Structure-sensitive Superpixels via Geodesic Distance

Gang Zeng Peng Wang Jingdong Wang[†] Rui Gan Hongbin Zha
Key Laboratory on Machine Perception, Peking University [†]Microsoft Research Asia
{gang.zeng,jerrywang,raygan}@pku.edu.cn jingdw@microsoft.com zha@cis.pku.edu.cn

Abstract

Over-segments (i.e. superpixels) have been commonly used as supporting regions for feature vectors and primitives to reduce computational complexity in various image analysis tasks. In this paper, we describe a structure-sensitive over-segmentation technique by exploiting Lloyd's algorithm with a geodesic distance. It generates smaller superpixels to achieve lower 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 the geodesic distances amongst pixels, and in the segmentation procedure, the density of over-segments is automatically adjusted according to an energy functional that embeds color homogeneity, structure density and compactness constraints. Comparative experiments with the Berkeley database show that the proposed algorithm outperforms prior arts while offering a comparable computational efficiency with fast methods, such as TurboPixels.

1. Introduction

Image over-segmentation has been widely applied in various computer vision pipelines, such as segmentation [37, 36, 11], recognition [13], tracking [28], localization [7] and modeling [10, 26]. In these applications, over-segments (also known as *superpixels* in [29]) represent small regions with homogeneous appearance and conform to local image structure, and thus they 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.

The challenge of superpixels is that on one hand they are required to reduce image complexity by locally grouping pixels respecting 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 various optimization techniques have been adopted to make trade-offs, for example, the mean shift algorithm [2], the normalized cuts [32], the local viaration [6], the geometric flows [16] and the watershed [35, 21, 34],

Fig. 1 shows the segmentation results obtained using *Graph-based method* [6], *Lattice* [22], *N-Cuts* [25, 15], *TurboPixels* [16] and our method. Graph-based method [6] lacks compactness constraints and may generate under-segmentation with regions of irregular shapes and sizes. The other methods employ compactness constraints and markedly restrict under-segmentation. The advantage of utilizing compactness has also been demonstrated in [16].

Lattice [22] generates superpixels by detecting vertical or horizon strips, and it naturally maintains a grid structure of regions. Later the authors combined scene shape prior to achieve an adaptive lattice [24]. Further investigation of lattice superpixel [23] is derived from global optimization. The superpixel generation is initialized with a grid, and the graph cut algorithm is adopted to iteratively optimize the vertical and horizontal seams.

N-cuts-based superpixels [25, 15] are variations of the normalized cuts algorithm by [32], 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 greatly with the number of pixels and image size.

Recently, Levinshtein *et al.* proposed a geometric-flow-based algorithm (i.e. TurboPixels) for superpixel segmentation [16]. 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.

A further observation in Fig. 1 shows that the density of image contents often differs in different parts of the image, given that there are a large diversity of scene layout and that imaging process unavoidably introduces prospective distortion. The over-segments of *Lattice*, *N-Cuts* and *TurboPixels* in Fig. 1 (b),(c)&(d) are too large to represent image appearance and lead to under-segmentation in regions near intensity boundaries, while the segments are rather small in

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

Figure 1. Over-segmentations obtained with five algorithms: (a) Local variation [6], (b) Superpixel Lattice [22], (c) N-Cuts [32], (d) TurboPixels [16] and (e) the method presented in this paper. The second row zooms in on the regions of interest defined by the white rectangles.

homogeneous regions resulting in unnecessary overhead in high-level applications. That says, superpixels with quasi-uniform distribution is in a dilemma, since the number of superpixels is hard to choose. This is also proven by several methods that use multiple over-segments as a starting point for further scene segmentation [18, 30].

A better image representation would be achieved when the density of superpixels is assigned adaptively with respect to the co-occurrence of image contents or "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.

The density function is also motivated by the following analysis on similarity measure among pixels. A commonly used similarity measure is the Euclidean distance in a high-dimensional space that is based on three components of color and two image coordinates. The main disadvantage of such measure 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 [27]. The aforementioned density function forms the basis of the geodesic distance, namely that 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 used for interactive segmentation and matting in [1, 3, 9]. 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.

1.1. Overview

The proposed algorithm resembles Lloyd's algorithm [17] but with the geodesic distance defined in Eqn. (1). It's based on an energy functional (in Sec.2) that

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

embeds structure and compactness constraints. Fig. 3 shows an overview of our system. The density of superpixels is sensitive to image structure and changes adaptively during the algorithm.

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

The location of the centers and shape of each superpixel keep changing in turn as the algorithm runs, and 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 [31] is employed to calculate the geodesic distance and 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 Sec. 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 description of this part is in Sec. 3.2.

