Intrinsic Manifold SLIC: A Simple and Efficient Method for Computing Content-Sensitive Superpixels

Yong-Jin Liu, *Senior Member, IEEE*, Minjing Yu, Bing-Jun Li, and Ying He

Abstract—Superpixels are perceptually meaningful atomic regions that can effectively capture image features. Among various methods for computing uniform superpixels, simple linear iterative clustering (SLIC) is popular due to its simplicity and high performance. In this paper, we extend SLIC to compute content-sensitive superpixels, i.e., small superpixels in content-dense regions with high intensity or colour variation and large superpixels in content-sparse regions. Rather than using the conventional SLIC method that clusters pixels in $\mathbb{R}^5$, we map the input image I to a 2-dimensional manifold $\mathcal{M} \subset \mathbb{R}^5$, whose area elements are a good measure of the content density in I. We propose a simple method, called *intrinsic manifold SLIC* (IMSLIC), for computing a geodesic centroidal Voronoi tessellation (GCVT)—a uniform tessellation—on $\mathcal{M}$, which induces the content-sensitive superpixels in I. In contrast to the existing algorithms, IMSLIC characterizes the content sensitivity by measuring areas of Voronoi cells on $\mathcal{M}$. Using a simple and fast approximation to a closed-form solution, the method can compute the GCVT at a very low cost and guarantees that all Voronoi cells are simply connected. We thoroughly evaluate IMSLIC and compare it with eleven representative methods on the BSDS500 dataset and seven representative methods on the NYUV2 dataset. Computational results show that IMSLIC outperforms existing methods in terms of commonly used quality measures pertaining to superpixels such as compactness, adherence to boundaries, and achievable segmentation accuracy. We also evaluate IMSLIC and seven representative methods in an image contour closure application, and the results on two datasets, WHD and WSD, show that IMSLIC achieves the best foreground segmentation performance.

Index Terms—Superpixel, image segmentation, centroidal Voronoi tessellation, geodesic distance, image manifold

1 Introduction

Superpixels are perceptually meaningful atomic regions that can effectively capture image features and greatly reduce the complexity of subsequent image processing tasks, such as segmentation [1], contour closure [2], object location [3], object tracking [4], stereo 3D reconstruction [5], and many others.

Superpixels generally have the following characteristics: (1) *Connectivity:* each superpixel is a simply connected region, and each pixel in the image is assigned to exactly one superpixel; (2) *Compactness:* in the non-feature region, superpixels are regular in both size and shape; (3) *Feature preservation:* superpixels should adhere well to image boundaries; (4) *Content sensitivity:* the density of superpixels is adaptive to the co-occurrence of image contents; (5) *Performance:* computing superpixels should be fast, memory efficient, and scale well on high-resolution images; (6) *Easy to use:* users simply specify the desired number of superpixels and should not be bothered by tuning other parameters (if any).

There are two major classes of algorithms for computing superpixels, namely, graph-based and clustering-based methods. By representing an image by a graph whose nodes are pixels, the graph-based algorithms minimize a cost function defined on the graph. Representative works include normalized cuts (NC) [6], the Felzenszwalb-Huttenlocher (FH) method [7], superpixel lattices (SL) [8], and the graph-cut-based energy optimization method (GraphCut) [9]. NC generates regular and compact superpixels, but does not adhere to image boundary and has a high computational cost—time complexity $O(N^{1.5})$ for an N-pixel image as observed in [10]. FH preserves the boundary well and runs in $O(N \log N)$ time, but it produces superpixels with irregular sizes and shapes. Although SL runs in $O(N^{1.5} \log N)$ *theoretically*, it has an *empirical* time complexity $O(N)$, making it one of the fastest algorithms. However, the produced superpixels conform to grids rather than adhering to image boundaries. GraphCut is elegant and theoretically sound, but it is difficult to use in practice due to its many parameters.

