel size. Nevertheless, the problem can be resolved when increasing K to 300, as can be observed in Fig. 11.

C. Parameter Investigation

The distance measure in the feature space contains two coefficients (w_s and w_g) to determine the relative importance of spatial and contour features to the color feature (note that the coefficient of color feature is fixed as 1). In the above experiments, DES is tested with $w_s = 0.005$ and $w_g = 0.3$. For w_s, if we increase the value to 0.025, as shown in Fig. 12, the performance of DES decreases to a considerable extent (particularly in considering the BR values). This is because the pixels in local regions commonly share similar color properties, so they would be very sensitive to the differences of spatial coordinates. When increasing w_s, the influence of color feature in the distance measure decreases significantly, resulting in the unsatisfactory segmentation performance. On the contrary, if we decrease w_s to 0.001, the performance of DES is slightly improved. However, at the same time, the resulting superpixels will become less regular. For w_g, as shown in Fig. 12, if the contour feature is given a very small weight (e.g., 0.005 which is equal to w_s), the effect of this feature can hardly be expressed. The performance of DES decreases to some extent. On the other side, if we set w_g

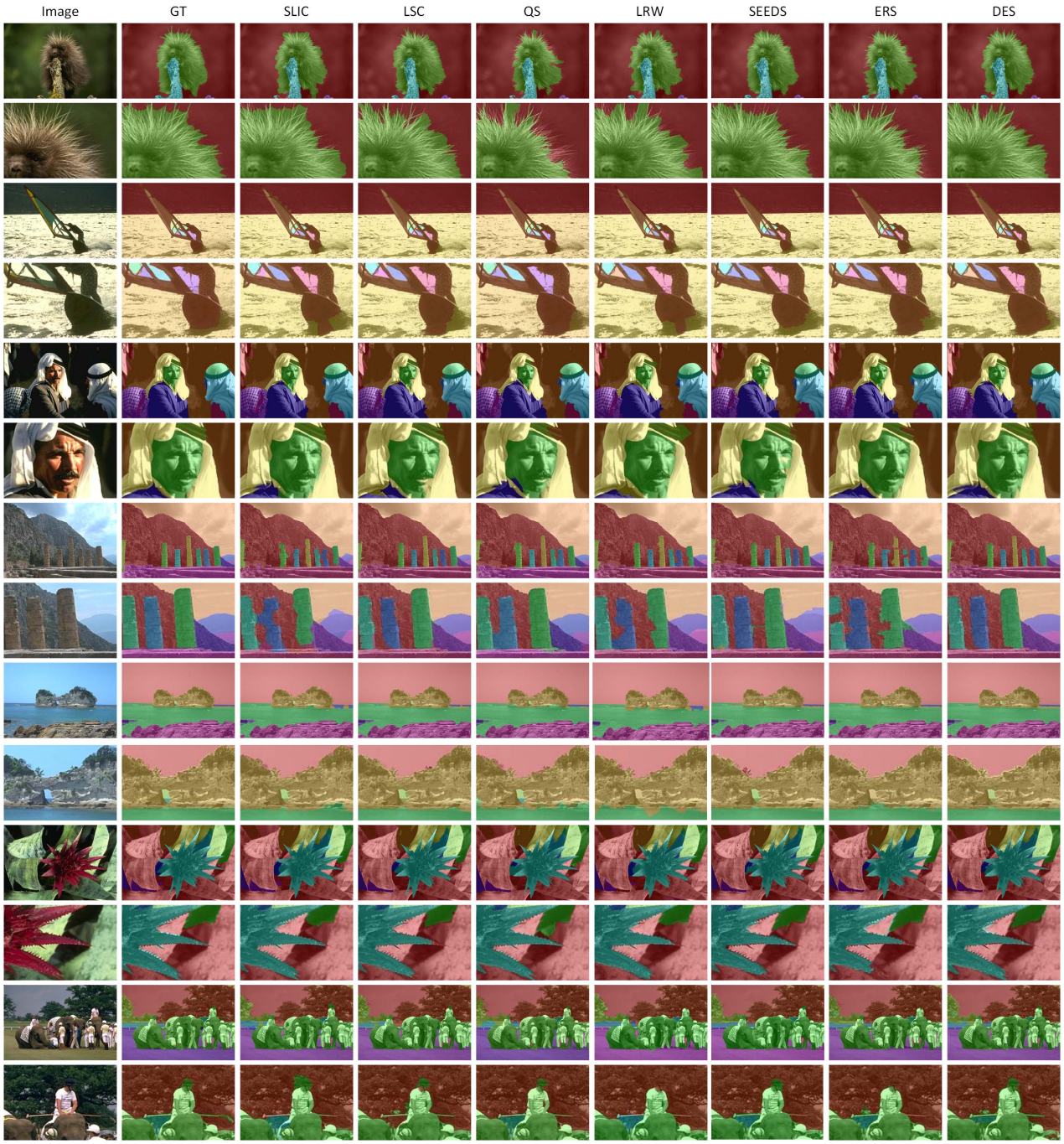


Fig. 13. Achievable segmentation results. From left to right: input image, ground-truth segmentation, and results based on SLIC, LSC, QS, LRW, SEEDS, ERS, and the proposed DES.