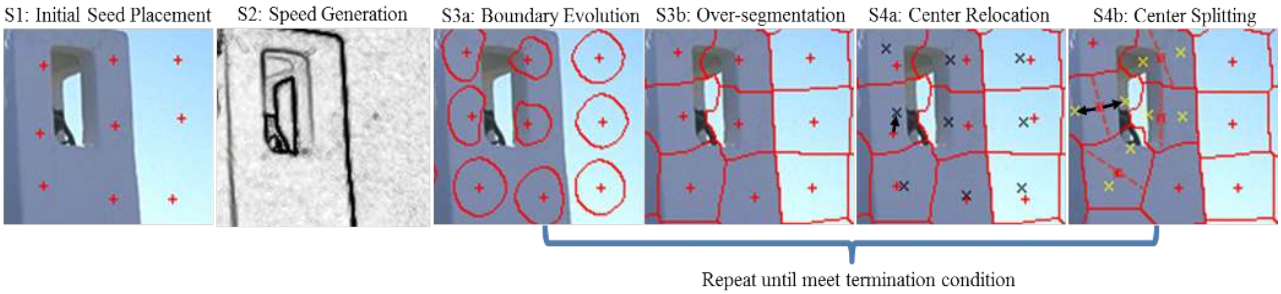


Figure 3. The procedure of our algorithm: Initial seeds (S1) grow with the speed (S2) to form an over-segmentation (S3), and the centers are relocated or split (S4) by certain criteria related by shape or size. In S4a the red"+" is the original places of center points and black "x" is the recalculated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

2. Problem Formulation via Geodesic Distance

Given an input image $I(\mathbf{x})$, the goal is to over-segment $I(\mathbf{x})$ into dense small regions representing superpixels at different locations. We assign a unique label to each superpixel and use $L(\mathbf{x})$ to denote the label of the current pixel $\mathbf{x}$. Thus, all pixels belonging to the l-th superpixel S_l can be detected by using $S_l = \{\mathbf{x}|L(\mathbf{x}) = l\}$.

As shown in Fig. 2, image contents in-between could provide a crucial evidence for measuring the similarity between two pixels. We exploit a 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:

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

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 inspired by [16], it takes the form as follows:

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

where ν is a scaling parameter. $E(\mathbf{x})$ 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 guarantees that very weak intensity boundaries do not effect too much in the density computation.

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. Moreover, 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 makes the superpixels compact so as to avoid large under-segmentation when the image regions contain little edge information.

With the geodesic distance $D_g(\mathbf{x}_i, \mathbf{x}_j)$ defined in Eqn. (1), the superpixels are required to be compact and conform to image boundaries, which leads to the following criterion:

$$\text{Compactness} \quad : \quad L(\mathbf{x}) = \arg\min_l D_g(\mathbf{c}_l, \mathbf{x}), \quad (3)$$

where $\mathbf{c}_l$ denotes the center of the l-th superpixel.

With the compactness constraint, the distribution of the centers $\{\mathbf{c}_l\}$ uniquely determines the density and shapes of superpixels, and hence the over-segmentation problem could be formulated as an optimal quantization problem [8] for computing centroidal Voronoi tessellations on the image.

2.1. Energy Minimization

Besides the compactness constraints in Eqn. (3), we further adapt the magnitudes of superpixels for better representing local structures on the image:

$$\text{Structure} \quad : \quad A_l \approx A_{l'}, \forall l \neq l', \text{ with } A_l = \int_{S_l} D(\mathbf{x}) d\mathbf{x}, \quad (4)$$

where A_l denotes the area of superpixel S_l.

From Eqn. (2), the density function $D(\mathbf{x})$ is high in the regions with much intensity variation and thus leads to smaller area S_l on the image. This motivates us to define an energy term:

$$E_{structure} = \sum_l (A_l - \overline{A})^2, \quad (5)$$

where $\overline{A}$ is the average of $\{A_l\}$, which can be easily calculated by $\frac{\sum_l A_l}{N} = \frac{\int_{\mathbf{x}} D(\mathbf{x}) d\mathbf{x}}{N}$ in which N is the total number of superpixels specified by users.

