Int J Comput Vis (2013) 103:1–21
DOI 10.1007/s11263-012-0588-6

Structure-Sensitive Superpixels via Geodesic Distance

Peng Wang · Gang Zeng · Rui Gan · Jingdong Wang · Hongbin Zha

Received: 16 February 2012 / Accepted: 9 October 2012 / Published online: 9 November 2012

Abstract Segmenting images into superpixels as supporting regions for feature vectors and primitives to reduce computational complexity has been commonly used as a fundamental step in various image analysis and computer vision tasks. In this paper, we describe the structure-sensitive superpixel technique by exploiting Lloyd's algorithm with the geodesic distance. Our method generates smaller superpixels to achieve relatively low under-segmentation in structure-dense regions with high intensity or color variation, and produces larger segments to increase computational efficiency in structure-sparse regions with homogeneous appearance. We adopt geometric flows to compute geodesic distances amongst pixels. In the segmentation procedure, the density of over-segments is automatically adjusted through iteratively optimizing an energy functional that embeds color homogeneity, structure density. Comparative experiments with the Berkeley database show that the proposed algorithm outperforms the prior arts while offering a comparable computational efficiency as TurboPixels. Further applications in image compression, object closure extraction and video segmentation demonstrate the effective extensions of our approach.

Keywords Superpixel segmentation · Geodesic distance · Iterative optimization · Structure-sensitivity

P. Wang · G. Zeng (✉) · H. Zha
Key Laboratory on Machine Perception, Peking University, Beijing, China
e-mail: g.zeng@ieee.org

P. Wang
e-mail: jerrywang@pku.edu.cn

H. Zha
e-mail: zha@cis.pku.edu.cn

R. Gan
School of Mathematical Sciences, Peking University, Beijing, China
e-mail: raygan@ieee.org

J. Wang
Microsoft Research Asia, Beijing, China
e-mail: jingdw@microsoft.com

1 Introduction

Image over-segmentation has been widely applied in various computer vision pipelines, such as segmentation (Arbelaez et al. 2009; Xiao and Quan 2009; Wang et al. 2008; Hoiem et al. 2005), recognition (Kaufhold et al. 2006), object tracking (Wang et al. 2011; Rasmussen 2007), localization (Fulkerson et al. 2009) and modeling (He et al. 2006; Nwogu and Corso 2008; Micusík and Kosecká 2010).

In these applications, over-segments (aka superpixels) represent small regions with homogeneous appearance and conform to local image structures, and thus provide a better support for region-based features than local windows. With superpixels, the computational cost significantly decreases especially for probabilistic, combinatorial or discriminative approaches, since the underlying graph is greatly simplified in terms of graph nodes and edges. Most superpixel methods have to face the following challenges: on one hand they are required to reduce image complexity by locally grouping pixels regarding intensity boundaries, and on the other hand they should avoid under-segmentation and maintain a certain level of detailed structures. These two aspects conflict with each other, and there have been various optimization techniques proposed to make trade-offs in order to solve this dilemma, for example, the mean shift algorithm (Comaniciu and Meer 2002), the normalized cuts (Shi and Malik 2000), the local viaration (Felzenszwalb and Huttenlocher 2004), the geometric flows (Levinshtein et al. 2009b) and

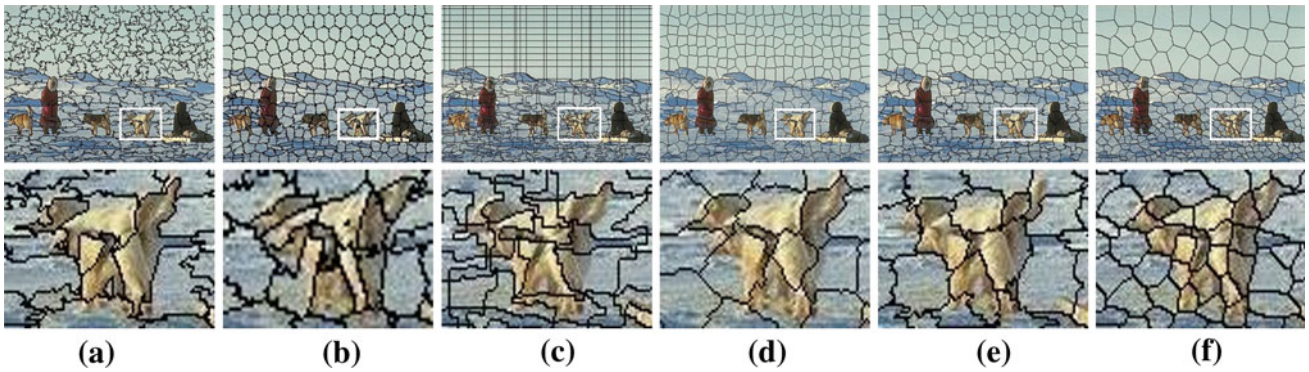


Fig. 1 Over-segmentations obtained with five algorithms: **a** Graph-based method (Felzenszwalb and Huttenlocher 2004), **b** N-Cuts (Shi and Malik 2000), **c** Superpixel Lattice (Moore et al. 2008), **d** TurboPixels (Levinshtein et al. 2009b), **e** GraphCut superpixel (Veksler et al. 2010) and **f** the method presented in this paper. The *second row* shows the view which zooms in on the regions of interest defined by the *white rectangles*

the watershed (Vincent and Soille 1991; Meyer and Maragos 1999; Tai et al. 2007).

As a visual and comparative illustration, Fig. 1 shows the segmentation results obtained by using several state-of-the-art superpixel methods: *Graph-based method* (Felzenszwalb and Huttenlocher 2004), *Lattice* (Moore et al. 2008), *N-Cuts* (Mori 2005; Levinshtein et al. 2009a), *TurboPixels* (Levinshtein et al. 2009b), *GraphCut superpixel* (Veksler et al. 2010) and also by using our method. It is noted by previous art (Levinshtein et al. 2009b) that the Graph-based method could easily generate under-segmentation for regions of irregular shapes and sizes due to the lack of compactness constraints. While other methods employ compactness constraints and markedly restrict under-segmentation. The advantage of utilizing compactness has also been demonstrated in Levinshtein et al. (2009b).

N-Cuts-based superpixel methods (Mori 2005; Levinshtein et al. 2009a) are variations of the normalized cuts algorithm (Shi and Malik 2000), in which the compactness is guaranteed by normalizing the cut cost using edge weights. However, the global optimization is computationally costly, and the time complexity of the segmentation increases significantly with the number of pixels and image size.

Lattice (Moore et al. 2008) generates superpixels by detecting vertical or horizontal strips, and thus naturally maintains a grid structure of regions. In order to achieve an adaptive lattice, the scene shape prior is then combined into the lattice framework (Moore et al. 2009). Their further investigation of lattice superpixels (Moore et al. 2010) is derived from global optimization of a well-designed energy function. The superpixel generation is initialized with a grid, and the graph cut algorithm is adopted to optimize the vertical and horizontal seams alternatively.

GraphCut superpixel (Veksler et al. 2010) over-segment an image using roughly regularly-placed patches under a single energy framework. Comparing with N-Cuts-based superpixel methods, this algorithm also keeps the character of compactness and provides comparable results. However, by using the graph-cut optimization method, it becomes much more efficient.

The most related work with ours is proposed by Levinshtein et al., namely a geometric flow based algorithm (aka TurboPixels) for superpixel segmentation (Levinshtein et al. 2009b). Starting from initial seeds regularly placed onto the image, TurboPixels uses the level set method for superpixels' evolution. It yields a lattice-like structure of compact regions, and more importantly it is efficient especially when compared with N-Cuts-based over-segmentation.

As shown in Fig. 1, a further observation is that given a large diversity of scene layout, the prospective distortion is unavoidably introduced by imaging process, and the density of image contents often varies in different parts of the image. The over-segments of *Lattice*, *N-Cuts*, *TurboPixels* and *GraphCut superpixel* in Fig. 1b–e are too large to represent image appearances and lead to under-segmentation in regions near intensity boundaries, while the segments are rather small in homogeneous regions resulting in unnecessary overhead in high-level applications. To this end, quasi-uniform distribution or layout of superpixel on an image raises a dilemma situation for over-segmentation, since the number of superpixels is hard to choose. This has also been proven by several methods that exploit multiple level over-segments technique as a starting point for further scene segmentation (Malisiewicz and Efros 2007; Russell et al. 2006).

In order to overcome the aforementioned problem and maintain the intuitive consistency with the human vision system, a better image representation could be achieved by assigning the density of superpixels adaptively with respect to the co-occurrence of image contents or the "density" of image structures. This motivates us to introduce a structure-sensitive density function and to generate superpixels as regions with similar sizes in terms of this density function.

Fig. 2 Geodesic distance versus Euclidean distance: Image contents in-between could provide a crucial evidence for measuring the similarity between two pixels

The density function is also driven by the following analysis on the similarity measure among pixels. A commonly used similarity measure is the Euclidean distance in a high-dimensional space based on three color components and two image coordinates. The main disadvantage of such measurement is the irrelevance to the image contents in-between: the measure remains the same no matter whether there is a path along which the appearance transits smoothly (see Fig. 2). It often leads to disconnection and irregularity on segments' shape and size. In order to avoid the above flaws, the similarity measure in the proposed algorithm is defined by the geodesic distance (Peyré et al. 2010). The aforementioned density function forms the basis of the geodesic distance, namely the distance increment at a particular image point becomes large if the local density is high, and vice versa.

Most recently, the geodesic distance has been taken in use for interactive segmentation and matting in Bai and Sapiro (2007), Criminisi et al. (2008), and Gulshan et al. (2010). To the best of our knowledge, however, it has never been used as criteria for determining the distribution and magnitude of superpixels in over-segmentation.

In summary, the contributions of this paper includes the following three aspects: (1) we propose an explicit energy for superpixel segmentation which aims at developing compact and structure-sensitive superpixels; (2) we introduce an efficient iterative optimization technique which includes two steps: over-segmentation with known superpixel centers and center relocation given known over-segments; (3) the geodesic distance is induced for measuring the structure and layout of superpixels, which is aware of the image contents.

1.1 Our Approach

Figure 3 shows an overview of our system. The proposed algorithm resembles Lloyd's algorithm (Lloyd 1982) but with the geodesic distance defined in Eq. (7). It is based on an energy functional (see Sect. 2) that integrates structure, compactness and homogeneous constraints. The density and placement of superpixels are required to be sensitive to image structure and thus are adjusted adaptively during this iterative algorithm.

Given a user-specified amount of superpixels, the algorithm first puts some seeds along with small disturbance in order to avoid the placement on strong intensity boundaries. The seeds are sampled based-on the "density" of the image structure and serve as initial estimates of the superpixel centers.

There are two key components in this iterative approach. The first one generates over-segments from the current set of centers. The fast marching method (Sethian 1996b) is employed to compute the geodesic distance and thus to generate a Voronoi diagram based on the distance. It has high computational efficiency and requires more restricted forms of the underlying velocity function. Our velocity function is based on the structure density with special care for satisfying the required forms. The details of this part can be found in Sect. 3.1.

The second component refines the locations of the centers according to superpixels' distribution and magnitudes. The relocation is based on an energy minimization formulation defined with the geodesic distance. Additional superpixels are created by splitting existing ones when certain conditions of their density are satisfied. The splitting strategy guarantees a descent of the energy and accelerates the algorithm. The description of this part is in Sect. 3.2.

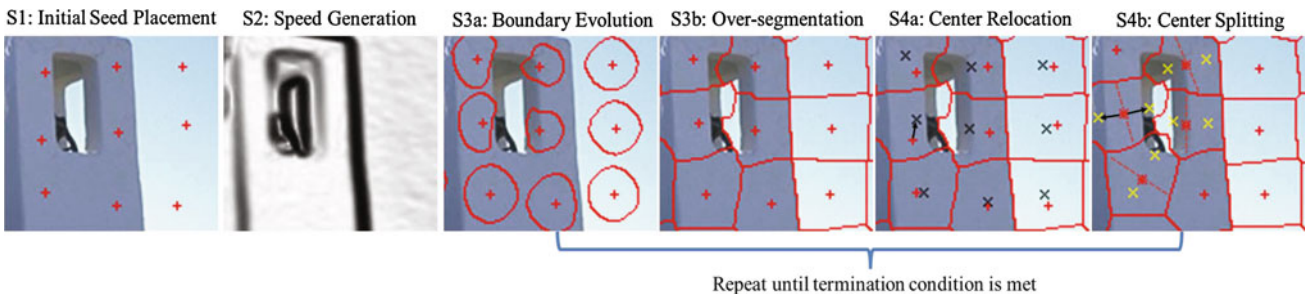


Fig. 3 The procedure of our algorithm: Initial seeds (*S1*) grow with the speed (*S2*) to form an over-segmentation (*S3*), and the centers are relocated or split (*S4*) by certain criteria related by shape or size. In *S4a* the *red plus* is the original places of center points and *black cross* is the recalculated places. In *S4b* the *red asterisk* represents the seeds detected to be split and the *yellow cross* is the newly generated seeds. The *arrows* on graph illustrate the motion of particular seeds (Color figure online)

In addition, we further introduce an alternative optimization strategy which includes merging scheme and discuss the pros and cons. An efficient implementation for acceleration is also proposed. The details are presented in Sects. 4.1, 4.2 and 4.3 respectively.