The clustering-based algorithms group pixels into clusters (i.e., superpixels) and iteratively refine them until certain convergence criteria are satisfied. Popular clustering methods include TurboPixels [10], VCells [11], simple linear

- *Y.J. Liu, M. Yu, and B.J. Li are with the Tsinghua National Laboratory for Information Science and Technology, the Department of Computer Science and Technology, Tsinghua University, Haidian Qu, Beijing 100084, China. E-mail: liuyongjin@tsinghua.edu.cn, {yumj14, libj15}@mails.tsinghua.edu.cn.*
- *Y. He is with the School of Computer Science and Engineering, Nanyang Technological University, 50 Nanyang Ave, Singapore 639798, Singapore. E-mail: yhe@ntu.edu.sg.*

Manuscript received 9 Oct. 2016; revised 16 Feb. 2017; accepted 20 Mar. 2017. Date of publication 22 Mar. 2017; date of current version 13 Feb. 2018. Recommended for acceptance by T. Brox.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2017.2686857

iterative clustering (SLIC) [12], structure-sensitive[1] superpixels (SSS) [14], manifold SLIC (MSLIC) [13], and unimodular Gaussian generative model (GGM) [15]. All these methods are initialized with a set of evenly distributed seeds $\{s_i\}_{i=1}^K$ in an image domain. They differ in the way of clustering.

TurboPixels [10] generates a geometric flow for each seed and propagates them using the level set method. The superpixel boundary is defined by the points where two flows meet. Although under certain assumptions, TurboPixels has a theoretical linear time complexity $O(N)$, computational results show that it runs slowly on real-world datasets [12]. Furthermore, TurboPixels cannot guarantee that the superpixels from their numerical level-set evolution solution cover the whole image. Therefore, TurboPixels employs a postprocess to label any remaining large unassigned connected regions as new superpixels and remove very small superpixels.

VCells [11] generates a 2D Euclidean centroidal Voronoi tessellation (CVT) on the image plane and then adjusts the boundaries of the Voronoi cells to image edges. Although the main time complexity of VCells is $O(n_i\sqrt{KN})$, where n_i and K are the numbers of iterations and superpixels, the method involves a time-consuming preprocessing step that computes a 2D Euclidean CVT using Lloyd's algorithm.

SLIC [12] is an adaptation of K-means that clusters pixels in a 5-dimensional Euclidean space combining colours and images. It assigns each pixel to a cluster of the nearest seed and iteratively updates the cluster center by computing a 5D Euclidean CVT. SLIC is conceptually simple, easy to implement, and highly efficient in practice.

To compute content-sensitive superpixels, Wang et al. [14] introduced the geometric flow method [10] into a CVT optimization framework referred to as SSS. A key difference between SSS and SLIC is that SLIC measures the distance between clusters using Euclidean distance, whereas SSS takes geodesic distances into account. As a result, SLIC produces *uniform* superpixels everywhere, whereas SSS can effectively capture non-homogenous features in an image, i.e., small superpixels in content-dense regions (e.g., with high intensity or colour variation) and large superpixels in content-sparse regions. However, SSS is computationally expensive.

Manifold SLIC (MSLIC) [13] maps the input image I to a 2-dimensional manifold $\mathcal{M}$ in $\mathbb{R}^5$, whose area elements are a good measure of content density in I. Content-sensitive superpixels are then achieved by computing a restricted centroidal Voronoi tessellation (RCVT) on $\mathcal{M}$. Because the RCVT can be computed with very little lost, MSLIC runs 10 times faster than SSS.

GGM [15] measures the spatial compactness and color homogeneity in two separate terms $E_{spatial}$ and E_{color}. $E_{spatial}$ is defined by a novel Mahalanobis CVT energy in the color space, whereas E_{color} is formulated by an ordinary 2D Euclidean CVT in the image space. Since minimizing $E_{spatial}$ leads to good compactness but low segmentation accuracy, and minimizing E_{color} leads to good segmentation accuracy but seriously distorted superpixel boundary, GGM adopts a parameter λ to balance the effects. In [15], Cai and Guo suggested $\lambda E_{spatial} = E_{color}$ so the two terms make equal contribution. Such a choice of λ often results in nearly uniform superpixels in both content dense and sparse regions.