Moreover, inspired by the robustness of recent clustering methods using geodesic distance [5, 14], we penalize the label inconsistency between a pixel and its closest center on the image if their geodesic distance is small:

$$E_{image} = \sum_l \int_{S_l} W_{\mathbf{x}} D_g(\mathbf{c}_l, \mathbf{x})^2 d\mathbf{x}, \quad (6)$$

where $W_{\mathbf{x}}$ is a weight function defined as $e^{-D_g(\mathbf{c}_{L(\mathbf{x})},\mathbf{x})/\varphi}$. $W_{\mathbf{x}}$ measures the probability that pixel $\mathbf{x}$ and its closest center $\mathbf{c}_{L(\mathbf{x})}$ have the same label based on their geodesic distance $D_g(\mathbf{c}_{L(\mathbf{x})},\mathbf{x})$. $\varphi = 0.5$ is a scaling parameter.

$$E_{total} = E_{image} + \alpha E_{structure}, \quad (7)$$

where α is a balancing factor. Thus, the superpixels $\{S_l\}$ are generated with the compactness constraint from a set of centers $\{\mathbf{c}_l\}$ and they optimize the total energy functional that embeds image homogeneity and structure density.

3. Structure-sensitive Superpixels

Due to its highly non-convex properties, we choose to use an iterative scheme to minimize the energy E_{total}. The optimization process is similar to Lloyd's algorithm [17] and converges to some local minimum. The convergency and robustness of the algorithm has been elaborated by Du *et al*. [4]. In our algorithm the centers $\{\mathbf{c}_l\}$ and pixel labels $L(\mathbf{x})$ are alternatively updated in turn during the iterative procedure. Both of the two routines are designed according to the energy functional in Eqn. (7), and their descriptions are in Sec. 3.1& 3.2 respectively.

3.1. Over-segmentation with Known Centers

Given a set of centers $\{\mathbf{c}_l\}$, the goal in this step is to compute local segments $L(\mathbf{x})$ by the compactness constraint in Eqn. (3) and energy functional in Eqn. (7).

In order to generate geodesic distances, we here employ the fast marching method [31] for better computational efficiency since this over-segmentation step may get involved several times during the outer iterations. Moreover, in our configuration, the frontend of the evolving contour can only move in the direction of the outward normal (*i.e.* the contour expands rather than shrink), which fits well with the restricted forms of the underlying velocity functions of the fast marching.

The velocity function for calculating the geodesic distance in Eqn. (1) is defined as follows:

$$V(\mathbf{x}) = D(\mathbf{x})^{-1}, \quad (8)$$

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

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

$$\begin{aligned} & V(\mathbf{x})\|\nabla D_g(\mathbf{c}_l,\mathbf{x})\| = 1, \\ \text{with} \quad & D_g(\mathbf{c}_l,\mathbf{c}_l) = 0, \quad \forall l. \end{aligned} \quad (9)$$

With $D_g(\mathbf{c}_l,\mathbf{x})$ at hand, $L(\mathbf{x})$ may be determined directly through the compactness constraint in Eqn. (3). It can be easily proven that E_{image} in Eqn. (6) is minimized. However, the minimization of $E_{structure}$ in Eqn (5) is not achieved.

Mathematically, it can be proven that the velocity V_s to optimize $E_{structure}$ is $\frac{V(x)}{(A_l(d)-\overline{A})\frac{\partial A_l(d)}{\partial d}}$, where d is short for $D_g(\mathbf{c}_l,\mathbf{x})$ and $A_l(d)$ represents the current area encircled by the evolving contour. For efficiency, we simplify V_s but keep it sensitive at final stages when $A_l(d) - \overline{A}$ approaches 0, resulting in $V_s(x) = \frac{V(x)}{A_l(d)-\overline{A}} \cdot \text{const}$. The velocity $V_l(x)$ for E_{total} satisfies $\frac{1}{V_l(x)} = \frac{1}{V(x)} + \frac{\alpha}{V_s(x)}$, *i.e.* $V_l(x) = \frac{V(x)}{1+\alpha*(A_l(d)-\overline{A})}$. To tolerate to errors in $A_l(d)$'s computation, we further use a Gaussian function to substitute the denominator leading to the equation:

$$\begin{aligned} V_l(\mathbf{x},d) &= V(\mathbf{x}) \cdot G'_{\sigma'}(\max\{0, A_l(d) - \overline{A}\}), \quad \text{with} \\ A_l(d) &= \int_{\{\mathbf{x}|\mathbf{x}\in S_l, D_g(\mathbf{c}_l,\mathbf{x})<d\}} D(\mathbf{x})d\mathbf{x}, \end{aligned} \quad (10)$$

where $G'_{\sigma'}(\cdot)$ denotes an un-normalized Gaussian function with its standard deviation $\sigma' = \overline{A}/\alpha$. Thus, in order to minimize the total energy functional in Eqn. (7), we adopt the following form for the Eikonal equation:

$$\begin{aligned} & V_l(\mathbf{x}, D_g(\mathbf{c}_l,\mathbf{x}))\|\nabla D_g(\mathbf{c}_l,\mathbf{x})\| = 1, \\ \text{with} \quad & D_g(\mathbf{c}_l,\mathbf{c}_l) = 0, \quad \forall l. \end{aligned} \quad (11)$$