The paper is organized as follows: Sect. 2 introduces the formulation of structure sensitive superpixels, and the optimization method are presented in Sect. 3. For deeper understanding of our algorithm, we discussed two optimization methods in Sect. 4. In Sect. 5, we introduce the implementation details and evaluation experiments. Section 6 discusses some applications based on our algorithm and the conclusion follows in Sect. 7.

2 Problem Formulation via Geodesic Distance

Given an input image $I(\mathbf{x})$, where $\mathbf{x}$ indicates the pixel's position (x, y), the goal is to over-segment $I(\mathbf{x})$ into dense small regions representing superpixels at different locations. We assign a unique label l to each superpixel and use $L(\mathbf{x})$ to denote the label of the current pixel $\mathbf{x}$. The set of label is represented as $\mathscr{L}$. All pixels belonging to the l_{th} superpixel S_l can be represented by $S_l = \{\mathbf{x}|L(\mathbf{x}) = l\}$. To this end, the over-segmentation problem belongs to a clustering problem (aka unsupervised learning), in general.

As shown in Fig. 2, image contents in-between could provide crucial evidence for measuring the similarity between two pixels. We exploit the geodesic distance $D_g(\mathbf{x}_i, \mathbf{x}_j)$ to define the similarity between two pixels $\mathbf{x}_i$ and $\mathbf{x}_j$ on an image:

$$D_g(\mathbf{x}_i, \mathbf{x}_j) = \min_{P_{\mathbf{x}_i,\mathbf{x}_j}} \int_0^1 D(P_{\mathbf{x}_i,\mathbf{x}_j}(t)) \|\dot{P}_{\mathbf{x}_i,\mathbf{x}_j}(t)\| dt, \tag{1}$$

where $P_{\mathbf{x}_i,\mathbf{x}_j}(t)$ is a path connecting the pixel $\mathbf{x}_i$, $\mathbf{x}_j$ (for $t = 0$ and $t = 1$ respectively). The density function $D(x)$ is used as the distance increment, and we employ a similar definition as Levinshtein et al. (2009b), because it performs well in searching for the edges of the image. It takes the form as follows:

$$D(\mathbf{x}) = e^{\frac{E(\mathbf{x})}{\nu}}, \quad E(\mathbf{x}) = \frac{\|\nabla I\|}{G_\sigma * \|\nabla I\| + \gamma}, \tag{2}$$

where ν is a scaling parameter. Notice that the color image I here is slightly smoothed to avoid certain noisy edges. $E(\mathbf{x})$ is an edge measurement which provides normalization of gradient magnitude $\|\nabla I\|$ of color image. This allows weak but isolated edges to have a significant effect on density. G_σ is the Gaussian function with its standard deviation being σ. The parameter γ is set to guarantee that very weak intensity boundaries do not effect the density computation too much.

Since $D(\mathbf{x})$ is a monotonically increasing function of gradient magnitude which is large on edges, the geodesic distance of a path across an intensity boundary is always larger than that in a homogeneous region. In addition, we can see the term $D(\mathbf{x})$ produces a constant distance increment (i.e. $D(\mathbf{x}) = 1$ if $E(\mathbf{x}) = 0$) in regions of homogeneous appearance, and thus retains the minimum possible isoperimetric ratio. This also makes the superpixels compact so as to avoid large under-segmentation when the image regions contain little edge information.

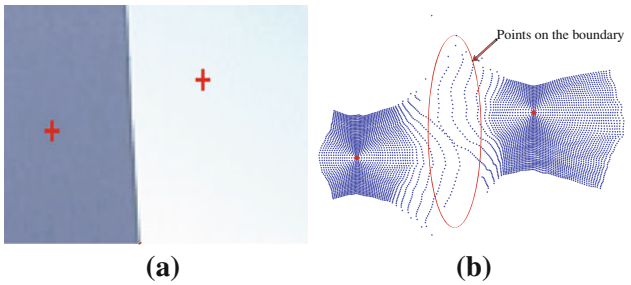


Fig. 4 Toy example of data points got through geodesic distance. **a** The image and located two center points. **b** The geodesic distance plotted based on the Euclidean coordination. As can be seen, the points on the edge are floating far away from the two cluster

With the distance measurements at hand, the problem is to cluster the pixels into regions, yielding the superpixels. Recent geodesic-distance-based clustering methods use K-means (Feil and Abonyi 2007), or Fuzzy C-means (Kim et al. 2007).

However, using the geodesic distance, as showed in Fig. 4, given a superpixel center close to a boundary, the pixels in the boundary are floating away, and they are regarded as noisy point, especially when the boundary has large contrast. In such a case, we do not want to assign those pixels into the cluster for certain. Thus, we consider the soft-assign model, i.e. Soft K-means, to formulate the over-segmentation, because such a kind of method is robust to the noisy points.

2.1 Energy Minimization

2.1.1 Homogeneity Penalization

Formally, soft K-means is minimizing the weighted distance between each point to each center:

$$E_{image} = \sum_l \int_{I(x)} W_{\mathbf{x},l} D_g^2(\mathbf{c}_l, \mathbf{x}) d\mathbf{x}, \tag{3}$$

where $\mathbf{c}_l$ is the center of l_{th} cluster and $W_{\mathbf{x},l}$ is a assign weight measuring the membership that data point $\mathbf{x}$ belongs to the cluster.

In previous works such as Fuzzy K-means, $W_{\mathbf{x},l}$ is inferenced through lagrange dual function. We suppose that the

points in each cluster are the set of observations drawn independently from a Gaussian distributions, formally:

$$p(\mathbf{x}|\mathbf{c}_l) = \frac{1}{(2\pi\varepsilon)^{1/2}} \exp\left\{\frac{-D_g(\mathbf{c}_l, \mathbf{x})^2}{2\varepsilon}\right\}. \tag{4}$$

Then the $W_{\mathbf{x},l}$ in Eq. (3) could be computed as:

$$W_{\mathbf{x},l} = \frac{p(\mathbf{x}|\mathbf{c}_l)}{\sum_{l\in\mathscr{L}} p(\mathbf{x}|\mathbf{c}_l)}. \tag{5}$$

2.1.2 Structure Penalization

In addition, we further enforce the structure information brought by the magnitudes of superpixels for better representing local structures on the image, formally we wish the area of different superpixels over the density map are similar:

$$\text{Structure}: \ A_l \approx A_{l'}, \forall l \neq l', \ \text{with}$$
$$A_l = \int_{S_l} D(\mathbf{x})d\mathbf{x}, \tag{6}$$

where A_l denotes the area of superpixel S_l.

From Eq. (2), the density function is high in the regions with much intensity variation and thus leads to smaller S_l on the image. This motivate us to penalize the image area A_l that is relatively large. Here we define an average area $\overline{A}$, which is calculated as $\frac{\sum_l A_l}{N} = \frac{\int_{\mathbf{x}} D(\mathbf{x})d\mathbf{x}}{N}$ in which N is the total number of superpixels specified by users.

We choose to penalize the geodesic distance $D_g(\mathbf{c}_l, \mathbf{x})$ increment when the area of the l_{th} superpixel is larger than the $\overline{A}$ to limit the area difference. The rectified distance is defined as:

$$D'_g(\mathbf{c}_l, \mathbf{x}) = f(D_g(\mathbf{c}_l, \mathbf{x}), A_l(\mathbf{c}_l, \mathbf{x}); \alpha) \tag{7}$$

where $A_l(\mathbf{c}_l, \mathbf{x})$ is the current area encircled by the evolving contour with the distance to the center $\mathbf{c}_l$ being $D_g(\mathbf{c}_l, \mathbf{x})$, f is an implicit function that penalize $D_g(\mathbf{c}_l, \mathbf{x})$ when $A_l(\mathbf{c}_l, \mathbf{x})$ is larger than $\overline{A}$, and α is the balancing factor for balancing the influence brought by the area. We will give the detail of the f in Sect. 3.

Using the rectified distance in Eq. (3), our problem is to solve the minimization of the energy functional:

$$\min_{\mathscr{C}} E_{total} = \min_{\mathscr{C}} \sum_l \int_{I(x)} W'_{\mathbf{x},l} D'^2_g(\mathbf{c}_l, \mathbf{x})d\mathbf{x}, \tag{8}$$

where $\mathscr{C} = \{\mathbf{c}_l\}_{l=1}^K$ is the set of centers. $W'_{\mathbf{x},l}$ is the membership in Eq. (4) with the rectified geodesic distance. In summary, we design an energy functional which embeds image homogeneity and structure density. At last, the superpixels $\{S_l\}$ are generated based on optimized $W'_{\mathbf{x},l}$ by setting the $L(\mathbf{x}) = \max_l W'_{\mathbf{x},l}$ which makes the final superpixels compact with similar area.

3 Iterative Optimization

Due to the non-convex property of Eq. (8) and the induced latent variable, we choose an iterative scheme to minimize the energy functional E_{total}, More precisely, our optimization process is similar to Lloyd's algorithm (Lloyd 1982) as mentioned in Sect. 1.1, The convergence and robustness of the Lloyd's algorithm has been elaborated by Du et al. (2006). During the iterative procedure, the weight of soft K-means $W_{\mathbf{x},l}$, the centers $\{\mathbf{c}_l\}$ are alternatively updated.

The main difference to the traditional EM optimization scheme is the usage of a top-down hierarchical optimization strategy during the iteration for adaptively generating new components through splitting. This strategy accelerates the optimization which is also known as the bisecting k-means (Li and Chung 2007). Moreover, we treasure the efficiency of the algorithm and use approximations to reduce the computational burdens during the iterative optimization. Since superpixel segmentation is a preprocessing step for many vision tasks, the computational efficiency is essential for its capability in various applications. The details about strategies and accelerations of the two steps of our algorithm are described in Sects. 3.1 and 3.2 respectively.

3.1 Weight Estimation

Given a set of centers $\{\mathbf{c}_l\}$, the goal of this step is to compute the $W_{\mathbf{x},l}$

3.1.1 The Geodesic Distance Computation

In order to generate the geodesic distances defined in Eq. (7), we employed the fast marching method which is proposed by Sethian (1996a) for better computational efficiency since this over-segmentation step may get called several times during the outer iterations. Moreover, in our configuration, the front end of the evolving contour can only move in the outward normal direction (i.e. the contour expands rather than shrinking), which fits well with the restricted forms of the underlying velocity functions of the fast marching.

The velocity function for calculating the geodesic distance D_g in Eq. (7) is defined as follows:

$$V(\mathbf{x}) = D(\mathbf{x})^{-1}, \tag{9}$$

where $D(\mathbf{x})$ is the density function defined in Eq. (2).

With the above velocity function, we use the fast marching method as the numerical solver for the boundary problems of the Eikonal equation,

$$V(\mathbf{x})\|\nabla D_g(\mathbf{c}_l, \mathbf{x})\| = 1, \quad \text{with } D_g(\mathbf{c}_l, \mathbf{c}_l) = 0, \ \forall l. \tag{10}$$

As stated in Sect. 2.1, we apply the f to the original distance in order to penalize the distance increment when the $A(\mathbf{c}_l, \mathbf{x})$ is larger than $\overline{A}$. Here we use $A_l(d)$ for short to represent currently encircled area with distance being d. Intuitively, if $A_l(d)$ exceeds $\overline{A}$, the speed of the evolving contour should decrease such that any evolving contour of a nearby cluster center $\mathbf{c}_{l'}$ tends to cover a relative bigger (or smaller) field on the image if $A'_l(d) < A_l(d)$ turns out to be true.

To make the computation of the distance robust against image noise or cluster edges, we adopt a Gaussian function to the velocity defined in Eq. (9):

$$V_l(\mathbf{x}, d) = V(\mathbf{x}) \cdot G'_{\sigma'}(\max\{0, A_l(d)/\overline{A} - 1\}),$$
$$\text{with} \quad A_l(d) = \int_{\{\mathbf{x}|\mathbf{x}\in S_l, D'_g(\mathbf{c}_l,\mathbf{x})<d\}} D(\mathbf{x})d\mathbf{x}, \tag{11}$$

where $G'_{\sigma'}(\cdot)$ denotes an un-normalized Gaussian function with its standard deviation σ'. For efficiency, $V_l(\mathbf{x}, d)$ is only calculated with superpixels larger than $\overline{A}$ (due to the splitting strategy in Sect. 3.2 the superpixels are generally larger than $\overline{A}$). Thus, the Gaussian is a weight function to reduce the velocity when $A_l(d)$ exceeds $\overline{A}$. In our experiments, the $\sigma' = \frac{1}{\alpha}$ for balancing the effect brought by the area. We set the α to be $\sqrt{M/N_{iter}}$, where M is the pixel number in an image and N_{iter} is the center number in current iteration. This makes the superpixel respect gradient information when the superpixel number is small at the beginning and consider the area more when the superpixel number is sufficient.

With the new velocity, the $D'_g(\mathbf{c}_l, \mathbf{x})$ can be calculated through the following Eikonal equation:

$$V_l(\mathbf{x}, d)\|\nabla D'_g(\mathbf{c}_l, \mathbf{x})\| = 1, \quad \text{with } D'_g(\mathbf{c}_l, \mathbf{c}_l) = 0, \ \forall l. \tag{12}$$

3.1.2 Weight Approximation for Acceleration

In this step, the main task is to compute the weight $W_{\mathbf{x},l}$ in Eq. (5). Nevertheless, using Eq. (5) needs to calculate the geodesic distance between each pixel to each center, which is extremely time costly and makes the practical usage intractable. For efficiency, we consider to compute the geodesic distance as less as possible and approximate the Eq. (5). From the equation, the weight is negative related with the distance between the pixel $\mathbf{x}$ and the center vector $\mathbf{c}_l$ in most cases. In experiments, we observe that $W_{\mathbf{x},l}$ is decreasing with the increasing distance similar with the numerator of the calculation equation.