It is worth noting that all the above clustering methods cannot produce the exact number of superpixels specified by the user, because

- VCells, SLIC, SSS, and MSLIC may produce Voronoi cells consisting of disjoint components, and thus, they all adopt a split-and-merge heuristic to postprocess the superpixels;
- GGM applies a variational merging-swapping framework to minimize its composite CVT energy.

Moreover, many parameters are involved in these split-and-merge or merging-swapping steps, making parameter tuning difficult.

In this paper, we improve upon our previous work on MSLIC [13] in three respects and make the following contributions:

- Instead of using RCVT with a Euclidean metric, we compute a geodesic centroidal Voronoi tessellation (GCVT) on the image manifold $\mathcal{M}$. Thanks to the intrinsic geodesic metric, all Voronoi cells in GCVT are guaranteed to be simply connected, compact and uniform on $\mathcal{M}$.
- We establish an elegant computational framework in which GCVTs can be computed efficiently using a simple and fast approximation to a closed-form solution. Our method runs in $O(N)$ amortized time for an N-pixel image, which is independent of the number of superpixels K.
- Because GCVT completely avoids heuristic-based postprocessing, it is easy to use and exhibits consistent performance among different image data sets. Moreover, it is able to produce exactly the same number of superpixels required by the user.

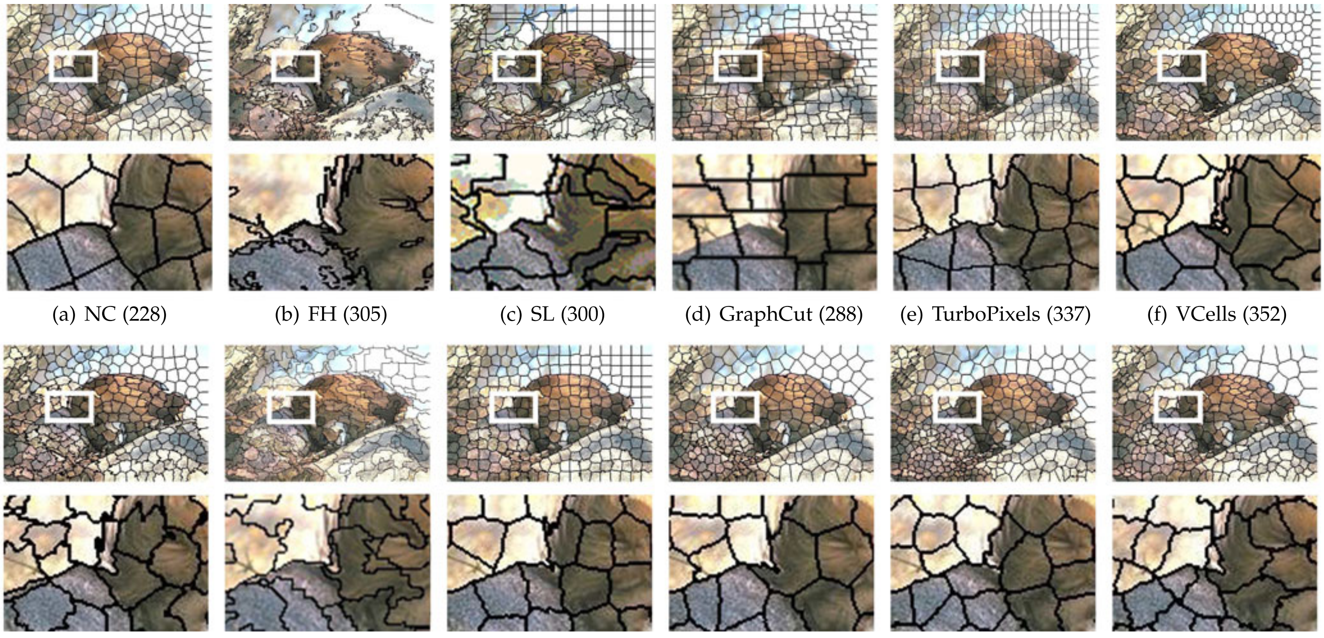
We refer to our method as *intrinsic MSLIC* (or IMSLIC) because of its intrinsic nature inherent in the geodesic metric. We thoroughly evaluate IMSLIC and eleven representative methods on two benchmarks, BSDS500 and NYUV2. Computational results show that our method outperforms the existing methods in terms of under-segmentation error, boundary recall and achievable segmentation accuracy. See Fig. 1 for an example. As a case study, we apply these superpixel methods to an image contour closure application [16]. We observe that IMSLIC produces results that are consistently better than those of the other methods, due to content sensitivity, better boundary adherence and good compactness.