to a large value (e.g., $w_g = 1$ as the weight of color feature), the superpixels generated by DES can achieve slightly better BR, because the contour feature favors the boundary detection. However, the performance on the UE and ASA decreases and the generated superpixels become less regular.

D. Preprocessing Performance for Image Segmentation

The superpixel segmentation technique is commonly used as a preprocessing procedure in image segmentation and the related fields. Because of this, it is desired that, by using superpixel segmentation, the subsequent algorithm obtains not only substantial speedup but also promising segmentation performance. This can be investigated by assuming that an ideal classifier is performed after the superpixel segmentation. The ideal classifier is able to assign each superpixel with the most "proper" class, namely, the label of the ground-truth segment within which the superpixel overlaps the most. The method is similar to evaluating the ASA, but now we can visualize the results, namely, the achievable segmentations.

TABLE III
PREPROCESSING PERFORMANCE FOR AN IDEAL IMAGE SEGMENTATION ALGORITHM ON THE BSDS 500

	SLIC	LSC	DBSCAN	QS	LRW	TP	Lattice	SEEDS	ERS	EOpt0	EOpt1	DES
Dice	0.956	0.964	0.955	0.957	0.958	0.950	0.930	0.962	0.962	0.952	0.937	**0.965**
Jaccard	0.922	0.935	0.921	0.924	0.926	0.912	0.883	0.933	0.931	0.916	0.893	**0.936**
Conformity	0.904	0.922	0.902	0.899	0.910	0.887	0.843	0.916	0.917	0.890	0.817	**0.923**

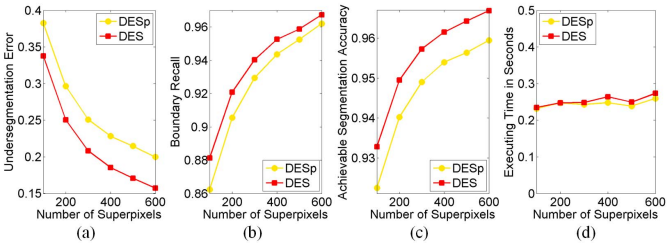


Fig. 14. Performance comparison curves of DESp and DES on the BSDS 500. (a) Undersegmentation error. (b) Boundary recall. (c) Achievable segmentation accuracy. (d) Executing time per image.

Fig. 15. Visual superpixel results of DESp (top) and DES (bottom).