Thus, we approximate $W_{\mathbf{x},l}$ by:

$$\hat{W}_{\mathbf{x},l} = \begin{cases} e^{-D'_g(\mathbf{c}_l,\mathbf{x})^2/\varepsilon} & \text{if } l = \arg\min_l 0 D'_g(\mathbf{c}_l, \mathbf{x}) \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

where ε is a scaling parameter. It helps balancing the effect from the softness. As the ε goes larger, the algorithm goes closer to simply K-means. With such approximation, the geodesic distance only needs to be computed once.

To validate that the $W_{\mathbf{x},l}$ reasonably approximates the probability, we have done two experiments. Firstly, we calculate the statistic of a probability distribution of all pixels belonging to a center $\mathbf{c}_l$ with a real example. As shown by Fig. 5, the probability is decreasing exponentially as we predicted.

Secondly, we run several toy examples by over-segmenting images and show the relocated centers of one image in

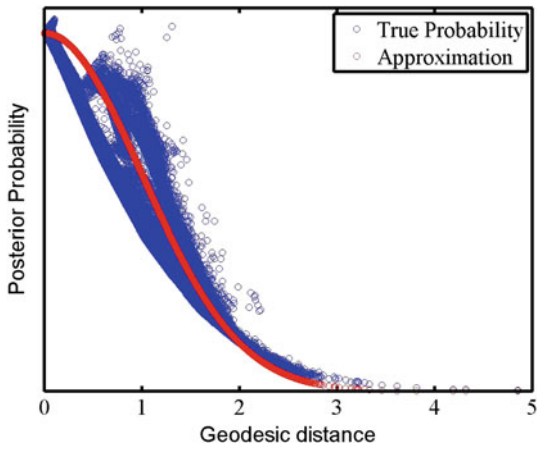


Fig. 5 The approximation of the exponential function in $W'_{\mathbf{x},l}$ to the real posterior probability

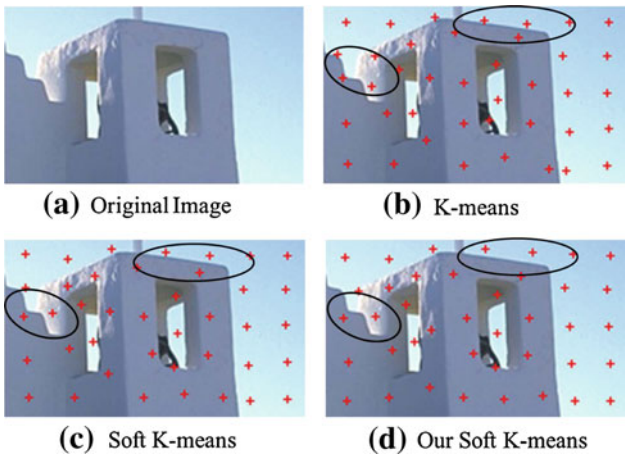


Fig. 6 Relocated centers (*plus*) from different segmentation strategy. In the *black circle* of **b**, K-means locates the centers over the boundary because of the boundary points. While in *circle* of **c**, soft k-means avoids such defects. In **d**, our approximated weight could achieve the same effect

Fig. 6a by K-means, soft K-means, and our approximated soft K-means. As can be seen in Fig. 6b, K-means clustering does not consider the membership, and easily locates the centers over the image edges (due to the noisy points illustrated in Fig. 4), which will negatively affect the results in our experiments. In our experiments, because of the splitting strategy, K-means would generate many useless centers near the cluttered edges. However, as in Fig. 6c, soft k-means locates the center at more visually pleasing positions while it costs huge computation time by computing the distance between each pixel to each center. For efficiency, we use the weight in Eq. (12) and adjust the ε through a validation set. In Fig. 6d, though we did not perform the time costly normalization, the approximation locates the centers well by avoiding the edges. Moreover, in our experiments, combining with the splitting strategy, we achieve close optimized results with soft k-means.

3.2 Center Refinement

Given a set of superpixels $L(\mathbf{x})$, the goal of this step is to re-estimate the centers' positions $\{\mathbf{c}_l\}$ according to E_{total} in Eq. (8).

The location of each center should be updated by:

$$\mathbf{c}'_l = \arg\min_{\mathbf{x}' \in I(\mathbf{x})} \int_{I(\mathbf{x})} \hat{W}_{\mathbf{x},l} D'_g(\mathbf{x}', \mathbf{x})^2 d\mathbf{x}. \tag{14}$$

An exhaustive search as in Feil and Abonyi (2007) is computational costly and infeasible for this iterative approach. Based on calculus, $\mathbf{c}'_l$ is a stationary point where the derivative of E_{image} equals to $\mathbf{0}$, which leads to:

$$\frac{\partial E_{image}}{\partial \mathbf{c}'_l} = 2 \int_{S_l} \hat{W}_{\mathbf{x},l} D'_g(\mathbf{c}'_l, \mathbf{x}) \nabla D'_g(\mathbf{c}'_l, \mathbf{x}) d\mathbf{x} = \mathbf{0}. \tag{15}$$

Nevertheless, it is intractable to solve the derivative equation to get an analysis solution because $\nabla D'_g(\mathbf{c}'_l, \mathbf{x})$ can not be written explicitly. As a compromise, we use the approximate solution to replace the exact solution for computational efficiency. Similar to fixed point algorithm (Hyvärinen 1999), in the current updating iteration, we use the result of $\mathbf{c}_l$ from the last updating iteration to compute the part $D_g(\mathbf{c}'_l, \mathbf{x})$. Moreover, to find the gradient of geodesic distance, $\nabla D_g(\mathbf{c}'_l, \mathbf{x})$, we substitute the $D_g(\mathbf{c}'_l, \mathbf{x})$ with the Euclidean distance $v\|\mathbf{x} - \mathbf{c}'_l\|$ for a tractable solution. This is under the assumption that the appearance within one superpixel in an image is more likely to be homogeneous, i.e. with a constant velocity being a small value v, which makes the geodesic distance homogeneous with the Euclidean distance.

Then, the derivative with respect to $\mathbf{c}_l$ becomes:

$$\begin{aligned} \frac{\partial E_{image}}{\partial \mathbf{c}'_l} &\cong 2 \int_{S_l} \hat{W}_{\mathbf{x},l} D'_g(\mathbf{c}_l, \mathbf{x}) \nabla \|\mathbf{x} - \mathbf{c}'_l\| d\mathbf{x} \\ &\cong -2 \int_{S_l} \hat{W}_{\mathbf{x},l} D'_g(\mathbf{c}_l, \mathbf{x}) \frac{(\mathbf{x} - \mathbf{c}'_l)}{\|\mathbf{x} - \mathbf{c}_l\|} d\mathbf{x} \\ &= \mathbf{0}, \end{aligned} \tag{16}$$

where we additionally apply the trick of taking place of $\|\mathbf{x} - \mathbf{c}'_l\|$ by $\|\mathbf{x} - \mathbf{c}_l\|$ computed in the previous iteration for solving $\mathbf{c}'_l$.

By solving the equation, the result has a similar form as the computation of the centroid, i.e. the center of mass, of the segment S_l with its center being $\mathbf{c}'_l$ and its mass equal to $\int_{S_l \setminus \{\mathbf{c}_l\}} \hat{W}_{\mathbf{x},l} \frac{D'_g(\mathbf{c}_l, \mathbf{x})}{\|\mathbf{x} - \mathbf{c}_l\|} d\mathbf{x}$. The new center relocates to:

$$\begin{aligned} \mathbf{c}'_l &= \frac{\int_{S_l \setminus \{\mathbf{c}_l\}} m_{(\mathbf{x}|L(\mathbf{x})=l)} \mathbf{x} d\mathbf{x}}{m_l}, \\ \text{with} \quad m_{(\mathbf{x}|L(\mathbf{x})=l)} &= \hat{W}_{\mathbf{x},l} \frac{D'_g(\mathbf{c}_l, \mathbf{x})}{\|\mathbf{x} - \mathbf{c}_l\|}, \\ m_l &= \int_{S_l \setminus \{\mathbf{c}_l\}} m_{(\mathbf{x}|L(\mathbf{x})=l)} d\mathbf{x}. \end{aligned} \tag{17}$$

Notice that v should be given a positive value to achieve a distance increment, and here we set $v = 0.01$. To validate the proximation of the distance, we conducted several experiments to test the energy term from E_{total}. The results shown in Fig. 7 clearly demonstrate that the re-estimation of the centers decreases the defined energy.

3.2.1 Center Splitting

As mentioned in Sect. 1, one of the main goals is to generate superpixels that are sensitive to image structure (see Eq. (6)).

During the energy minimization process, given a superpixel S_l whose area A_l is much larger than $\overline{A}$ while its center $\mathbf{c}_l$ shifts little from last iteration, the algorithm splits the center $\mathbf{c}_l$ into two since the later generated segments by the new ones would produce a lower value of the energy functional in Eq. (8). Such a process accelerates the process for finding the optimal centers much quickly.

Divisive clustering algorithms has been well discussed in Savaresi and Boley (2004). There are mainly two strategy for splitting: the bisecting K-means algorithm and the principal direction divisive partitioning (PDDP). Such schemes increase one cluster each time and suffer from the defect of propagation error from upper levels. In our case, increasing one cluster at one time is inefficient and the propagation error is also undesired. Considering both efficiency and accuracy, we choose to bisect the superpixel whenever it meets our splitting criteria which are heuristically generated based-on

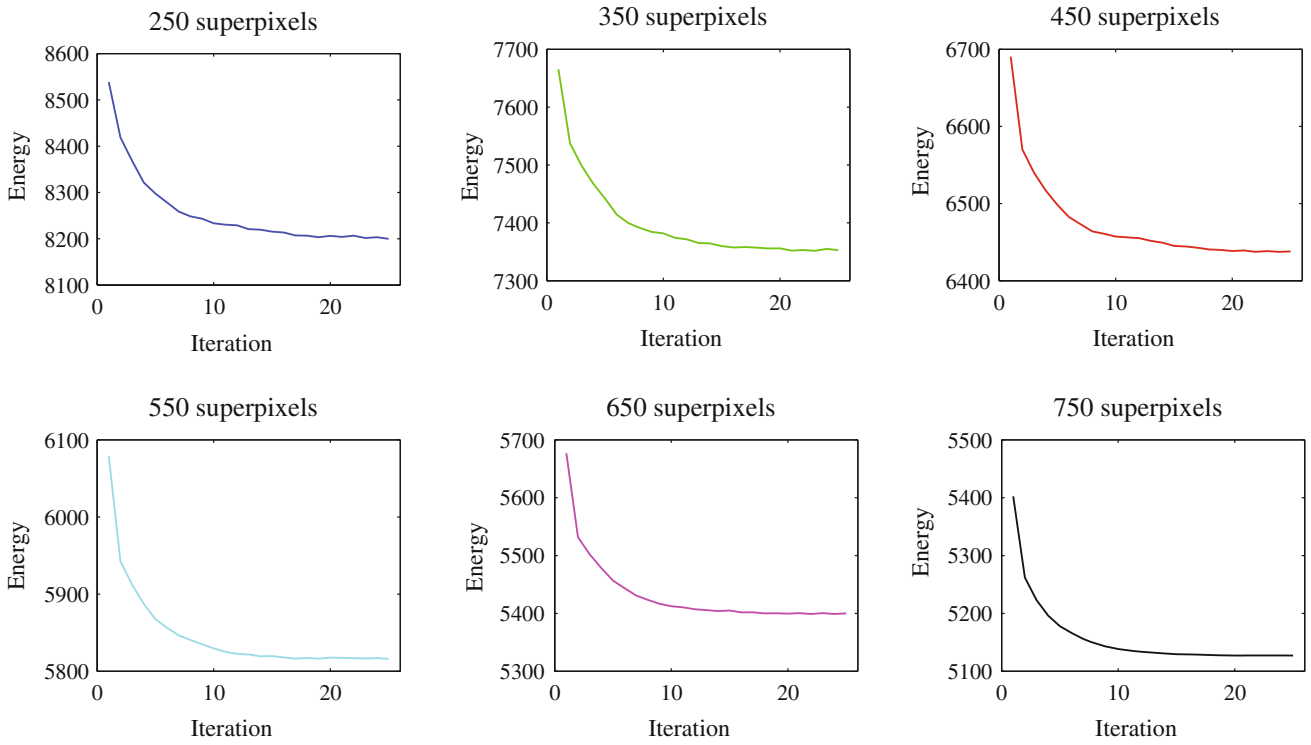


Fig. 7 The E_{total} is dropping down by using our relocation strategy under different superpixels number

the human intuition and E_{total}, and we propose to re-estimate all the centers' location after splitting during the optimization.