2 Preliminaries on SLIC

Because our method is based on SLIC [12], we briefly introduce it before presenting our intrinsic MSLIC.

Let us denote by I an input image with N pixels. For a pixel $p \in I$, SLIC represents its colour $\mathbf{c}(p)$ in the CIELAB colour space, i.e., $\mathbf{c}(p) = (l(p), a(p), b(p))$. Given two pixels $p_1 = (u_1, v_1)$ and $p_2 = (u_2, v_2)$, SLIC measures the distance

[1] Because in computer vision, the word "structure" often refers to high-order structures in an image, in this paper we follow [13] and use the word "content".



(a) NC (228) (b) FH (305) (c) SL (300) (d) GraphCut (288) (e) TurboPixels (337) (f) VCells (352)

(g) GGM (297) (h) SEEDS (195) (i) SLIC (265) (j) SSS (316) (k) MSLIC (261) (l) Ours (300)

Fig. 1. Superpixels obtained by normalized cuts (NC) [6], the Felzenszwalb-Huttenlocher (FH) method [7], superpixel lattices (SL) [8], the graph-cut-based energy optimization method (GraphCut) [9], turboPixels [10], VCells [11], simple linear iterative clustering (SLIC) [12], structure-sensitive superpixels (SSS) [14], manifold SLIC (MSLIC) [13] and our method. The user-specified number of superpixels is 300, and the actual numbers are shown in brackets. Only SL and intrinsic MSLIC can match the user's input. Both FH and SL generate under-segmentations in content-rich regions due to the lack of a compactness constraint, whereas the other methods produce regular superpixels. It is worth noting that only SSS, MSLIC and IMSLIC can produce content-sensitive superpixels. IMSLIC outperforms the other methods in terms of under-segmentation error, boundary recall and achievable segmentation accuracy. See Section 6 for a detailed comparison and evaluation. Images are provided in high resolution for zoom-in examination.

between the pixels using a normalized Euclidean distance in the combined colour and image space $\mathbb{R}^5$

$$D(p_1, p_2) = \sqrt{\left(\frac{d_s}{N_s}\right)^2 + \left(\frac{d_c}{N_c}\right)^2}, \tag{1}$$

where N_c and N_s are two constants, and

$$d_s = \|p_1 - p_2\|_2 = \sqrt{(u_1 - u_2)^2 + (v_1 - v_2)^2}$$

$$d_c = \|\mathbf{c}(p_1) - \mathbf{c}(p_2)\|_2 = \sqrt{(l(p_1) - l(p_2))^2 + (a(p_1) - a(p_2))^2 + (b(p_1) - b(p_2))^2}.$$

Given a set $\{s_i\}_{i=1}^K$ of evenly distributed seeds in the image I, SLIC partitions I using Voronoi diagram $\{\mathcal{V}(s_i)\}_{i=1}^K$ and then iteratively improves the Voronoi cells by moving the seeds to their centers of mass, resulting in the so-called *centroidal Voronoi tessellation*.