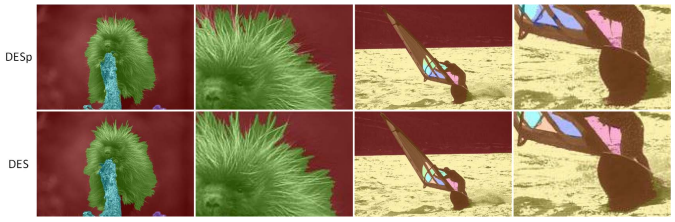


Fig. 16. Achievable segmentation results of DESp (top) and DES (bottom).

In terms of the ASA, according to the numerical comparison in Subsection B, the generally best performed superpixel algorithms are DES, LSC, ERS, SEEDS, LRW, QS, and SLIC. Fig. 13 further visualizes the results by a few image examples. It can be observed that the visual results of DES and LSC are better than the others, which are very similar to the human-labelled ground truth. The proposed DES algorithm slightly outperforms LSC (e.g., boat in the second image, headdress and chin in the third image). In addition, we use three image segmentation measures, namely the Dice, Jaccard, and Conformity coefficients [60], to compare all algorithms in a quantitative way. The results are presented in Table III. The conclusions are consistent with those of our qualitative comparisons: DES and LSC outperforms the other algorithms (while DES slightly outperforms LSC). The two algorithms exhibit better preprocessing performance than the others and they are hence more suitable to be applied to image segmentation.

E. More Comparisons on the Performance

Further, we conduct experimental comparisons between DES and DESp to see the differences of performance brought by our new algorithm design. Both algorithms are tested on the BSDS 500 dataset and evaluated by the standard performance metrics: UE, BR, ASA. The quantitative and quantitative results are reported in Figs. 14 and 15, respectively. As shown in Fig. 14(a)-(c), the results of DES are generally better than those of DESp. Particularly, the undersegmentation rate is significantly reduced. These results verify the effectiveness of the improvements made on the algorithm design, namely, introducing the kernel points and the contour feature. From Fig. 14(d), it can be observed that DES requires longer execution time than DESp. This is because of the additional computational cost derived from computing the kernel points and the contour feature. However, the increase in the required processing time of DES is insignificant. The visual results in Fig. 15 show that both algorithms generate relatively regular superpixels, while the DES achieves better boundary adherence.

In addition, we also compare the preprocessing performance of DESp and DES in Fig. 16. The experimental settings are identical with those in Section IV-C. As expected, DES also outperforms DESp in the preprocessing performance.

V. CONCLUSION

We develop a novel superpixel segmentation algorithm, DES, which is able to yield accurate superpixel results efficiently. The promising performance of DES owes much to the comprehensive objective function that considers several global properties in the segmentation. To the best of our knowledge, this work is the first trial to optimize the boundary adherence in a straightforward way by embedding a boundary gradient term in the objective. In this way, the generated superpixels are atomic regions that are unlikely to span multiple objects in the image. A regularizer is also considered in the objective, so as to enforce generating superpixels with similar sizes. These two terms, together with the within-superpixel error, are then aggregated into a single objective function to optimize. For solving such a complex model, we use a DE algorithm inspired by the natural evolution process. The algorithm is efficient for solving the global optimization problems while posing no restrictions on the form of objective functions. The computational complexity of DES is $O(N)$.

Qualitative and quantitative experimental results show that the proposed algorithm outperforms state-of-the-art superpixel algorithms in terms of undersegmentation error, boundary recall, and achievable segmentation accuracy. Further experiments on image achievable segmentation verify the promising preprocessing performance of the algorithm. In the future, we will further investigate the algorithm in noisy and cluttered environments. Besides, owing to the excellent performance of DES, it is appealing to apply the proposed algorithm to preprocessing many computer vision tasks.

REFERENCES

[1] W. Zhu, S. Liang, Y. Wei, and J. Sun, "Saliency optimization from robust background detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2014, pp. 2814–2821.