3.2. Center Refinement with Known Regions

Center Relocation Given a set of superpixels $L(\mathbf{x})$, in this step the centers $\{\mathbf{c}_l\}$ relocate according to E_{image} in Eqn. (6). The location of each center should be updated by:

$$\mathbf{c'}_l = \arg\min_{\mathbf{x}'\in S_l} \int_{S_l} W_{\mathbf{x}} D_g(\mathbf{x}',\mathbf{x})^2 d\mathbf{x}. \quad (12)$$

An exhaustive search as in [5] 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:

$$\begin{aligned} \frac{\partial E_{image}}{\partial \mathbf{c'}_l} &= 2\int_{S_l} W_{\mathbf{x}} D_g(\mathbf{c'}_l,\mathbf{x}) \nabla D_g(\mathbf{c'}_l,\mathbf{x}) d\mathbf{x} \quad (13) \\ &\approx 2\int_{S_l} W_{\mathbf{x}} D_g(\mathbf{c}_l,\mathbf{x}) \frac{(\mathbf{x}-\mathbf{c'}_l)}{\|\mathbf{x}-\mathbf{c}_l\|} d\mathbf{x} = \mathbf{0}, \end{aligned}$$

where we use $\mathbf{c}_l$ to substitute $\mathbf{c'}_l$ for computational convenience. In the last line of the above inference equations, we use $\frac{(\mathbf{x}-\mathbf{c'}_l)}{\|\mathbf{x}-\mathbf{c}_l\|}$ as the approximation of $\nabla D_g(\mathbf{c'}_l,\mathbf{x})$, since the resulting equation has a similar form as the computation of the center of mass of the segment S_l with its center being $\mathbf{c'}_l$ and its mass equal to $\int_{S_l\setminus\{\mathbf{c}_l\}} W_{\mathbf{x}} \frac{D_g(\mathbf{c}_l,\mathbf{x})}{\|\mathbf{x}-\mathbf{c}_l\|} d\mathbf{x}$. Thus, the new center relocates to:

$$\mathbf{c}'_l = \frac{\int_{S_l\setminus\{\mathbf{c}_l\}} W_{\mathbf{x}} \frac{D_g(\mathbf{c}_l,\mathbf{x})}{\|\mathbf{x}-\mathbf{c}_l\|} \mathbf{x} d\mathbf{x}}{\int_{S_l\setminus\{\mathbf{c}_l\}} W_{\mathbf{x}} \frac{D_g(\mathbf{c}_l,\mathbf{x})}{\|\mathbf{x}-\mathbf{c}_l\|} d\mathbf{x}}. \quad (14)$$

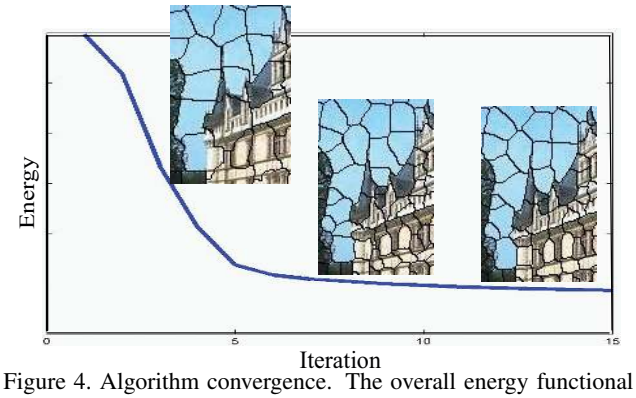


Figure 4. Algorithm convergence. The overall energy functional decreases and the superpixels perform better with every iteration.

The experiments of overall energy in Fig. 4 also validate the estimation of new center locations as the energy functional keep decreasing during the iteration.

Center Splitting As mentioned in Sec. 1, one of the main goals is to generate superpixels that are consistent with image structure which leads that they are almost the same size based on the density term (See Eqn. (4)).

Thus during the energy minimization process, given a superpixel S_l whose area A_l is large while its center $\mathbf{c}_l$ shifts little from last iteration, the algorithm splits the center $\mathbf{c}_l$ into two centers since the latter generated segments by the new ones would produce a lower value of the energy functional in Eqn. (7).