Mathematically, a CVT is defined as follows: Let $\rho : \mathbb{R}^n \to \mathbb{R}^+$ be a density function defined on $\mathbb{R}^n$, where n is a positive integer. For a Voronoi cell $\mathcal{V}(s_i)$ of a seed s_i in $\mathbb{R}^n$, its center of mass c_i is

$$c_i = \frac{\int_{x \in \mathcal{V}(s_i)} x\rho(x)dx}{\int_{x \in \mathcal{V}(s_i)} \rho(x)dx}. \tag{2}$$

The Voronoi tessellation $\{\mathcal{V}(s_i)\}_{i=1}^K$ is a CVT if, for each Voronoi cell, the center of mass coincides with the seed, i.e., $c_i = s_i$, $1 \le i \le K$. Du et al. [17] showed that a CVT minimizes the following energy E

$$E\Big(\{s_i\}_{i=1}^K, \{\mathcal{V}(s_i)\}_{i=1}^K\Big) = \sum_{i=1}^K \int_{x \in \mathcal{V}(s_i)} \rho(x) d(x, s_i) dx, \tag{3}$$

where $d(x, y)$ is the Euclidean distance between x and y in $\mathbb{R}^n$.

With the Euclidean distance defined by Eqn. (1), SLIC computes the CVT using the Lloyd method [18], which iteratively moves each seed to the corresponding center of mass. The Lloyd method is easy to implement; however, it is computationally expensive to construct the Voronoi diagram and find the center of mass for each Voronoi cell.

To improve the performance of the Lloyd method, SLIC adopts two heuristic strategies: First, rather than explicitly constructing the Voronoi cells, for each seed s_i, it computes distances from s_i to the pixels within a $2S \times 2S$ window centered at s_i, where $S \times S$ is the expected spatial extent of each Voronoi cell, and then assigns a pixel to the seed with the least distance. This local search also distinguishes SLIC from the conventional K-means method, which has to compute the distances between s_i and *every* pixel in the image. Second, it does not compute the center of mass using the area integral in Eqn. (2). Instead, it finds the centroid c_i as the mean vector of all the pixels belonging to the Voronoi cell.

Computational results show that 10 iterations are sufficient in SLIC for most real-world images [12]. As a result, SLIC has $O(N)$ time complexity, which is independent of the number of superpixels K. In contrast, the conventional K-means method runs in $O(KNN_{iter})$, where N_{iter} is the number of iterations.

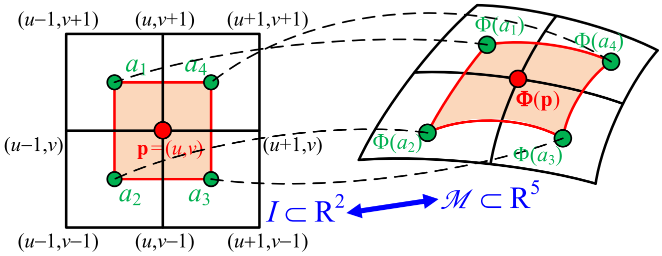
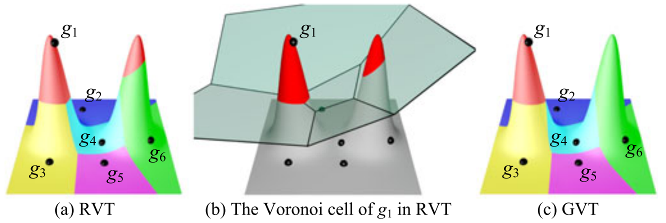


Fig. 2. Measuring area on the 2-manifold $\mathcal{M}$ embedded in $\mathbb{R}^5$.