[2] H. Li, H. Lu, Z. Lin, X. Shen, and B. Price, "Inner and inter label propagation: Salient object detection in the wild," *IEEE Trans. Image Process.*, vol. 24, no. 10, pp. 3176–3186, Oct. 2015.

[3] H. Lu, X. Zhang, J. Qi, N. Tong, X. Ruan, and M.-H. Yang, "Co-bootstrapping saliency," *IEEE Trans. Image Process.*, vol. 26, no. 1, pp. 414–425, Jan. 2017.

[4] J. Shen, Y. Du, and X. Li, "Interactive segmentation using constrained Laplacian optimization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 7, pp. 1088–1100, Jul. 2014.

[5] X. Dong, J. Shen, L. Shao, and M.-H. Yang, "Interactive cosegmentation using global and local energy optimization," *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 3966–3977, Nov. 2015.

[6] X. Dong, J. Shen, L. Shao, and L. Van Gool, "Sub-Markov random walk for image segmentation," *IEEE Trans. Image Process.*, vol. 25, no. 2, pp. 516–527, Feb. 2016.

[7] L. Zhang, Y. Gao, Y. Xia, K. Lu, J. Shen, and R. Ji, "Representative discovery of structure cues for weakly-supervised image segmentation," *IEEE Trans. Multimedia*, vol. 16, no. 2, pp. 470–479, Feb. 2014.

[8] M. Jian and C. Jung, "Interactive image segmentation using adaptive constraint propagation," *IEEE Trans. Image Process.*, vol. 25, no. 3, pp. 1301–1311, Mar. 2016.

[9] A. Farag, L. Lu, H. R. Roth, J. Liu, E. Turkbey, and R. M. Summers, "A bottom-up approach for pancreas segmentation using cascaded superpixels and (deep) image patch labeling," *IEEE Trans. Image Process.*, vol. 26, no. 1, pp. 386–399, Jan. 2017.

[10] J. Shen, J. Peng, X. Dong, L. Shao, and F. Porikli, "Higher order energies for image segmentation," *IEEE Trans. Image Process.*, vol. 26, no. 10, pp. 4911–4922, Oct. 2017.

[11] W. Wang, J. Shen, X. Li, and F. Porikli, "Robust video object cosegmentation," *IEEE Trans. Image Process.*, vol. 24, no. 10, pp. 3137–3148, Oct. 2015.

[12] X. Dong, J. Shen, and L. Shao, "HSP2P: Hierarchical superpixel-to-pixel dense image matching," *IEEE Trans. Circuits Syst. Video Technol.*, to be published. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/7523995/

[13] Y. Wang and Q. Zhao, "Superpixel tracking via graph-based semi-supervised SVM and supervised saliency detection," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jun./Jul. 2015, pp. 1–6.

[14] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 7, pp. 1442–1468, Jul. 2014.

[15] J. Tighe and S. Lazebnik, "SuperParsing: Scalable nonparametric image parsing with superpixels," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2010, pp. 352–365.

[16] K. Yamaguchi, M. H. Kiapour, L. E. Ortiz, and T. L. Berg, "Retrieving similar styles to parse clothing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 5, pp. 1028–1040, May 2015.

[17] X. Ren and J. Malik, "Learning a classification model for segmentation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2003, pp. 10–17.

[18] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.

[19] Z. Li and J. Chen, "Superpixel segmentation using linear spectral clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1356–1363.

[20] J. Shen, X. Hao, Z. Liang, Y. Liu, W. Wang, and L. Shao, "Real-time superpixel segmentation by DBSCAN clustering algorithm," *IEEE Trans. Image Process.*, vol. 25, no. 12, pp. 5933–5942, Dec. 2016.

[21] A. Vedaldi and S. Soatto, "Quick shift and kernel methods for mode seeking," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2008, pp. 705–718.

[22] J. Shen, Y. Du, W. Wang, and X. Li, "Lazy random walks for superpixel segmentation," *IEEE Trans. Image Process.*, vol. 23, no. 4, pp. 1451–1462, Apr. 2014.