Thus, we define criteria to distinguish superpixels' centers to be split and those to be relocated:

$$C_{shape}(\mathbf{c}_l) = \mathbb{1}\left(\frac{\lambda_{1,l}}{\lambda_{2,l}} > T_c\right);$$
$$C_{size}(\mathbf{c}_l) = \mathbb{1}\left(\frac{A_l}{\overline{A}} > T_s\right); \tag{18}$$

where $\mathbb{1}$ is the indicator function and equals to 1 if the inside criterion is true. The center l is marked as a splitting candidate if either of C_{shape} or C_{size} is positive. T_c and T_s are thresholds, while $\lambda_{1,l}$ and $\lambda_{2,l}$ are the first and second eigenvalue obtained by the PCA (Jolliffe 1986) of the following 2×2 matrix:

$$\int_{S_l \setminus \{\mathbf{c}_l\}} \frac{D'_g(\mathbf{c}_l, \mathbf{x})^2}{\|\mathbf{x} - \mathbf{c}_l\|^2} (\mathbf{x} - \mathbf{c}_l)(\mathbf{x} - \mathbf{c}_l)^T d\mathbf{x}. \tag{19}$$

Furthermore, in color images, we make use of the color homogeneity within a superpixel, which is also an important factor for judging the splitting. In our experiments, we limit the standard deviation of color within each superpixel under normalized CIElab color space which is known for having distance between colors similar to the human visual system. In addition, CIElab color space is also widely applied in many vision tasks such as Shotton et al. (2006) etc.

Specifically, our criterion for color variation is:

$$C_{Var}(\mathbf{c}_l) = \mathbb{1}(SD_{S_l} > SD_{I(\mathbf{x})}/(\varepsilon_v N_{iter})), \tag{20}$$

where SD_{S_l} and $SD_{I(\mathbf{x})}$ are the standard deviation of color within superpixel S_l and on the whole image $I(\mathbf{x})$ respectively, ε_v is a constant and set to be 0.015 and N_{iter} is the center number in current iteration. Intuitively, we wish the constraint is stricter at the beginning iterations making the algorithm split the center having large variation, then handle the rest having smaller variation latter.

With the splitting candidates, our algorithm performs the splitting from the center that has a largest average normalized score over the three criteria. In practice, this could lead to the better dropping of the energy. After splitting of center $\mathbf{c}_l$, two new centers $\mathbf{c}'_{l,1}$ and $\mathbf{c}'_{l,2}$ are generated to split and replace the current one $\mathbf{c}_l$ by calculating:

$$\mathbf{c}'_{l,1} = \frac{\int_{\{\mathbf{x}|\mathbf{x}\in S_l, (\mathbf{x}-\mathbf{c}_l)\cdot\mathbf{n}>0\}} \frac{D_g(\mathbf{c}_l,\mathbf{x})}{\|\mathbf{x}-\mathbf{c}_l\|}\mathbf{x}d\mathbf{x}}{\int_{\{\mathbf{x}|\mathbf{x}\in S_l, (\mathbf{x}-\mathbf{c}_l)\cdot\mathbf{n}>0\}} \frac{D_g(\mathbf{c}_l,\mathbf{x})}{\|\mathbf{x}-\mathbf{c}_l\|}d\mathbf{x}},$$
$$\mathbf{c}'_{l,2} = \frac{\int_{\{\mathbf{x}|\mathbf{x}\in S_l, (\mathbf{x}-\mathbf{c}_l)\cdot\mathbf{n}<0\}} \frac{D_g(\mathbf{c}_l,\mathbf{x})}{\|\mathbf{x}-\mathbf{c}_l\|}\mathbf{x}d\mathbf{x}}{\int_{\{\mathbf{x}|\mathbf{x}\in S_l, (\mathbf{x}-\mathbf{c}_l)\cdot\mathbf{n}<0\}} \frac{D_g(\mathbf{c}_l,\mathbf{x})}{\|\mathbf{x}-\mathbf{c}_l\|}d\mathbf{x}}, \tag{21}$$

where $\mathbf{n}$ denotes the corresponding eigenvector of $\lambda_{1,l}$.

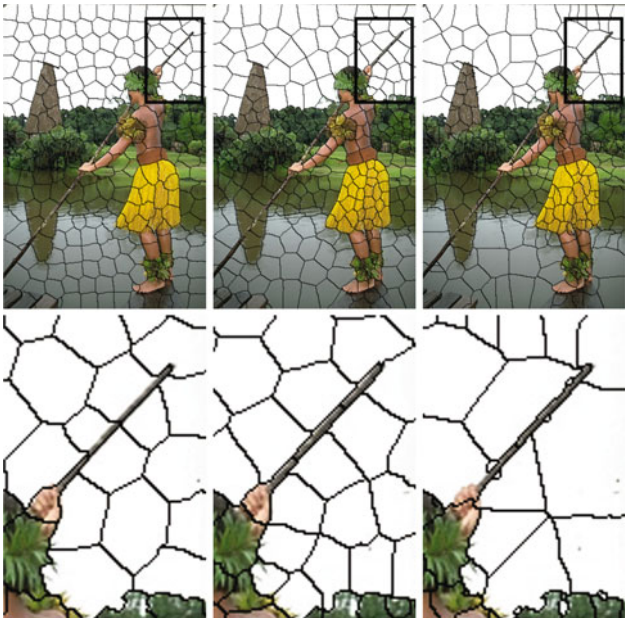


Fig. 8 The same image segmented into 250 superpixels using **a** only relocation, **b** both relocation and splitting, and **c** only splitting. We zoomed in the area of interest in the *second row*

Besides the rare cases in which no splitting criterion is met while the demanding number of superpixels has not been reached, we selected the largest few superpixels (10 in our implementation) to do the splitting. In a nutshell, the energy functional, i.e. E_{total}, keeps decreasing.

Additionally, to validate the effectiveness of the splitting strategy, we conducted the superpixel segmentation by only relocation, relocation and splitting, and only splitting as shown in Fig. 8. The relocation finds more reasonable positions of centers and the splitting makes the segmentation structure-sensitive. This is also shown in the detail region indicated by the black rectangle that we zoomed in. The boundaries of the brown stick held by the person could hardly be captured without the splitting operation, because it is too thin for a center to be located inside. However, the result is also undesired if the positions of centers are not relocated, i.e. the centers are not well-distributed near the stick. Generally, in the comparison, our most pleasing result is achieved by combining the splitting strategy with relocation.

3.3 Initialization and Termination

3.3.1 Initial Seeds Placement

One way to place the initial seeds is similar as TurboPixels in Levinshtein et al. (2009b), K initial seeds are placed in a lattice formation such that the distance between neighbor seeds is roughly equal to $\sqrt{M/K}$, where M is the total pixel number of the image. They also perturb the seeds by moving them away from the pixels with high gradient magnitude to avoid strong intensity boundaries and bad initialization for later iterations. Different from TurboPixels algorithm, we alternatively set K to be a portion of the total number of superpixels N (specified by users). During the optimization process, additional superpixels are generated by splitting existing ones until the number of superpixels reaches N.

The other available initialization scheme is to sample the K seeds based on the density map $D(\mathbf{x})$ on the image. This would make the initial energy lower than lattice layout as structure-sensitive layout, thus the functional would converge faster. In practice, we sample the seeds from a special designed distribution calculated from $D(\mathbf{x})$. Mathematically, a Gaussian kernel G_{σ_s} is combined as:

$$D_{sample}(\mathbf{x}) = \frac{1}{Z_D}(G_{\sigma_s} * (D(\mathbf{x}) + \lambda\overline{D(\mathbf{x})}), \tag{22}$$

where $D(\mathbf{x})$ is defined in Eq. (2), the σ_s of our Gaussian kernel is set to be $2\sqrt{M/K}$, which generally guarantees the minimum area A_l generated by the initial seeds larger than $\overline{A}$ (Due to the splitting scheme, superpixels could not be merged in later iterations). $\overline{D(\mathbf{x})}$ is the mean value of $D(\mathbf{x})$, which makes sure that there exists a certain density of initial seeds placed on homogeneous regions. The Z_D is a normalization operator which makes the integral of D_{sample} equal to 1. Thus $D_{sample}(\mathbf{x})$ could be taken as a two dimensional probability distribution.

Furthermore, for better formation of the initial seeds, our sampling method generates a projection $\mathbf{c}_s \rightarrow \mathbf{c}_t$ between the lattice placement of seeds $\mathbf{c}_s = [x_s, y_s]$ from the first placement scheme to the new placement $\mathbf{c}_t = [x_t, y_t]$. Formally, the $[x_s, y_s]$ and $[x_t, y_t]$ are correlated by:

$$\begin{aligned} \frac{\int_1^{x_t} D_{sample}(x, y_t)dx}{\int_1^{w} D_{sample}(x, y_t)dx} &= \frac{x_s}{w}; \\ \frac{\int_1^{y_t} D_{sample}(x_t, y)dy}{\int_1^{h} D_{sample}(x_t, y)dy} &= \frac{y_s}{h}. \end{aligned} \tag{23}$$

From Eq. (23), when only looking at the first one, given any y_t, we can get a x_t in response. Then a curve extending in y direction could be generated. Similarly, the second equation produces a curve in x direction. The $\mathbf{c}_t$ is the intersection point of the two curves. Notice that for the σ_s controls the variety of the D_{sample}, and determines the tortuosity of the curves, in most cases, only one solution could be found for each $\mathbf{c}_s$. In the case that multiple solutions are existing, we chose the one with the lowest density value.

As shown in Fig. 9, the seeds sampled by such scheme generates a visually pleasing estimation of the final structure-sensitive position of all the centers. Additionally, the initial seeds are also well-separated to keep compactness of superpixels according to the lattice starting formation and our sampling density.

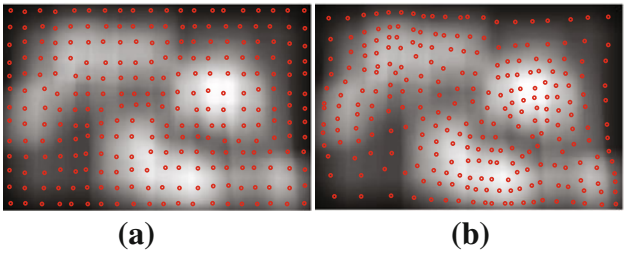


Fig. 9 Initial seeds sample scheme: **a** regular seeds placement with perturbation and **b** sampled seeds based on the edge density distribution

3.3.2 Termination Conditions

We use the following termination conditions: (1) the change of energy between two successive iteration steps is less than a threshold ε_E; (2) the total number of iterations exceeds the predefined number N_{max}.

In the final stage, very small superpixels are detected and removed, which results in a small number of unassigned pixels. The final superpixel result is generated by the over-segmentation (in Sect. 3.1) with the remaining centers.

3.4 Algorithm Complexity and Convergence

As the algorithm iteratively performs two routines in turn, it is easily known that the time complexity of our algorithm is $O((T_{segment} + T_{center})N_I)$, where $T_{segment}$ and T_{center} are the complexities of the over segmentation in Sect. 3.1 and center refinement in Sect. 3.2 respectively. N_I is the total number of iterations.

Let M denote the number of pixels on an image. The complexity of the fast marching can be decreased to roughly $O(M)$ (Yatziv et al. 2006). It can be also proven that T_{center} is $O(M)$, since the center refinement can be achieved by a single scan of all pixels on an image. Thus, the complexity of the whole algorithm becomes $O(MN_I)$.

Experiments show that the algorithm terminates within 30 iterations. The number of centers increases quickly over the first several iterations when over-segments have larger sizes, which makes the energy functional decrease rapidly. Figure 10 shows the energy functional decreases with each iteration of the algorithm. The number of iterations rarely exceeds $N_{max} = 30$. With such a constraint, the complexity of the algorithm approaches $O(M)$.

4 Alternative Strategy for Optimization

4.1 Center Merging

Merging adjacent superpixels with very similar appearance is another strategy to make the superpixels' area represent the structure of image. Based on the structure term in Eq. (8),

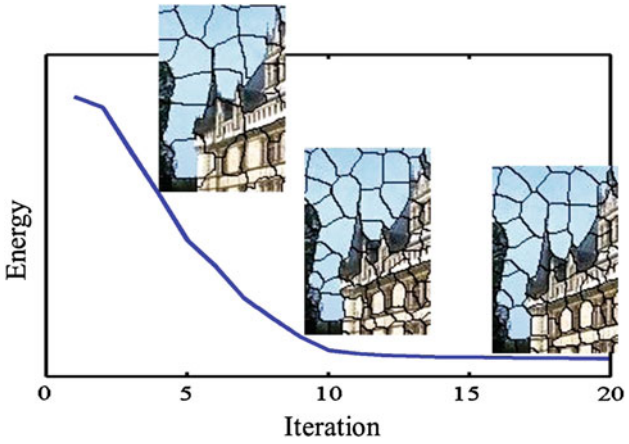


Fig. 10 Algorithm convergence. The overall energy functional decreases and the superpixels perform better with every iteration

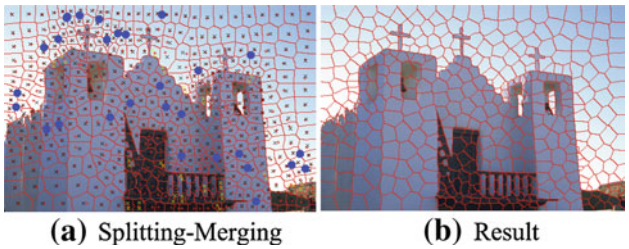


Fig. 11 An example of the Splitting-Merging scheme: **a** the *blue filled circle* represents the merged centers and the *yellow cross* means the splitting centers' position. **b** The result from splitting and merging scheme (Color figure online)

merging a pair of adjacent superpixels in low density regions of image contents while split a superpixel in high density drops the energy.