To this end, we define criteria to distinguish superpixels' centers to be split and those to be relocated:

$$\begin{aligned} C_{shape}(\mathbf{c}_l) &= \max\{\frac{EigV_1}{EigV_2}, \lambda Std_l\} > T_c; \\ C_{size}(\mathbf{c}_l) &= \frac{A_l}{\overline{A}} > T_s, \end{aligned} \tag{15}$$

The splitting of centers is performed if either C_{shape} or C_{size} is satisfied. T_c and T_s are thresholds, while Std_l denotes the standard deviation of pixel colors within each superpixel under normalized CIElab color space as in [33]. $EigV_1$ and $EigV_2$ are the first and second eigenvalue obtained by the PCA [12] of the following 2×2 matrix:

$$\int_{S_l \setminus \{\mathbf{c}_l\}} \frac{D_g(\mathbf{c}_l, \mathbf{x})^2}{\|\mathbf{x} - \mathbf{c}_l\|^2} (\mathbf{x} - \mathbf{c}_l)(\mathbf{x} - \mathbf{c}_l)^T d\mathbf{x}. \tag{16}$$

If either splitting criterion in Eqn. (15) is satisfied, 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:

$$\begin{aligned} \mathbf{c}'_{l,1} &= \frac{\int_{\{\mathbf{x}|\mathbf{x}\in S_l, (\mathbf{x}-\mathbf{c}_l)\cdot\mathbf{n}>0\}} \frac{D_g(\mathbf{c}_l,\mathbf{x})}{\|\mathbf{x}-\mathbf{c}_l\|} \mathbf{x} d\mathbf{x}}{\int_{\{\mathbf{x}|\mathbf{x}\in S_l, (\mathbf{x}-\mathbf{c}_l)\cdot\mathbf{n}>0\}} \frac{D_g(\mathbf{c}_l,\mathbf{x})}{\|\mathbf{x}-\mathbf{c}_l\|} d\mathbf{x}}, \\ \mathbf{c}'_{l,2} &= \frac{\int_{\{\mathbf{x}|\mathbf{x}\in S_l, (\mathbf{x}-\mathbf{c}_l)\cdot\mathbf{n}<0\}} \frac{D_g(\mathbf{c}_l,\mathbf{x})}{\|\mathbf{x}-\mathbf{c}_l\|} \mathbf{x} d\mathbf{x}}{\int_{\{\mathbf{x}|\mathbf{x}\in S_l, (\mathbf{x}-\mathbf{c}_l)\cdot\mathbf{n}<0\}} \frac{D_g(\mathbf{c}_l,\mathbf{x})}{\|\mathbf{x}-\mathbf{c}_l\|} d\mathbf{x}}, \end{aligned} \tag{17}$$

where $\mathbf{n}$ denotes the corresponding eigenvector of $EigV_1$.

Moreover, in the rare cases that no splitting criteria is met while demanded seeds number is not reached, we select the largest few superpixels (10 in our implementation) to perform the splitting.

3.3. Initialization and Termination

Initial Seeds Placement Similar to TurboPixels [16], we place K initial seeds 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. We also perturb the seeds by moving away from the pixels with high gradient magnitude to avoid strong intensity boundaries and bad initialization for latter iteration.

Different from TurboPixels algorithm, we set K to be a portion of the total amount 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.

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}.

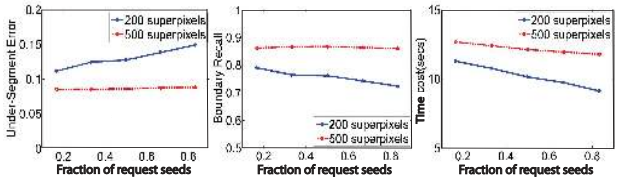
In the final stage, very small superpixels are detected and removed, which is resulting in a small amount of unassigned pixels. The final segmentation is generated by the over-segmentation (in Sec. 3.1) with the remaining centers.

3.4. Algorithm Complexity and Convergence

As the algorithm iteratively performs two routines in turn, 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 Sec. 3.1 and center refinement in Sec. 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 $O(M)$ [38]. 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 $O(MN_I)$.

Our experiments show that the algorithm mostly terminates within 20-30 iterations. The number of centers increases quickly at the first several iterations when oversegments have larger sizes, which makes the energy functional decreases rapidly. Fig. 4 shows the energy functional decreasing 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 $O(M)$.



(a) (b) (c)

Figure 5. Performance related to different portion of initial seeds: (a) Under-segmentation Error (b) Boundary Recall (c) Time Cost.