[23] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi, "TurboPixels: Fast superpixels using geometric flows," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2290–2297, Dec. 2009.

[24] A. P. Moore, S. J. D. Prince, J. Warrell, U. Mohammed, and G. Jones, "Superpixel lattices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2008, pp. 1–8.

[25] M. Van den Bergh, X. Boix, G. Roig, B. de Capitani, and L. Van Gool, "SEEDS: Superpixels extracted via energy-driven sampling," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2012, pp. 13–26.

[26] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa, "Entropy rate superpixel segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2011, pp. 2097–2104.

[27] O. Veksler, Y. Boykov, and P. Mehrani, "Superpixels and supervoxels in an energy optimization framework," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2010, pp. 211–224.

[28] Y.-J. Gong, Y. Zhou, and X. Zhang, "A superpixel segmentation algorithm based on differential evolution," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2016, pp. 1–6.

[29] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.

[30] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution—An updated survey," *Swarm Evol. Comput.*, vol. 27, pp. 1–30, Apr. 2016.

[31] K. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Berlin, Germany: Springer-Verlag, 2006.

[32] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 5, pp. 898–916, May 2011.

[33] Y. Li, X. Hou, C. Koch, J. M. Rehg, and A. L. Yuille, "The secrets of salient object segmentation," in *Proc. CVPR*, 2014, pp. 280–287.

[34] J. Chen, Z. Li, and B. Huang, "Linear spectral clustering superpixel," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3317–3330, Jul. 2017.

[35] R. Achanta and S. Süsstrunk, "Superpixels and polygons using simple non-iterative clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, Jul. 2017, pp. 4895–4904.

[36] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, May 2002.

[37] Y. Liang, J. Shen, X. Dong, H. Sun, and X. Li, "Video supervoxels using partially absorbing random walks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 5, pp. 928–938, May 2016.

[38] F. Meyer, "Color image segmentation," in *Proc. Int. Conf. Image Process. Appl. (ICIP)*, 1992, pp. 303–306.

[39] V. Machairas, M. Faessel, D. Cárdenas-Peña, T. Chabardes, T. Walter, and E. Decencière, "Waterpixels," *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 3707–3716, Nov. 2015.

[40] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[41] X. He, R. S. Zemel, and D. Ray, "Learning and incorporating top-down cues in image segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2006, pp. 338–351.

[42] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *Int. J. Comput. Vis.*, vol. 59, no. 2, pp. 167–181, Sep. 2004.

[43] Y. Zhou, L. Ju, and S. Wang, "Multiscale superpixels and supervoxels based on hierarchical edge-weighted centroidal Voronoi tessellation," *IEEE Trans. Image Process.*, vol. 24, no. 11, pp. 3834–3845, Nov. 2015.

[44] M. Van den Bergh, X. Boix, G. Roig, and L. Van Gool, "SEEDS: Superpixels extracted via energy-driven sampling," *Int. J. Comput. Vis.*, vol. 111, no. 3, pp. 298–314, 2015.

[45] J. Peng, J. Shen, A. Yao, and X. Li, "Superpixel optimization using higher order energy," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 26, no. 5, pp. 917–927, May 2016.