Previous art (Muhr and Granitzer 2009) combines the splitting and merging together for searching the optimized cluster number. We here adopt splitting and merging together to search for better superpixel structure consistent with the image density. In our attempting, we exploit a heuristic design and propose to perform merging simultaneously with splitting during the iteration optimization.

Figure 11 illustrates an example of merging and splitting procedure and the final segmentation result starting with 500 seeds sampled from the sampling density distribution defined in Eq. (22). During the optimization procedure, merging scheme should be designed to make sure that the optimization will not fall into a loop or flip over, i.e. to avoid splitting previously merged superpixels or merging any split ones.

Based on this consideration, two adjacent centers $\mathbf{c}_p, \mathbf{c}_q$ in one iteration are considered as candidates with potential to merge according to the following criteria:

$$C_{stable}(\mathbf{c}_p, \mathbf{c}_q) = \mathbb{1}(\|\mathbf{c}_l - \mathbf{c}'_l\| < T_{shift},\ \forall\, l \in \{p, q\});$$
$$C_{area}(\mathbf{c}_p, \mathbf{c}_q) = \mathbb{1}\left(\frac{A_p + A_q}{\overline{A}} < 1\right). \tag{24}$$

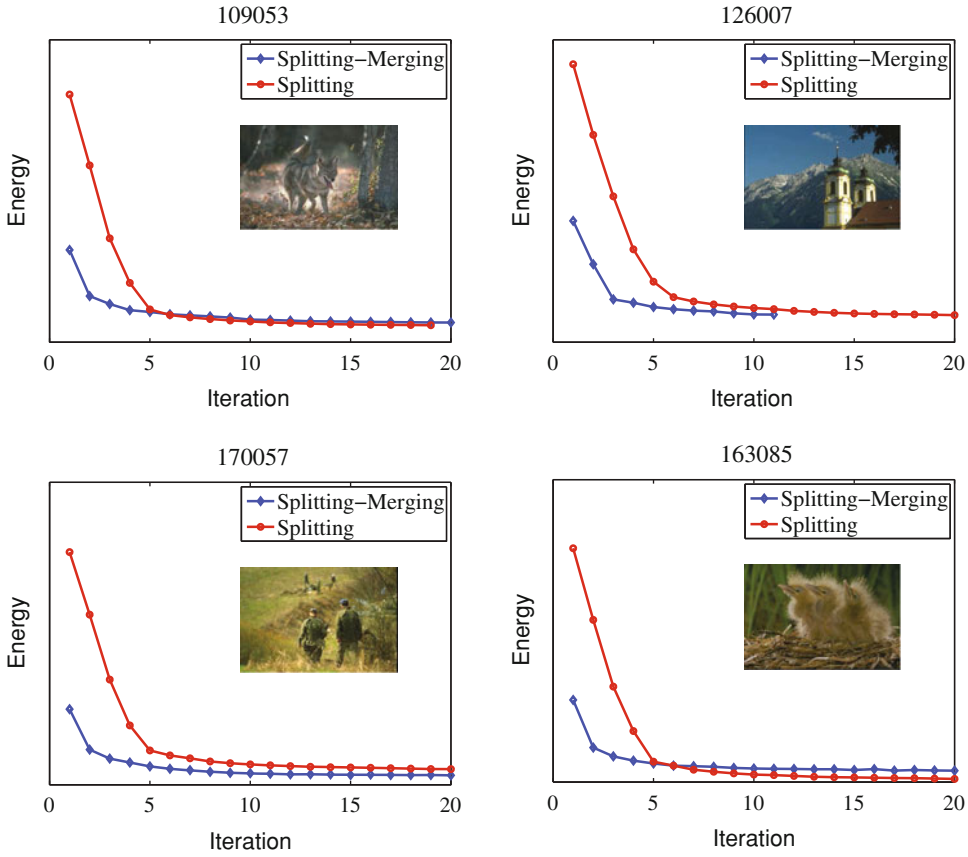


Fig. 12 The optimization process of Splitting scheme and Splitting-Merging scheme

Similar to the splitting criterion in Eq. (20), we consider the color homogeneity within the area covered by S_p and S_q for color images. Formally, our color variation measurement is defined as:

$$C_{dis}(\mathbf{c}_p, \mathbf{c}_q) = \mathbb{1}(Dis(\mathbf{c}_p, \mathbf{c}_q) < SD_{I(\mathbf{x})}/\varepsilon_v N), \tag{25}$$

where $SD_{I(\mathbf{x})}$ is the same as that in Eq. (20). We define that $ADJ(S_p, S_q)$ is the adjacent boundary length of S_p and S_q and $C(S_l)$ is the contour length of S_l. The distance embedded color homogeneity and adjacent length is calculated by $Dis(\mathbf{c}_p, \mathbf{c}_q) = \exp\left\{\frac{-ADJ(S_p,S_q)}{(C(S_p)+C(S_q)-ADJ(S_p,S_q))}\right\} SD_{(S_p \cup S_q)}$. We also regard $\mathbf{c}_p, \mathbf{c}_q$ as merging candidates if $C_{dis}(\mathbf{c}_p, \mathbf{c}_q)$ is positive.

Then, we rank the distance (i.e.$Dis(\mathbf{c}_p, \mathbf{c}_q)$) in ascending order and greedily merge the adjacent superpixels while keeps the number of superpixels larger than N_M. We set $N_M = 0.8N$ in our experiments. Though it is proven in Muhr and Granitzer (2009) that merging clusters would always increase the energy, we show in our experiments the energy could keep decreasing in most cases in Fig. 12 because the splitting and relocation strategy included could drop down the energy much more than the little increment brought by our merging strategy.

If two centers are merged, a new center is generated $\mathbf{c}_m$ and replaces the original two by computing the "centroid" of the two superpixels:

$$\mathbf{c}_m = \frac{m_p \mathbf{c}_p + m_q \mathbf{c}_q}{m_p + m_q}. \tag{26}$$

Additionally, to keep the balance of area between different superpixels, once two centers are merged, neither of the two can merge again in the current iteration. To further avoid the dead loop, the merged centers can not split for the next five iterations.

Unlike the center splitting, for initialization, we directly place N seeds according to the sampling scheme in Sect. 3.3.

4.2 Discussion Between Splitting and Splitting-Merging

Splitting-Merging scheme in Sect. 4.1 has three merits: (1) with both splitting and merging, the algorithm could converge in fewer iterations as shown in Fig. 12; (2) it makes the algorithm less dependent on the initial fraction K of seeds as required by the splitting strategy and thus cutting down the number of user setting parameters; and (3) some superpixels with small or tiny area are given a further possibility to merge with nearby superpixels. On the other hand,

(a) 6th Iter. **(b)** 11th Iter. **(c)** 18th Iter.

Fig. 13 An example of fixed centers during the iteration, the *blue filled circle* indicates the fixed centers that need no computation in the respective iteration (Color figure online)

Splitting-Merging scheme causes extra computational cost at each iteration, while Splitting requires no computation related to the relationship with neighborhood.

In general, the choices of superpixels that need to be split are much fewer than the ones needs to be merged after few iterations, which makes the Splitting more efficient than the Splitting-Merging in single iteration. In our experiments, though fewer iterations is needed by Splitting-Merging, the Splitting achieves higher efficiency.

In addition, from the comparison by our experiment in Sect. 5, the performance of Splitting (with K initialization seeds) and Splitting-Merging (with N initialization seeds) are very close under our parameters and experiment settings, which is consistent with the energy comparison between the two methods shown in Fig. 12. Actually, the energy curves from Splitting and Splitting-Merging algorithm are not always converging to the same value as the local minimum of the energy functional. However, in our experiments, we show that the performances of both algorithm are visually close.

4.3 Acceleration Scheme for Optimization

There are some room to further accelerate the iterations when taking a closer look at the algorithm. As indicated by the blue points in Fig. 13, after a few iterations, many centers shifted little and had no potential to split, and updating such centers has little influence on the final results. Thus in our implementation, it is meaningful to skip the calculation of such superpixels. In other words, fixing certain superpixels' center positions can reasonably increase computational efficiency. Here we defined the following two rules for center fixation:

1. The center $\mathbf{c}_l$ itself shifts little from the position in last iteration, i.e. $\|\mathbf{c}_l - \mathbf{c}'_l\| \leq T_{shift}$ and does not meet the splitting (or merging) criteria.
2. All of the center's adjacent neighbor centers meet the first rule.

Once a center is fixed, the geodesic distance and center refinement of the superpixel will not be updated in later iterations. Nevertheless, we wish that the fixed centers could be reactivated for optimization when their neighborhood superpixels changes significantly. Thus during the iterations, the labels of the neighborhood of each fixed center would be recorded, and the fixed center would be reactivated if any of its neighborhood violates the first rule as defined. Figure 14 visualizes the fixed centers and percentage of pixels included in calculation during the optimization process. As can be seen, with the increasing of fixed centers, the amount of pixels that are actually computed decreases rapidly, so is the computational cost. In our experiments, by conducting such acceleration, the computing time generally cuts down from nearly 10 to 5–6 s for images of size 481×321.

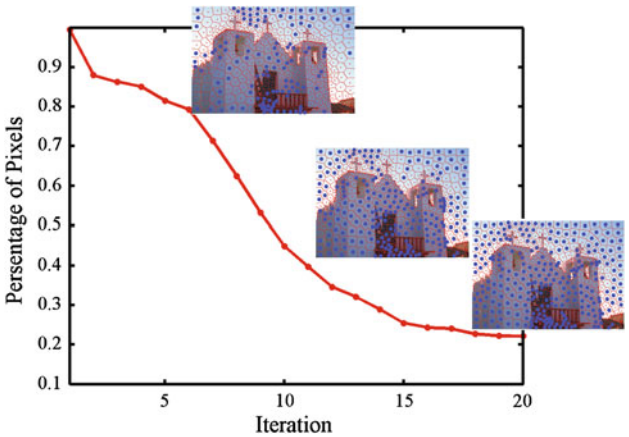


Fig. 14 The percentage of pixels that needs to be computed during the iterations

5 Experimental Evaluations

5.1 Parameter Settings

In all experiments, our algorithm is not sensitive to most of the parameters, such as the standard deviation σ and γ in Eq. (2). We set the σ adaptively as $\frac{\sqrt{M/N}}{2}$ and the γ as 0.12, where M is the total number of pixels in the image and N is the user-specified number of superpixels. For the some sensitive parameters, we constructed a validation set including 20 training images randomly chosen from BSD300 data set (Martin et al. 2001) and tune the parameters based

on the performance over these images. We set ε in Eq. (13) to be 2, $T_s = 2$, and $T_c = 4$ for criteria in Eq. (18) and the $T_{shift} = 2$ in Eq. (24).

In addition, we investigated an the initial seed number place at initialization. By changing the initial placed seeds fraction, i.e. the ratio of placed initial seeds number K to user-request seeds number N, Fig. 15 shows the testing results (using criteria in Sect. 5.2). We can see that neither a too small nor too large fraction of initial seeds performs well, for the reason that a too small fraction makes the superpixels less compact and a too large fraction makes the superpixels' layout less structure-sensitive and sub-optimal. Another difference is that the larger the fraction, the less time our algorithm needs to converge. In our experiment, we set the initial seeds to $N/3$ for splitting, which generally ensures the minimum area A_l in the first iteration larger than $\overline{A}$ and performs the best.

5.2 Quantitative Evaluation

We evaluated the performance of the proposed algorithm by comparing its accuracy with several leading approaches: TurboPixels (Levinshtein et al. 2009b), N-Cuts (Shi and Malik 2000), Graph-based method (Felzenszwalb and Huttenlocher 2004), Lattice (Moore et al. 2008), GraphCut superpixel (Veksler et al. 2010) and SLIC superpixel (Radhakrishna et al. 2010). We also show the evaluation of the Splitting-Merging algorithm, in which we evaluate the performance with N initial seeds. Finally, to explicitly illustrate the effect of the iterative optimization, another baseline is constructed by single fast-marching using the sampled N initial seeds based-on Eq. (22), which we call "Sample Seeds".

We use the Fast Marching Toolbox[1] to compute geometric flows. The Multi-scale Normalized Cuts Segmentation Toolbox[2] is applied for N-Cuts. We downloaded the TurboPixels implementation[3] for TurboPixels, the Graph-Based method implementation[4] online, the GraphCut superpixel implementation[5] and the Lattice.[6] In all testing, we use the author's raw implementation and tune their parameters on a small validation set.

All experiments are performed on a quad-core 3.2 GHz computer, and the evaluation is based on the BSD300 data set (Martin et al. 2001), which contains 100 test images and 200 training images with 481×321 (or 321×481) pixel resolution. The performance is averaged over a random subset (50 images) of the test set for the high computational cost of N-Cuts.

All the above mentioned algorithms keep compactness except Graph-based method. We compare ours with these algorithms in following quantitative criteria.

5.2.1 Under-Segmentation Error

Intuitively, under-segmentation error penalizes the superpixels that do not overlap tightly with a ground truth segmentation. Given a ground truth segmentation into segments $\mathcal{G} = \{G_1, \ldots, G_K\}$ and a superpixel segmentation into superpixels $\mathcal{S} = \{S_1, \ldots, S_L\}$, we quantify the under-segmentation error of a whole image as:

$$U_{\mathcal{G}} = \frac{1}{M}\left[\sum_{k=1}^{K}\left(\sum_{\{S_l||S_l\cap G_k|>B\}} Area(S_l)\right) - M\right], \quad (27)$$

where $Area(S_l)$ is the area of the superpixel, and M is the total number of pixels. B is the minimum area of overlapping, and we follow Radhakrishna et al. (2010) setting B to be 5 % of $Area(S_l)$.