4. Experimental Results

4.1. Implementation Details

In our experiments, the standard deviation σ and parameter γ in Eqn. (2) is set to $\sigma = \frac{\sqrt{M/N}}{2}$ and $\gamma = 0.12$, where M is the total number of pixels in the image and N is the user-specified number of superpixels. We set $\alpha = 1$ in Eqn. (7) and $T_s = 2, \lambda = 36, T_c = 4$ in Eqn. (15).

For the setting of initial seeds number, Fig. 5 shows the quantitative test (using criteria in Sec. 4.2) related to the ratio between the number of initial seeds and that of user's requirement. As shown, a smaller fraction performs little better but takes more time to converge. In experiments, we set the initial seeds to $\frac{N}{4}$, which generally ensures the minimum area A_l in the first iteration larger than $\overline{A}$.

We use Fast Marching Toolbox[1] to compute geometric flows. In the latter section, we evaluate the performance of the proposed algorithm by comparing its accuracy and running time with TurboPixels [16] and N-Cuts [32]. We use Multiscale Normalized Cuts Segmentation Toolbox[2] for N-Cuts and TP[3] for TurboPixels. All experiments are performed on a quad-core 3.2GHz computer. The evaluation is based on the BSD300 data set [20], 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 (20-30 images) of the test set as the time cost is very high when testing N-cuts.

4.2. Quantitative Evaluation

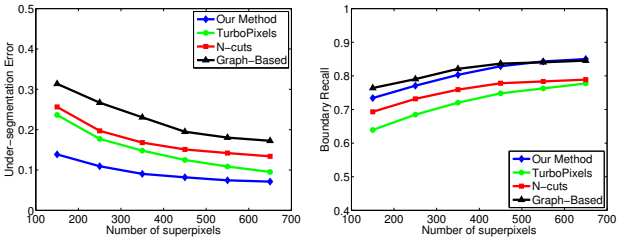
As the compactness is important for avoiding under-segmentation, we thus limit the comparison with TurboPixels and N-Cuts. We compare with these algorithms in following quantitative criteria.

Under-segmentation Error Under-segmentation Error intuitively penalties the superpixels that do not overlap tightly with a ground truth segmentation. Given a ground truth segmentation into segments $G_1, ..., G_K$ and a superpixel segmentation into superpiels $S_1, ..., S_L$, we quantify

[1]Fast Marching Toolbox is written by Gabriel Peyre (http://www.mathworks.com/matlabcentral/fileexchange/6110).

[2]Multiscale Normalized Cuts Segmentation Toolbox Version 1.6 is written by Timothee Cour, Florence Benezit, and Jianbo Shi (http://www.seas.upenn.edu/~timothee/software/ncut_multiscale/ncut_multiscale.html).

[3]TP implementation is written by Alex Levinshtein (http://www.cs.toronto.edu/~babalex/research.html).



(a) (b)

Figure 6. Performance comparison with TurboPixels and N-Cuts: (a) Under-segmentation Error, and (b) Boundary Recall.

the under-segmentation error of a whole image as:

$$U = \frac{1}{M}\left[\sum_{k=1}^{K}\left(\sum_{\{S_l||S_l \cap G_k|>B\}} Area(S_l)\right) - M\right], \quad (18)$$

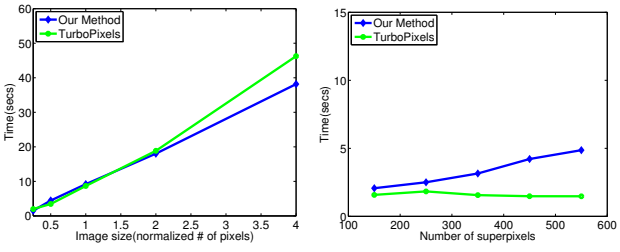
where $Area(S_l)$ is the area of the superpixel, and M is the total number of pixels. B is the minimum area of overlapping and is set to be 5% of $Area(S_l)$.

We average the value U across all test images and all ground-truth segments, and obtained a comparison in Fig. 6(a). As can be seen, our algorithm outperforms TurboPixels and N-Cuts, especially with small number of superpixels.

Boundary Recall A standard boundary recall measurement is also adopted, which computes what fraction of the ground truth edges fall within two pixel length from at least one superpixel boundary. The comparison of the boundary recall of TurboPixels, N-Cuts and our method is in Fig. 6(b). Again, with small amount of superpixels, our method outperforms the other two.

Time Cost As demonstrated in [16], TurboPixels is much faster than N-Cuts. We thus conduct comparisons with TurboPixels. In our experiments, the running time of the two algorithms is tested with respect to image size and superpixel number.