[46] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.
[47] Y. L. Li, Z. H. Zhan, Y. J. Gong, W. N. Chen, J. Zhang, and Y. Li, "Differential evolution with an evolution path: A DEEP evolutionary algorithm," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1798–1810, Sep. 2015.
[48] X. Qiu, J.-X. Xu, K. C. Tan, and H. A. Abbass, "Adaptive cross-generation differential evolution operators for multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 2, pp. 232–244, Apr. 2016.
[49] N. M. Hamza, D. L. Essam, and R. A. Sarker, "Constraint consensus mutation-based differential evolution for constrained optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 3, pp. 447–459, Jun. 2016.
[50] M. F. Zaman, S. M. Elsayed, T. Ray, and R. A. Sarker, "Evolutionary algorithms for dynamic economic dispatch problems," *IEEE Trans. Power Syst.*, vol. 31, no. 2, pp. 1486–1495, Mar. 2016.
[51] D. Qiao and G. K. H. Pang, "A modified differential evolution with heuristic algorithm for nonconvex optimization on sensor network localization," *IEEE Trans. Veh. Technol.*, vol. 65, no. 3, pp. 1676–1689, Mar. 2016.
[52] Z. Gao, Z. Pan, and J. Gao, "Multimutation differential evolution algorithm and its application to seismic inversion," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 6, pp. 3626–3636, Jun. 2016.
[53] S. Sarkar and S. Das, "Multilevel image thresholding based on 2D histogram and maximum Tsallis entropy—A differential evolution approach," *IEEE Trans. Image Process.*, vol. 22, no. 12, pp. 4788–4797, Dec. 2013.
[54] A. Khan, M. A. Jaffar, and L. Shao, "A modified adaptive differential evolution algorithm for color image segmentation," *Knowl. Inf. Syst.*, vol. 43, no. 3, pp. 583–597, 2015.
[55] U. Maulik and I. Saha, "Automatic fuzzy clustering using modified differential evolution for image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 48, no. 9, pp. 3503–3510, Sep. 2010.
[56] N. Armanfard, J. P. Reilly, and M. Komeili, "Local feature selection for data classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 6, pp. 1217–1227, Jun. 2016.
[57] M. Du, X. Nan, and L. Guan, "Monocular human motion tracking by using DE-MC particle filter," *IEEE Trans. Image Process.*, vol. 22, no. 10, pp. 3852–3865, Oct. 2013.
[58] S. Dasgupta, S. Das, A. Biswas, and A. Abraham, "On stability and convergence of the population-dynamics in differential evolution," *AI Commun.*, vol. 22, no. 1, pp. 1–30, 2009.
[59] P. Neubert and P. Protzel, "Superpixel benchmark and comparison," in *Forum Bildverarbeitung*. Karlsruhe, Germany: KIT Scientific Publishing, 2012, pp. 1–12.
[60] H.-H. Chang, A. H. Zhuang, D. J. Valentino, and W.-C. Chu, "Performance measure characterization for evaluating neuroimage segmentation algorithms," *NeuroImage*, vol. 47, no. 1, pp. 122–135, 2009.

Yue-Jiao Gong (M'15) received the B.S. and Ph.D. degrees in computer science from Sun Yat-Sen University, China, in 2010 and 2014, respectively. From 2015 to 2016, she was a Post-Doctoral Research Fellow with the Department of Computer and Information Science, University of Macau, Macau. She is currently an Associate Professor with the School of Computer Science and Engineering, South China University of Technology, China. Her research interests include evolutionary computation and machine learning methods, and also their applications to image processing. She has authored over 50 papers in her research area. She currently serves as a Reviewer for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON NEURAL NETWORK AND LEARNING SYSTEMS, and the IEEE TRANSACTIONS ON IMAGE PROCESSING.

Yicong Zhou (M'07–SM'14) received the B.S. degree in electrical engineering from Hunan University, Changsha, China, and the M.S. and Ph.D. degrees in electrical engineering from Tufts University, MA, USA. He is currently an Associate Professor and the Director of the Vision and Image Processing Laboratory, Department of Computer and Information Science, University of Macau, Macau, China. His research interests include chaotic systems, multimedia security, image processing and understanding, and machine learning.

He was a recipient of the Third Price of Macau Natural Science Award in 2014. He is the Co-Chair of Technical Committee on Cognitive Computing in the IEEE Systems, Man, and Cybernetics Society. He served as an Associate Editor for the *Neurocomputing*, the *Journal of Visual Communication and Image Representation*, and the *Signal Processing: Image Communication*.