We averaged the value U across all our test images and all ground-truth segments, and obtained a comparison result in Fig. 16a. As can be seen, our algorithm outperforms other methods, especially with small numbers of superpixels.

5.2.2 Boundary Recall

A standard boundary recall measurement is also adopted, which computes what fraction of the ground truth edges fall within ε-pixel length from at least one superpixel boundary. Mathematically: the boundary recall is:

$$B_{\mathcal{G}} = \frac{\sum_{\mathbf{p}\in\delta\mathcal{G}} \mathbb{1}(min_{\mathbf{q}\in\delta\mathcal{S}}\|\mathbf{p}-\mathbf{q}\| \leqq \varepsilon)}{\mid \delta\mathcal{G} \mid}, \quad (28)$$

where $\delta\mathcal{G}$ and $\delta\mathcal{S}$ denote the union set of ground truth boundaries and superpixels boundaries respectively, and the indicator $\mathbb{1}$ checks whether the nearest pixel is within distance ε. We also follow previous art (Radhakrishna et al. 2010) and set $\varepsilon = 2$ in our experiment.

The comparison of the boundary recall of some state-of-the-art methods and our method is in Fig. 16b. From the results, ours outperforms the competitors such as TurboPixels and N-Cuts while remaining comparable to the Graph-based method which has high boundary recall.

5.2.3 Achievable Segmentation Accuracy (ASA)

The ASA is a performance measurement which indicates the upper bound of segmentation, which gives the highest

[1] The Fast Marching Toolbox is written by Gabriel Peyre.

[2] The Multi-scale Normalized Cuts Segmentation Toolbox is written by Timothee Cour et al.

[3] The TurboPixels toolbox is written by Alex Levinshtein.

[4] The Graph-Based method toolbox is written by Felzensz.

[5] The GraphCut superpixel implementation is written by Olga Veksler.

[6] The Lattice superpixel is written by Alastair P. Moore.

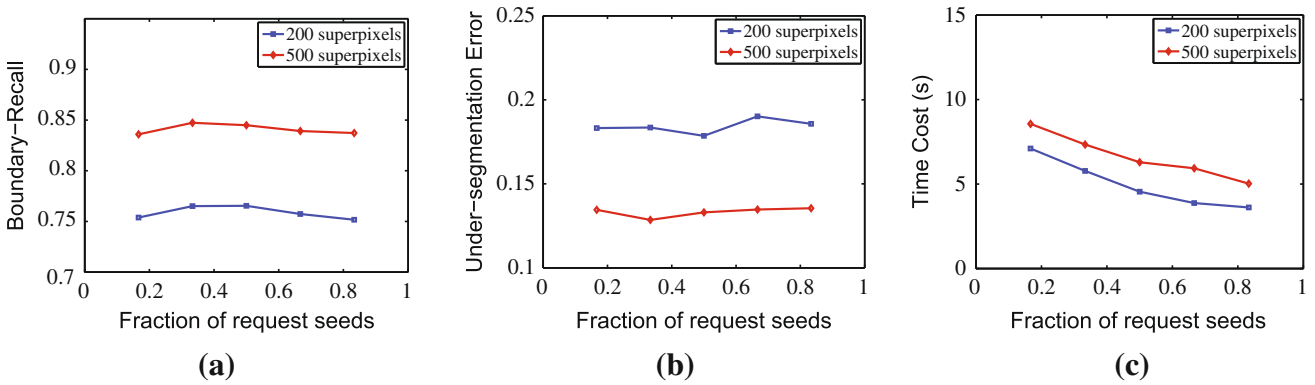


Fig. 15 Performance related to initial fraction of seeds given by user: **a** under-segmentation error, **b** boundary recall, and **c** time cost

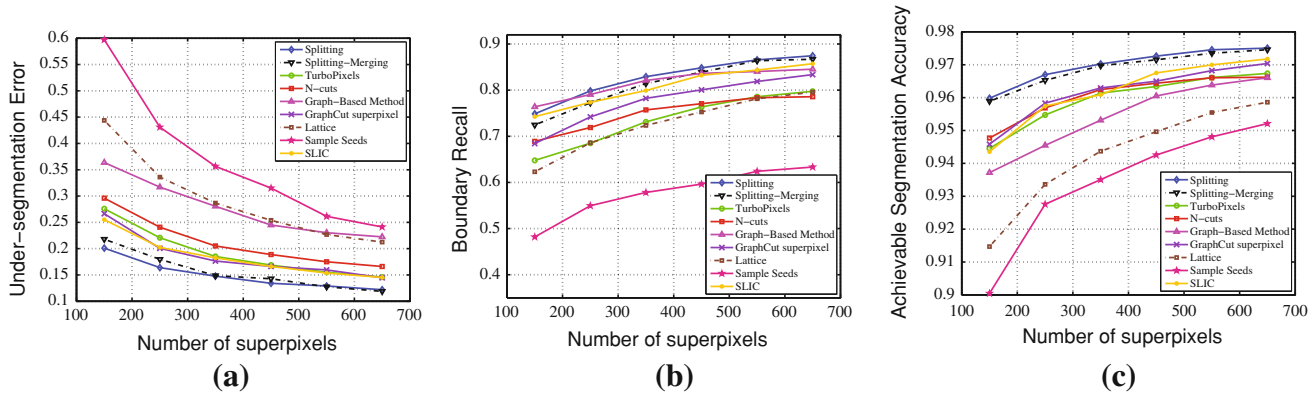


Fig. 16 Performance comparison between Splitting scheme, Splitting-Merging scheme, TurboPixels, N-Cuts, Graph-based method, GraphCut superpixel, Lattice, the Sample Seeds and SLIC: **a** under-segmentation error, **b** boundary recall, and **c** achievable segmentation accuracy

accuracy achievable for object segmentation that utilizes superpixels as units. It is computed by:

$$ASA_{\mathscr{G}} = \frac{\sum_k max_i |\, S_k \cap G_i \,|}{\sum_i |\, G_i \,|}. \tag{29}$$

Such metric is also plotted against number of superpixels in an image. The algorithm achieves a higher score when a smaller number of superpixels is preferred.

Figure 16c shows the comparison result between our approach and other algorithms, and we can see the proposed method yields a better achievable segmentation upper-bound.

5.2.4 Time Cost

As demonstrated in Levinshtein et al. (2009b), TurboPixels is much faster than N-Cuts and also exploits geometric flow for segmentation. We thus conducted time comparisons with TurboPixels. In the meanwhile, we also include Graph-based method, GraphCut superpixel and superpixel lattice for providing a whole view of relative time comparison of various superpixel methods. In our experiments, the running time of the comparing algorithms is tested with respect to image size and superpixel number.

The result in Fig. 17a shows that our algorithm terminates within linear time with respect to the image size, which has also been proven in Sect. 3.4. With the acceleration, our algorithm outperforms the TurboPixels in time. Figure 17b shows the running time when increasing superpixel density under a constant image size (241×161) in our experiments. The running time of our algorithm slightly increases. This is mainly because more iterations are required for minimization with a larger number of superpixels. As demonstrated in Sect. 3.4, the running time is a linear function with respect to number of iterations.

5.3 Qualitative Results

Figure 18 shows a qualitative comparison of the superpixel obtained by TurboPixels and our method in a variety of images from BSD300. The number of superpixels generated by TurboPixels and our method is almost the same. As can be noticed, the density of superpixels provided by our method is quite consistent with the image contents: the density is low in the homogenous regions and high near high intensity boundaries. This makes the superpixel boundaries react to salient edges better.

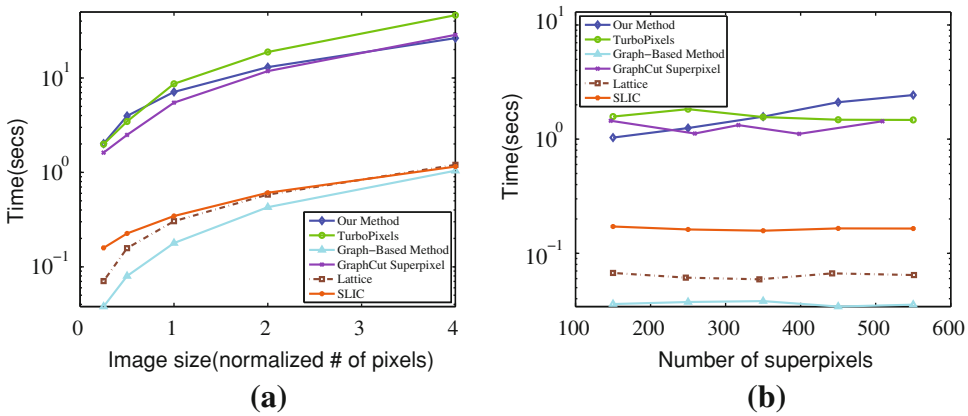


Fig. 17 Timing comparison with TurboPixels. **a** running time with respect to image size and **b** running time with respect to the density of superpixels

5.3.1 Combination with Supervised LEARNT EDGE Maps

Similar to the prior arts (Levinshtein et al. 2009b; Moore et al. 2008), our algorithm is not constrained to the image-gradient-based density functions. Different kinds of refined measures can be easily combined in the velocity function for calculating the geodesic distance. Figure 19 shows the performance of our algorithm when combined with the Pb-based (Martin et al. 2004) boundaries. The edges between the tiger and background are much better captured.

Furthermore, we investigate whether the segmentation results can be further improved by using edge maps extracted through learning methods. Then we collected various edge detectors including gPb (Maire et al. 2008), ucm (Arbelaez et al. 2009), BEL (Dollár et al. 2006), pb (Martin et al. 2004), and compared the results with gradient-based edge and linear combination of gradient and gPb edge map. Figure 20 shows the results using the evaluation methods in Sect. 5.2. From Fig. 20, the learned edge map performs better with smaller superpixels number, however, with the number of superpixels increasing, the results from the gradient map improve relatively faster and approximate the best when the number of superpixels becomes 650. This is mainly because learnt edges sometimes dismiss many local edges which may be the true object boundaries, even though these edges could be detected by image gradient.

5.3.2 Combination with Image Saliency

Additionally, our approach could naturally adapt to image saliency and learnt image shape priori similar to Moore et al. (2009). Image saliency detection is known to be an important aspect of computer vision, which is closely connected to human psychology. It has multiple applications in image segmentation, compression, recognition, tracking, and detection as well. Here we adopt the saliency of an image, which is defined as the part of the image that attracts more human attention, to help our segmentation. Heuristically, in salient parts of an image, as in the center of Fig. 21, better segmentation is desired and more detail should be preserved around the region.

Here we update the velocity map $V(\mathbf{x})$ in Eq. (9) by simply integrating the saliency as:

$$V(\mathbf{x}) = V(\mathbf{x})e^{-\lambda \mathbf{S}(\mathbf{x})}, \tag{30}$$

where $S(\mathbf{x})$ is the saliency value of image pixel $\mathbf{x}$ within the range of [0, 1]. It is computed from GBVS (Harel et al. 2006) in our case. By unifying the saliency map with the original velocity map as shown in Fig. 21a, our algorithm could adaptively output more detailed segmentation to the salient part of an image through splitting (Fig. 21b).

6 Applications

6.1 Image Compression

Besides the numerous applications mentioned, superpixels could be firstly considered as a compact representation for image compression. Our algorithm generates better visual effects when compared with Levinshtein et al. (2009b) due to the structure-sensitive distribution of superpixels. By using the same density map, Fig. 22 shows comparative results using 500 superpixels. The color of each superpixel is approximated by three polynomials (one per channel). With a limited number of superpixels, our algorithm produces better details and approaches the quality of the original image.

For a solid comparison, we experimented over the 100 images of BSD300 test set to measure the compression quality of our approach and the TurboPixels. The comparison results are also shown in Fig. 23, which indicate the superiority of our approach.