The result in Fig. 7(a) shows that our algorithm terminates within a linear time with respect to the image size, which has also been proven in Sec. 3.4. Two algorithms are comparable in time with each other. Fig. 7(b) shows the running time when increasing superpixel density under



(a) (b)

Figure 7. Timing comparison with TurboPixels (a) Running time with respect to image size (b) Running time with respect to the density of superpixels.

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

Figure 8. 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.

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 Sec. 3.4, the running time is a linear function with respect to number of iterations.

4.3. Qualitative Results

Fig. 8 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 pretty well consistent with the image contents: the density is low in the homogenous regions and high near high intensity boundaries. This makes the superpixel boundaries respect salient edges better.

Similar with the priori art [16], our algorithm is not constrained to the image-gradient-based density functions. Different kinds of refined measures can be combined in the velocity function for computing the geodesic distance. Fig. 9 shows the performance of our algorithm when combined with the Pb-based [19] boundaries. The edges between the tiger and background are much better captured.

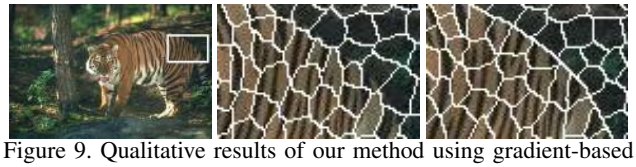


Figure 9. Qualitative results of our method using gradient-based (middle) and combined with Pb-based (right) affinity functions.

5. Application

Besides of the numerous applications as mentioned in Sec. 1, superpixels are also considered as a compact representation for image compression. Our algorithm generates better visual effects when compared with [16] due to the structure-sensitive distribution of superpixels. Fig. 10 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.

6. Conclusion

We describe a structure-sensitive over-segmentation algorithm for computing superpixels on an image. It greatly limits under-segmentation by considering the homogeneity of image appearance, density of image contents and compactness constraints. The over-segmentation is formulated as an energy minimization with the geodesic distance,

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

Figure 10. Quadratic fit to the color of the original image(a) within each superpixel got by TurboPixels(b) and our method(c). And a zoom-in on selected region showed by (c) and (d) respectively.

and the optimal solution is obtained via geometric flows and Lloyd's algorithm. Experimental results on Berkeley dataset demonstrate that our algorithm outperforms the state-of-the-art methods, and that the running time of the algorithm is comparable with that of TurboPixels.

Acknowledgements

This work is supported by National Nature Science Foundation of China (NSFC Grant) 61005037 and 90920304, Beijing Natural Science Foundation (BJNSF Grant) 4113071, and National Basic Research Program of China (973 Program) 2011CB302202.

References

[1] X. Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *ICCV*, pages 1–8. IEEE, 2007. 2

[2] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619, 2002. 1

[3] A. Criminisi, T. Sharp, and A. Blake. Geos: Geodesic image segmentation. In D. A. Forsyth, P. H. S. Torr, and A. Zisserman, editors, *ECCV (1)*, volume 5302 of *Lecture Notes in Computer Science*, pages 99–112. Springer, 2008. 2

[4] Q. Du, M. Emelianenko, and L. Ju. Convergence of the lloyd algorithm for computing centroidal voronoi tessellations. *SIAM J. Numerical Analysis*, 44(1):102–119, 2006. 4

[5] B. Feil and J. Abonyi. Geodesic distance based fuzzy clustering. *Soft Computing in Industrial Applications*, page 5059, 2007. 3, 4

[6] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004. 1, 2

[7] B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *ICCV*, pages 670–677. IEEE, 2009. 1

[8] P. M. Gruber. Optimum quantization and its applications. *Advances in Mathematics*, 186(2):456 – 497, 2004. 3

[9] V. Gulshan, C. Rother, A. Criminisi, A. Blake, and A. Zisserman. Geodesic star convexity for interactive image segmentation. In *CVPR*, pages 3129–3136. IEEE, 2010. 2

[10] X. He, R. S. Zemel, and D. Ray. Learning and incorporating top-down cues in image segmentation. In A. Leonardis, H. Bischof, and A. Pinz, editors, *ECCV (1)*, volume 3951 of *Lecture Notes in Computer Science*, pages 338–351. Springer, 2006. 1

[11] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *ICCV*, pages 654–661. IEEE Computer Society, 2005. 1

[12] I. T. Jolliffe. *Principal Component Analysis*. Springer, second edition, Oct. 2002. 5

[13] J. P. Kaufhold, R. Collins, A. Hoogs, and P. Rondot. Recognition and segmentation of scene content using region-based classification. In *ICPR (1)*, pages 755–760. IEEE Computer Society, 2006. 1