(a) RVT (b) The Voronoi cell of g_1 in RVT (c) GVT

Fig. 4. Restricted Voronoi tessellation (RVT) and geodesic Voronoi tessellation (GVT). (a) Given a set of generators $G = \{g_1, g_2, \cdots, g_6\}$ on a 2-manifold $\mathcal{M}$, the restricted Voronoi tessellation RVT has a Voronoi cell $V(g_1)$ consisting of two disjoint components. (b) In the RVT, Voronoi cell $V(g_1)$ is bounded by five trimmed planes, which are bisectors of $\{g_1, g_j\}$, $j = 2, 3, \cdots, 6$, respectively. (c) With the same set G, all Voronoi cells in the geodesic Voronoi tessellation (GVT) are simply connected.

3 Overview

Both MSLIC [13] and the proposed intrinsic MSLIC extend SLIC to compute content-sensitive superpixels; furthermore the proposed method inherits all the favourable features of SLIC, such as simplicity and high performance.

As in SLIC [12], we represent the colour in the CIELAB colour space and denote by $\mathbf{c}(u,v) = (l(u,v), a(u,v), b(u,v))$ the colour at pixel (u,v) in the image I. Then, we define a stretching map $\Phi : I \to \mathbb{R}^5$ to send pixels to a 2-manifold $\mathcal{M}$ embedded in the 5-dimensional combined image and colour space,

$$\Phi(u,v) = (\lambda_1 p, \lambda_2 \mathbf{c}) = (\lambda_1 u, \lambda_1 v, \lambda_2 l, \lambda_2 a, \lambda_2 b), \qquad (4)$$

where λ_1 and λ_2 are global stretching factors. We set $\lambda_1 = \frac{1}{N_s}$ and $\lambda_2 = \frac{1}{N_c}$ and adopt the same Euclidean metric D in Eqn. (1) in the combined image and colour space $\mathbb{R}^5$. The metric D remains the same for all images.

For a pixel $p = (u,v)$, we denote by $\square_p$ the unit square 1×1 centered at p. Refer to Fig. 2. Let a_1, a_2, a_3 and a_4 be the four corners of $\square_p$, each of which is determined by the average of the four neighbouring pixels. The square $\square_p$ consists of two triangles, i.e., $\square_p = \triangle a_1 a_2 a_3 \bigcup \triangle a_3 a_4 a_1$. Then we approximate the area of the *curved* triangle $\Phi(\triangle a_1 a_2 a_3)$ by the area of the *planar* triangle $\triangle \Phi(a_1)\Phi(a_2)\Phi(a_3)$,

$$Area(\Phi(\triangle a_1 a_2 a_3)) \approx \frac{1}{2} \|\overrightarrow{\Phi(a_2)\Phi(a_1)}\| \|\overrightarrow{\Phi(a_2)\Phi(a_3)}\| \sin\theta,$$

where θ is the angle between vectors $\overrightarrow{\Phi(a_2)\Phi(a_1)}$ and $\overrightarrow{\Phi(a_2)\Phi(a_3)}$, and the length $\|\overrightarrow{\Phi(x)\Phi(y)}\|$ is measured using the metric D in Eqn. (1). $Area(\Phi(\triangle a_3 a_4 a_1)$ can be computed similarly. The area of a region $\Phi(\Omega) \subset \mathcal{M}$ is simply the sum of $Area(\Phi(\square_{p_i}))$ for all pixels $p_i \in \Omega$.

Our method is based on an important observation: for a region $\Omega \subset I \subset \mathbb{R}^2$, the area of the corresponding region $\Phi(\Omega) \subset M$ on $\mathcal{M}$ depends on both the area of Ω and the intensity or colour variation in Ω. The higher the variation of colours in Ω, the larger the area of $\Phi(\Omega)$, and vice versa. If we can compute a *uniform* tessellation on $\mathcal{M}$, the inverse mapping Φ^{-1} will induce the *content-sensitive* tessellation on I. See Fig. 3.