6.2 Foreground Object Contour Closure and Segmentation

Our algorithm performs adaptively with image structure. This is also very useful for generates better segmentation. In the application of foreground segmentation, we adopted the

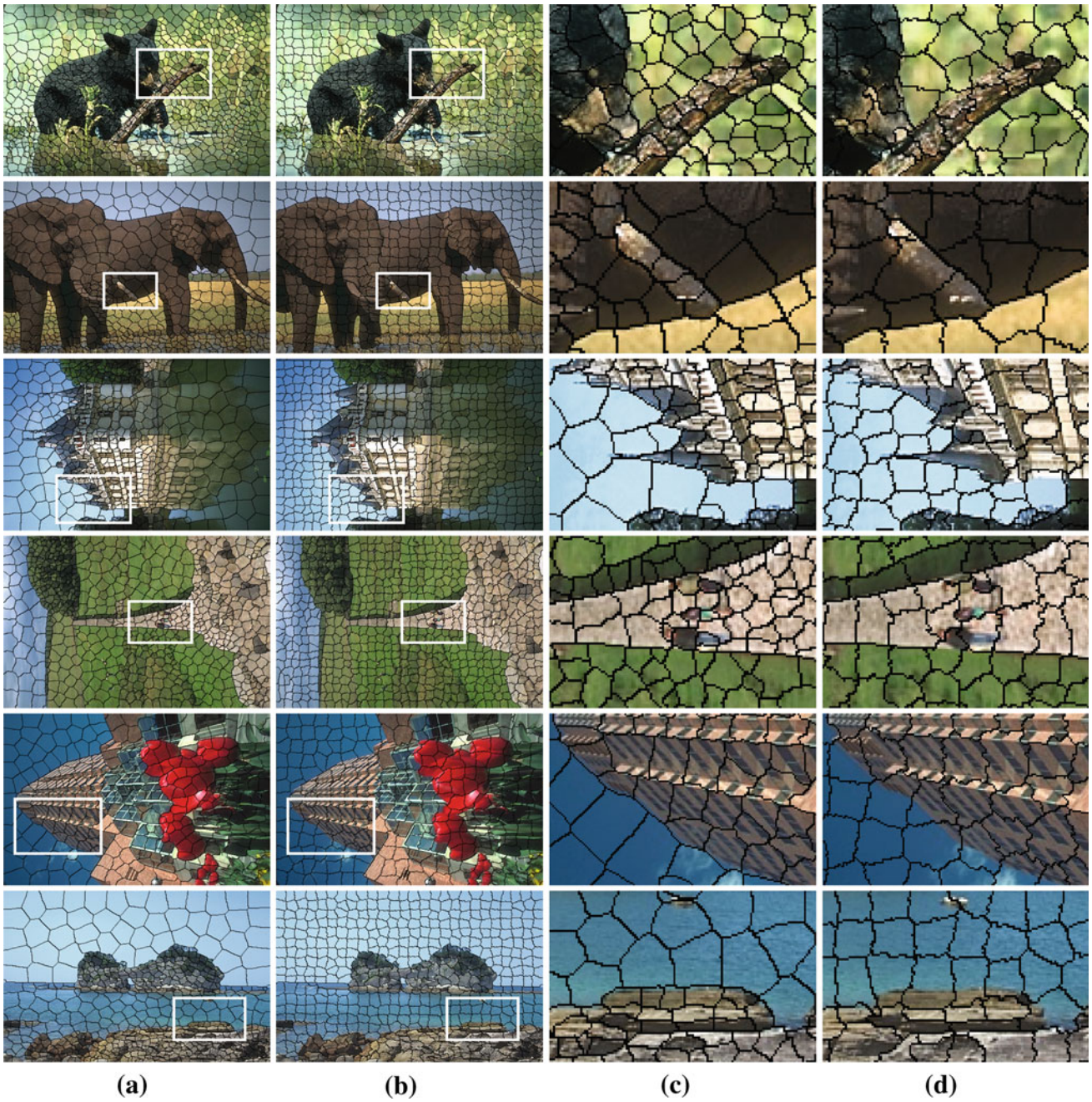


Fig. 18 Comparison between **a** our algorithm and **b** TurboPixels on a variety of images with a zoom in of regions of interest by the white rectangles in column **c** and **d** respectively

Fig. 19 A qualitative result of our method using gradient-based (*middle*) and combined with Pb-based (*right*) affinity functions

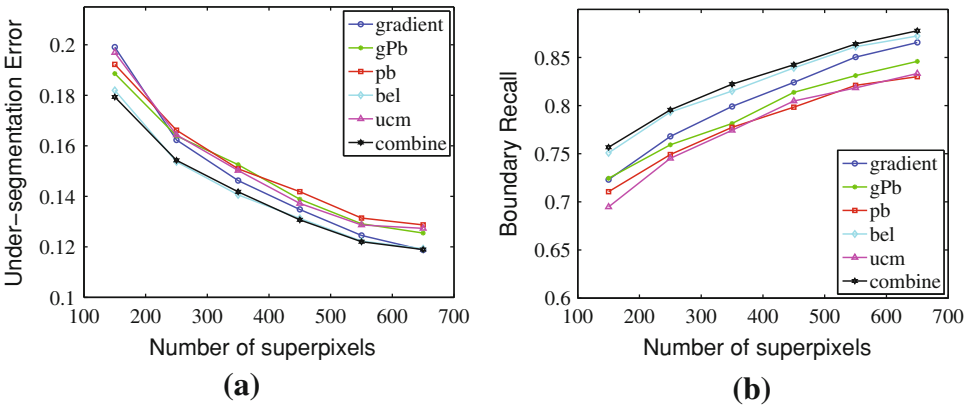


Fig. 20 Quantitative comparisons by combining different learnt edge maps under standard criteria. **a** under-segmentation error and **b** boundary recall

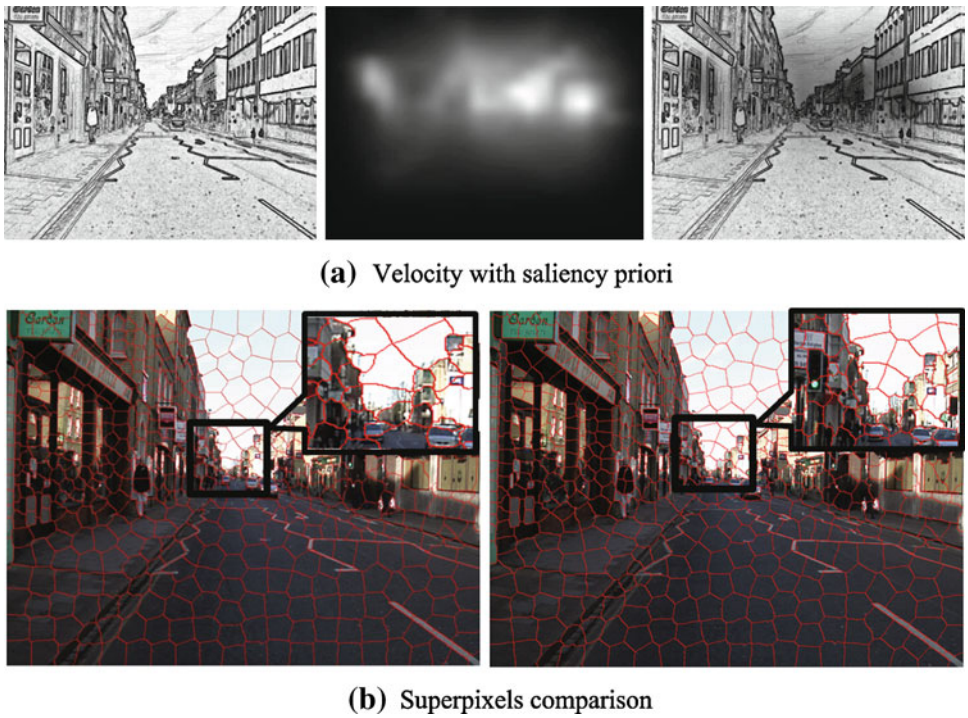


Fig. 21 Gradient-based velocity map (**a**, *left*), the computed saliency map (**a**, *middle*) and the new velocity map (**a**, *right*). The superpixels results generated using the two velocity maps are shown by (**b**, *left*) and (**b**, *right*) respectively

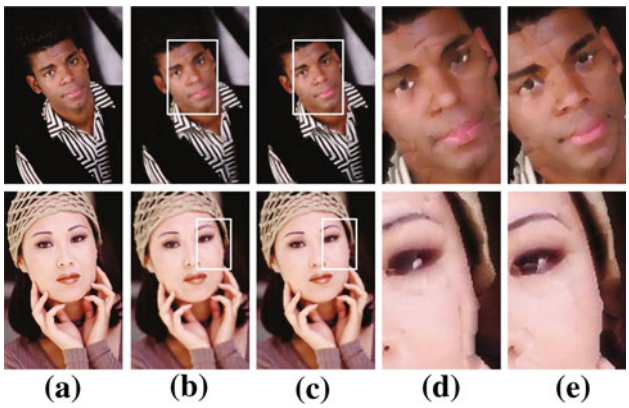


Fig. 22 **a** The original image. Quadratic fit to the color of the image within each superpixel given by **b** TurboPixels and **c** our method. **d** and **e** show a zoom-in on interested region respectively

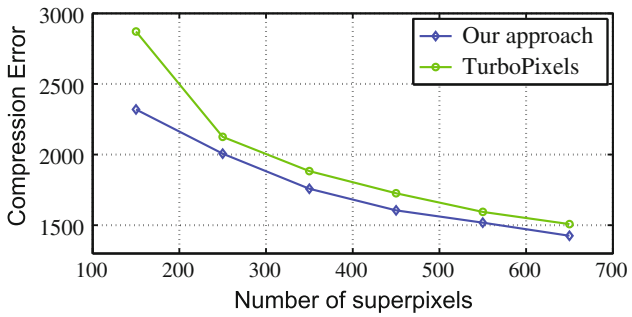


Fig. 23 Quantitative comparison of compression error between our approach and Turbopixels

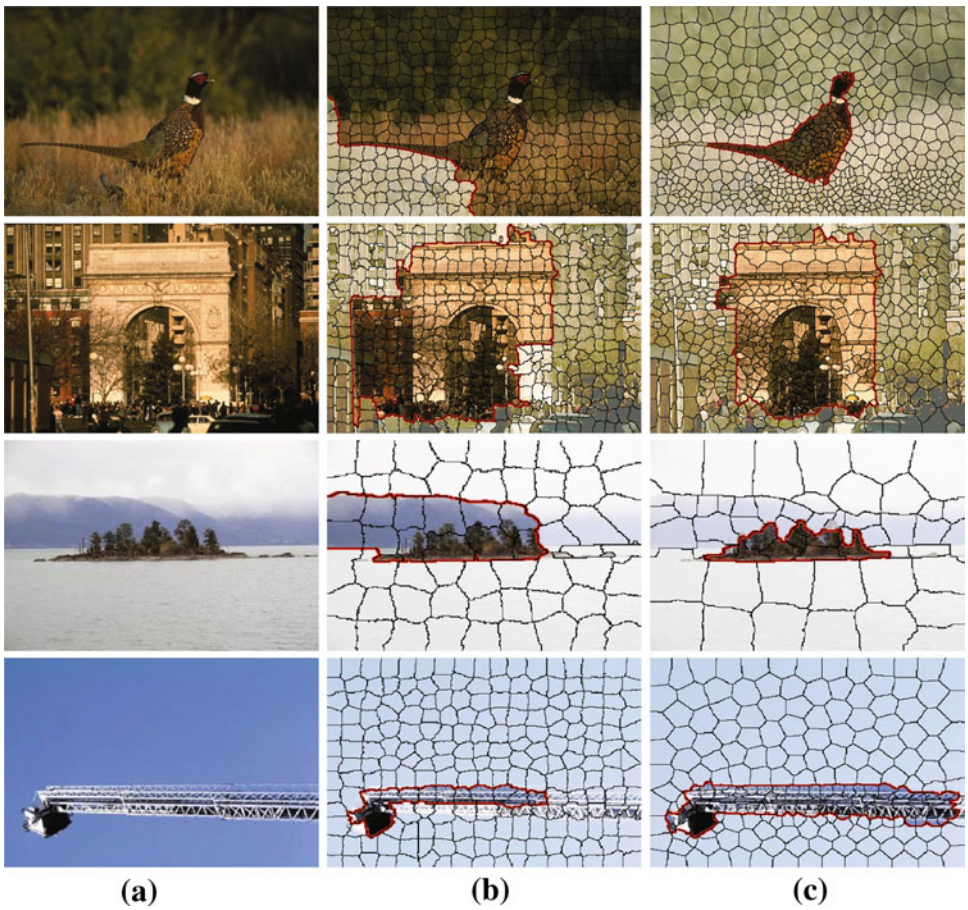


Fig. 24 Contour closure comparison. **a** Example images, **b** and **c** show the extracted object contour based-on TurboPixels (Levinshtein et al. 2009b) and our structure-sensitive superpixels respectively. *The red contour* indicates the detected object boundary (Color figure online)

contour closure extraction method from the work of Levinshtein et al. (2010). Specifically, the goal of contour closure is to find a circle of connected contour fragments that separates an object from its background. Previous art (Levinshtein et al. 2010) transforms such a problem to finding subsets of superpixels. In order to generate a better segmentation, the algorithm requires the superpixel boundaries to better capture the object edges, thus in their work, they apply the learned Pb edge map (Maire et al. 2008) for generating the superpixels. The algorithm optimizes the edge difference between the object and background and the edge homogeneous within the object region.

To show that our approach better captures the object boundary, Fig. 24 shows several comparisons of the best closure solutions generated under different number of superpixels segmentation using our algorithm and TurboPixels based on the density edge map defined in Eq. (2).We show that because the structure information, our algorithm captures much better edge information under smaller superpixels number setting. In addition, we also quantitatively compared the segmentation results between the two superpixel methods over the Weizmann Segmentation Database (WSD) (Alpert et al. 2007) which has the ground truth of single foreground segmentation, As the experimental settings of Levinshtein et al. (2010), the well known F-measure is used to measure the consistency of the detected foreground region and the human segmented region. The results are shown in Fig. 25, which indicates that the our approach promotes the segmentation ability especially with small number of superpixel.

This concludes that our approach achieves visually better segmentation results due to the structure sensitivity. Specifically, our superpixel captures the object boundary more effectively, thus provides better optimized solutions for the contour closure algorithm.