[14] J. Kim, K. hyun Shim, and S. Choi. Soft geodesic kernel k-means. In *In IEEE International Conference on Acoustics, Speech and Signal*, pages 429–432, 2007. 3

[15] A. Levinshtein, S. J. Dickinson, and C. Sminchisescu. Multiscale symmetric part detection and grouping. In *ICCV*, pages 2162–2169. IEEE, 2009. 1

[16] 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.*, 31(12):2290–2297, 2009. 1, 2, 3, 5, 6, 7

[17] S. P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–136, 1982. 2, 4

[18] T. Malisiewicz and A. A. Efros. Improving spatial support for objects via multiple segmentations. In *BMVC*. British Machine Vision Association, 2007. 2

[19] D. R. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(5):530–549, 2004. 7

[20] D. R. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, pages 416–425, 2001. 6

[21] F. Meyer and P. Maragos. Multiscale morphological segmentations based on watershed, flooding, and eikonal pde. In M. Nielsen, P. Johansen, O. F. Olsen, and J. Weickert, editors, *Scale-Space*, volume 1682 of *Lecture Notes in Computer Science*, pages 351–362. Springer, 1999. 1

[22] A. P. Moore, S. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel lattices. In *CVPR*. IEEE Computer Society, 2008. 1, 2

[23] A. P. Moore, S. J. D. Prince, and J. Warrell. "lattice cut" - constructing superpixels using layer constraints. In *CVPR*, pages 2117–2124. IEEE, 2010. 1

[24] A. P. Moore, S. J. D. Prince, J. Warrell, U. Mohammed, and G. Jones. Scene shape priors for superpixel segmentation. In *ICCV*, pages 771–778. IEEE, 2009. 1

[25] G. Mori. Guiding model search using segmentation. In *ICCV*, pages 1417–1423. IEEE Computer Society, 2005. 1

[26] I. Nwogu and J. J. Corso. $(bp)^2$: Beyond pairwise belief propagation labeling by approximating kikuchi free energies. In *CVPR*. IEEE Computer Society, 2008. 1

[27] G. Peyré, M. Péchaud, R. Keriven, and L. D. Cohen. Geodesic methods in computer vision and graphics. *Foundations and Trends in Computer Graphics and Vision*, 5(3-4):197–397, 2010. 2

[28] C. Rasmussen. Superpixel analysis for object detection and tracking with application to uav imagery. In G. Bebis, R. D. Boyle, B. Parvin, D. Koracin, N. Paragios, T. F. Syeda-Mahmood, T. Ju, Z. Liu, S. Coquillart, C. Cruz-Neira, T. Müller, and T. Malzbender, editors, *ISVC (1)*, volume 4841 of *Lecture Notes in Computer Science*, pages 46–55. Springer, 2007. 1

[29] X. Ren and J. Malik. Learning a classification model for segmentation. In *ICCV*, pages 10–17. IEEE Computer Society, 2003. 1

[30] B. C. Russell, W. T. Freeman, A. A. Efros, J. Sivic, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *CVPR (2)*, pages 1605–1614. IEEE Computer Society, 2006. 2

[31] J. Sethian. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*. Cambridge monographs on applied and computational mathematics. Cambridge University Press, 1999. 2, 4

[32] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000. 1, 2, 6

[33] J. Shotton, J. M. Winn, C. Rother, and A. Criminisi. *TextonBoost*: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In A. Leonardis, H. Bischof, and A. Pinz, editors, *ECCV (1)*, volume 3951 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2006. 5

[34] X.-C. Tai, E. Hodneland, J. Weickert, N. V. Bukoreshtliev, A. Lundervold, and H.-H. Gerdes. Level set methods for watershed image segmentation. In F. Sgallari, A. Murli, and N. Paragios, editors, *SSVM*, volume 4485 of *Lecture Notes in Computer Science*, pages 178–190. Springer, 2007. 1

[35] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(6):583–598, 1991. 1

[36] J. Wang, Y. Jia, X.-S. Hua, C. Zhang, and L. Quan. Normalized tree partitioning for image segmentation. In *CVPR*. IEEE Computer Society, 2008. 1

[37] J. Xiao and L. Quan. Multiple view semantic segmentation for street view images. In *ICCV*, pages 686–693. IEEE, 2009. 1

[38] L. Yatziv, A. Bartesaghi, and G. Sapiro. O(n) implementation of the fast marching algorithm. *Journal of Computational Physics*, 212(2):393–399, 2006. 5