To find a uniform tessellation on $\mathcal{M}$, MSLIC [13] computes the *restricted* centroidal Voronoi tessellation (RCVT), which restricts the Euclidean centroidal Voronoi tessellation in $\mathbb{R}^5$

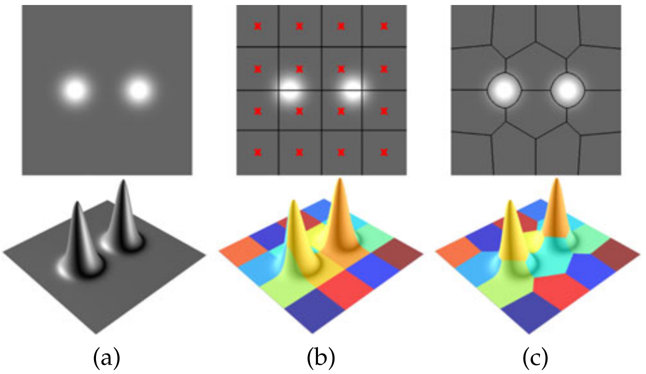
$$\mathcal{V}_{\mathbb{R}^5}(g_i) = \{x \in \mathbb{R}^5 | \|x - g_i\|_2 \leq \|x - g_j\|_2, \forall j \neq i, g_i, g_j \in G\}, \text{and} g_i = \frac{\int_{x\in\mathcal{V}_{\mathbb{R}^5}(g_i)} x dx}{\int_{x\in\mathcal{V}_{\mathbb{R}^5}(g_i)} 1 dx}, \qquad (5)$$

to $\mathcal{M}$ as

$$RCVT(G, \mathcal{M}) = \{\mathcal{V}_{\mathcal{M}}(g_i) \neq \emptyset | \forall g_i \in G\}, \qquad (6)$$

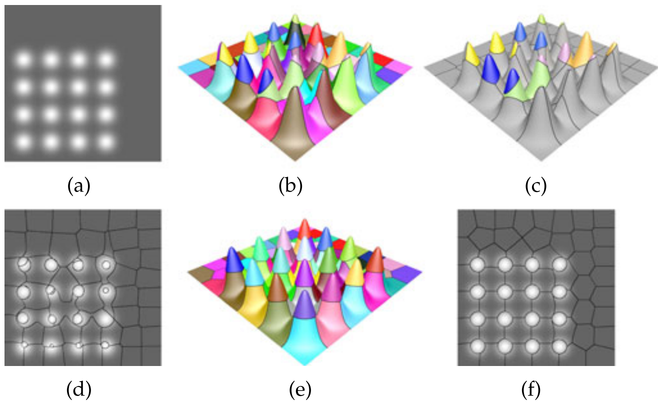
where G is a generator set and $\mathcal{V}_{\mathcal{M}}(g_i) \triangleq \mathcal{M} \bigcap \mathcal{V}_{\mathbb{R}^5}(g_i)$.

One problem in RCVT is that certain Voronoi cells can consist of disjoint components; see Figs. 4a and 4b. We observe that in real images, Voronoi cells in RCVT consisting of disjoint components frequently appear; see Fig. 5 for an example. Accordingly, a split-and-merge operator with



(a) (b) (c)

Fig. 3. Overview of a uniform tessellation on a synthetic greyscale image I. (a) We represent I as a 2-manifold embedded in $\mathbb{R}^3$, denoted by $\mathcal{M} = \Phi(u,v) \subset \mathbb{R}^3$, whose area elements are a good measure of content density in I. (b) A uniform tessellation of $K(=16)$ seeds $\{s_i\}_{i=1}^K$ in I leads to a non-uniform tessellation on $\mathcal{M}$. (c) A uniform Voronoi tessellation on $\mathcal{M}$, in which all cells are similar in size and conform to the geometric features on the surface, induces the content-sensitive superpixels in the image domain.



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

Fig. 5. (a) A 400×400 grey image. (b) The RCVT on the 2-manifold $\mathcal{M} = \Phi(I)$. (c) The Voronoi cells consisting of disjoint components in RCVT are shown by colours. (d) Superpixels computed by MSLIC [13] without the split-and-merge postprocessing. (e) The GCVT proposed in this paper, in which each of its Voronoi cells is simply connected. (f) Superpixels output from intrinsic MSLIC proposed in this paper.