6.3 Video Segmentation

Our approach could be easily extended to video segmentation and is more suitable for superpixels segmentation than previous superpixel methods with quasi-lattice formation such as Levinshtein et al. (2009b). Thanks to the spatial consistency between different frames, instead of re-calculation at each frame, the optimized position of centers could be easily transferred by methods like SIFT flow (Liu et al. 2009) or optical flow (Lucas and Kanade 1981). SIFT flow gives better correspondence, but it takes more time to perform a dense matching. Here, we conducted the superpixel flow based on the popular Lucas–Kanade (LK) algorithm to find stable

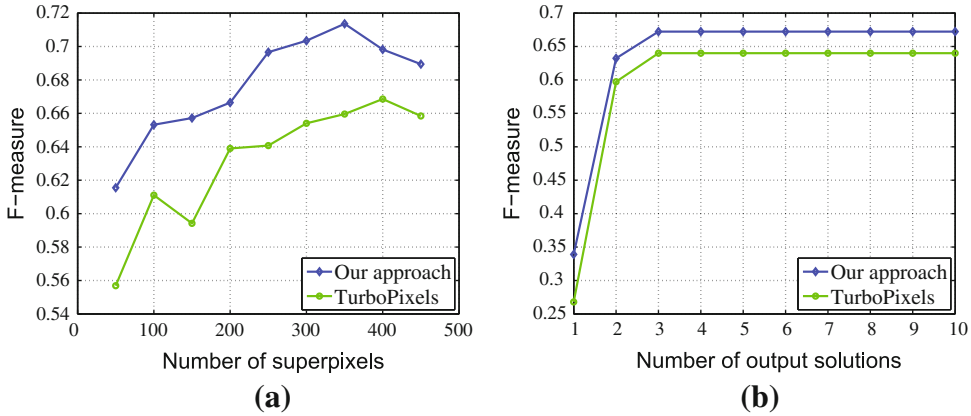
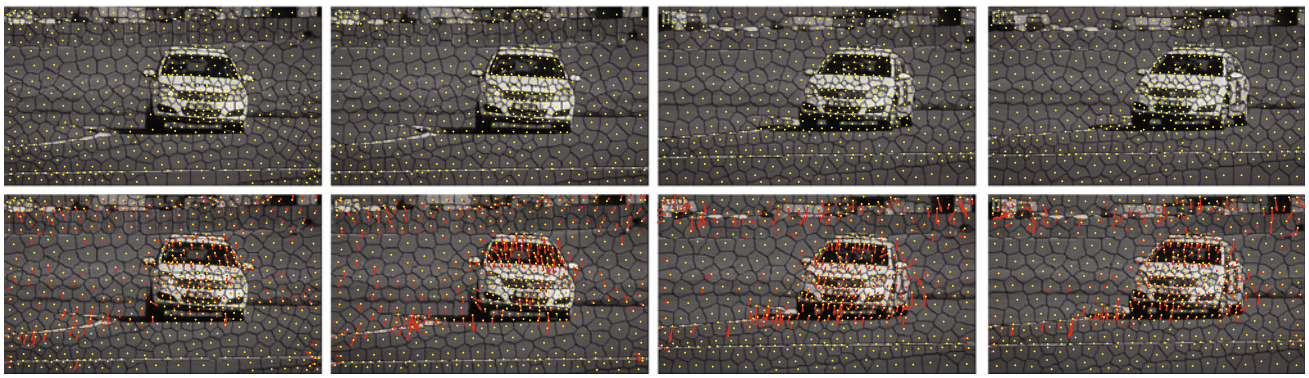


Fig. 25 Qualitative segmentation comparison between our approach and TurboPixel. **a** F-measure versus the superpixel number and **b** F-measure versus the solution number



(a) Frame #4 **(b)** Frame #5 **(c)** Frame #14 **(d)** Frame #15

Fig. 26 Results of a video's over-segmentation in multiple frames. The *first row*: superpixel segmentation results from our algorithm. The *second row*: optical flow at the centers' positions. The *red arrows* indicate the stable transfers from current center position to that of next frame (Color figure online)

correspondent centers between adjacent frames. In the video segmentation scenario, initial seeds have already been given a structure-sensitive placement which was optimized using the former frame. This informs us that the algorithm could converge much faster for the later frames.

Figure 26 gives an example of our video segmentation. The first row shows the superpixel segmentation results and the second row shows the center transfer between adjacent frames, in which the red arrow indicates the seeds' movement. The transferred centers are always the ones at structure dense regions such as the place on the car. These regions are more important for avoiding under-segmentation, and thus the transferring is satisfactory in most cases. Notice that besides the first frame, the latter frames cost only $2s-4s$ for images around 300×500.

7 Conclusion

We proposed a structure-sensitive over-segmentation algorithm for computing superpixels for images. It greatly limits under-segmentation by considering the homogeneity of image appearance, density of image contents, compactness of shape, and regularity of layout. The over-segmentation can be formulated as a soft clustering problem by exploiting geodesic distance, and a local optimal solution could be obtained via geometric flows and an efficient iterative optimization strategy through inducing center splitting. Experimental results on the Berkeley segmentation dataset demonstrate that our algorithm outperforms many state-of-the-art approaches, whilst the running time of the algorithm is fully adoptable for many practical usage.

Additionally, we discussed another optimization strategy inducing merging, while it gives similar performance with purely splitting strategy. At last, we provided several potential applications, e.g. supervised segmentation, image compression and video segmentation, and show the superiority of our structure-sensitive setting superpixel method compared with grid layout setting methods, such as TurboPixels.

Acknowledgments This work is supported by National Nature Science Foundation of China (NSFC Grant) 61005037 and 90920304,

National Basic Research Program of China (973 Program) 2011CB302202, and Beijing Natural Science Foundation (BJNSF Grant) 4113071.

References

Alpert, S., Galun, M., Basri, R., & Brandt, A. (2007). Image segmentation by probabilistic bottom-up aggregation and cue integration. In *CVPR*.

Arbelaez, P., Maire, M., Fowlkes, C. C., & Malik, J. (2009). From contours to regions: An empirical evaluation. In *CVPR* (pp. 2294–2301).

Bai, X., & Sapiro, G. (2007). A geodesic framework for fast interactive image and video segmentation and matting. In *ICCV* (pp. 1–8).

Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(5), 603–619.

Criminisi, A., Sharp, T., & Blake, A. (2008). Geos: Geodesic image segmentation. In *ECCV* (pp. 99–112).

Dollár, P., Tu, Z., & Belongie, S. (2006). Supervised learning of edges and object boundaries. In *CVPR* (Vol. 2, pp. 1964–1971).

Du, Q., Emelianenko, M., & Ju, L. (2006). Convergence of the lloyd algorithm for computing centroidal voronoi tessellations. *SIJNA: SIAM Journal on Numerical Analysis, 44*, 102–119.

Feil, B., & Abonyi, J. (2007). *Geodesic distance based fuzzy clustering*. Lecture notes in computer science, soft computing in industrial applications (pp. 50–59).

Felzenszwalb, P. F., & Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, *59*(2), 167–181.

Fulkerson, B., Vedaldi, A., & Soatto, S. (2009). Class segmentation and object localization with superpixel neighborhoods. In *ICCV* (pp. 670–677).

Gulshan, V., Rother, C., Criminisi, A., Blake, A., & Zisserman, A. (2010). Geodesic star convexity for interactive image segmentation. In *CVPR* (pp. 3129–3136).

Harel, J., Koch, C., & Perona, P. (2006). Graph-based visual saliency. In B. Schölkopf, J. C. Platt, & T. Hoffman (Eds.), *NIPS* (pp. 545–552). Cambridge, MA: MIT Press.

He, X., Zemel, R. S., & Ray, D. (2006). Learning and incorporating top-down cues in image segmentation. In *ECCV* (Vol. 1, pp. 338–351).

Hoiem, D., Efros, A. A., & Hebert, M. (2005). Geometric context from a single image. In *ICCV* (pp. 654–661).

Hyvärinen, A. (1999). The fixed-point algorithm and maximum likelihood estimation for independent component analysis. *Neural Processing Letters*, *10*(1), 1–5.

Jolliffe, I. T. (1986). Principal component analysis. In *Principal component analysis*. New York: Springer.

Kaufhold, J. P., Collins, R., Hoogs, A., & Rondot, P. (2006). Recognition and segmentation of scene content using region-based classification. In *ICPR* (Vol. 1, pp. 755–760).

Kim, J., Shim, K. H., & Choi, S. (2007). Soft geodesic kernel k-means. In *ICASSP* (pp. 429–432).

Levinshtein, A., Dickinson, S. J., & Sminchisescu, C. (2009a). Multiscale symmetric part detection and grouping. In *ICCV* (pp. 2162–2169).

Levinshtein, A., Sminchisescu, C., & Dickinson, S. J. (2010). Optimal contour closure by superpixel grouping. In *ECCV* (Vol. 2, pp. 429–493).

Levinshtein, A., Stere, A., Kutulakos, K. N., Fleet, D. J., Dickinson, S. J., & Siddiqi, K. (2009b). Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(12), 2290–2297.

Li, Y., & Chung, S. M. (2007). Parallel bisecting k-means with prediction clustering algorithm. *The Journal of Supercomputing, 39*, 19–37.

Liu, C., Yuen, J., & Torralba, A. (2009). Nonparametric scene parsing: Label transfer via dense scene alignment. In *CVPR* (pp. 1972–1979).

Lloyd, S. P. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, *28*, 128–137.

Lucas, B., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of the DARPA image understanding workshop* (pp. 121–130).

Maire, M., Arbelaez, P., Fowlkes, C., & Malik, J. (2008). Using contours to detect and localize junctions in natural images. In *CVPR*.

Malisiewicz, T., & Efros, A. A. (2007). Improving spatial support for objects via multiple segmentations. In *BMVC*.

Martin, D. R., Fowlkes, C., & Malik, J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*(5), 530–549.

Martin, D. R., Fowlkes, C., Tal, D., & Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV* (pp. 416–425).

Meyer, F., & Maragos, P. (1999). Multiscale morphological segmentations based on watershed, flooding, and eikonal PDE. In *Scale space* (pp. 351–362).

Micusík, B., & Kosecká, J. (2010). Multi-view superpixel stereo in urban environments. *International Journal of Computer Vision*, *89*(1), 106–119.

Moore, A. P., Prince, S. J. D., & Warrell, J. (2010). "lattice cut"—Constructing superpixels using layer constraints. In *CVPR* (pp. 2117–2124).

Moore, A. P., Prince, S., Warrell, J., Mohammed, U., & Jones, G. (2008). Superpixel lattices. In *CVPR*.

Moore, A. P., Prince, S. J. D., Warrell, J., Mohammed, U., & Jones G. (2009). Scene shape priors for superpixel segmentation. In *ICCV* (pp. 771–778).

Mori, G. (2005). Guiding model search using segmentation. In *ICCV* (pp. 1417–1423).

Muhr, M., & Granitzer, M. (2009). Automatic cluster number selection using a split and merge K-means approach. In A. M. Tjoa & R. Wagner (Eds)., *DEXA workshops* (pp. 363–367). IEEE Computer Society.

Nwogu, I., & Corso, J. J. (2008). $(bp)^2$: Beyond pairwise belief propagation labeling by approximating kikuchi free energies. In *CVPR*.

Peyré, G., Péchaud, M., Keriven, R., & Cohen, L. D. (2010). Geodesic methods in computer vision and graphics. *Foundations and Trends in Computer Graphics and Vision*, *5*(3–4), 197–397.

Radhakrishna, A., Appu, S., Kevin, S., Aurelien, L., Pascal, F., & Susstrunk, S. (2010). Slic superpixels. Technical Report 149300 EPFL (June), p. 15.

Rasmussen, C. (2007). Superpixel analysis for object detection and tracking with application to UAV imagery. In *Advances in visual computing* (Vol. I, pp. 46–55).

Russell, B. C., Freeman, W. T., Efros, A. A., Sivic, J., & Zisserman, A. (2006). Using multiple segmentations to discover objects and their extent in image collections. In *CVPR* (Vol. 2, pp. 1605–1614).

Savaresi, S. M., & Boley, D. (2004). A comparative analysis on the bisecting K-means and the PDDP clustering algorithms. *Intelligent Data Analysis*, *8*(4), 345–362.

Sethian, J. (1996a). A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, *93*, 1591–1694.

Sethian, J. A. (1996b). A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, *93*(4), pp. 1591–1595.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 888–905.

Shotton, J., Winn, J. M., Rother, C., & Criminisi, A. (2006). TextonBoost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *ECCV* (Vol. 1, pp. 1–15).

Tai, X. C., Hodneland, E., Weickert, J., Bukoreshtliev, N. V., Lundervold, A., & Gerdes, H. H. (2007). Level set methods for watershed image segmentation. In *Scale-space* (pp. 178–190).

Veksler, O., Boykov, Y., & Mehrani, P. (2010). Superpixels and supervoxels in an energy optimization framework. In *ECCV* (Vol. 5, pp. 211–224).

Vincent, L., & Soille, P. (1991). Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *13*(6), 583–598.

Wang, J., Jia, Y., Hua, X. S., Zhang, C., & Quan, L. (2008). Normalized tree partitioning for image segmentation. In *CVPR*.

Wang, S., Lu, H., Yang, F., & Yang, M. H. (2011). Superpixel tracking. In *ICCV* (pp. 1323–1330).

Xiao, J., & Quan, L. (2009). Multiple view semantic segmentation for street view images. In *ICCV* (pp. 686–693).

Yatziv, L., Bartesaghi, A., & Sapiro, G. (2006). O(n) implementation of the fast marching algorithm. *Journal of Computational Physics, 212*(2), 393–393.