TABLE 1
Major Notations Used in This Paper

MSLIC	Extrinsic manifold SLIC [13]
IMSLIC	Intrinsic manifold SLIC proposed in this paper
$S = \{s_i \in I\}_{i=1}^K$	The set of K seeds in image I
$G = \{\Phi(s_i)\}_{i=1}^K$	The set of K generators on image manifold $\mathcal{M}$
$d_g(x, y)$	Geodesic distance between x and y on $\mathcal{M}$
$\mathcal{V}_g(\Phi(s_i))$	Geodesic Voronoi cell (GVC) of the generator $\Phi(s_i)$ on $\mathcal{M}$
$GVT(G,\mathcal{M}) = \cup_{i=1}^K \mathcal{V}_g(\Phi(s_i))$	Geodesic Voronoi tessellation (GVT) of G on $\mathcal{M}$
$GCVT(G,\mathcal{M})$	Geodesic centroidal Voronoi tessellation (GCVT) of G on $\mathcal{M}$
$\mathcal{G}(P, E)$	Weighted image graph whose node set P includes all pixels in I and E includes edges $e = (p_i, p_j)$ with weights $l(e) = \|\Phi(p_i) - \Phi(p_j)\|_2$, where p_i and p_j are neighboring pixels in 8-connectivity.

extra heuristic parameters has to be used in MSLIC to merge tiny superpixels and split large superpixels. The number of output superpixels can only approximate the number designated by users. Furthermore, the optimal values of extra heuristic parameters in the split-and-merge operator may vary between different image data sets, resulting in bad scalability for MSLIC.

In this paper, we propose an intrinsic MSLIC that computes a uniform tessellation on $\mathcal{M}$ by computing a *geodesic* centroidal Voronoi tessellation (GCVT). The formulation of GCVT is presented Section 4, and a simple and fast approximation solution for computing GCVT is presented in Section 5. Due to the geodesic metric, each Voronoi cell in GCVT is guaranteed to be a simply connected region and intrinsic MSLIC can output the exact number of superpixels designated by users. Intrinsic MSLIC is simple and fast, without heuristic parameters to tune.

Table 1 summarizes the notations used in this paper.

4 Geodesic Voronoi Tessellation and GCVT

Let $S = \{s_i | s_i \in I, 1 \le i \le K\}$ be the set of seeds in the image I and $G = \{\Phi(s_i)\}_{i=1}^K$ be the set of corresponding generators on the manifold $\mathcal{M}$. The geodesic Voronoi cell $\mathcal{V}_g(\Phi(s_i))$ on $\mathcal{M}$ is defined as

$$\begin{aligned}\mathcal{V}_g(\Phi(s_i)) = \{x \in \mathcal{M} : d_g(x, \Phi(s_i)) \le d_g(x, \Phi(s_j)),\\ \forall j \neq i, 1 \le i, j \le K\},\end{aligned} \tag{7}$$

where $d_g(x, y)$ is the geodesic distance between x and y on $\mathcal{M}$. The geodesic Voronoi tessellation $GVT(G,\mathcal{M})$ of G on $\mathcal{M}$ is the union of all Voronoi cells, i.e., $GVT(G,\mathcal{M}) = \cup_{i=1}^K \mathcal{V}_g(\Phi(s_i))$.

Definition 1. *For each geodesic Voronoi cell $\mathcal{V}_g(\Phi(s_i))$, the nominal mass centroid m_i of $\mathcal{V}_g(\Phi(s_i))$ on $\mathcal{M}$ is defined to be the solution of the following problem*

$$\min_{z \in \mathcal{M}} \mathcal{E}_i(z), where\ \ \mathcal{E}_i(z) = \int_{y \in \mathcal{V}_g(\Phi(s_i))} d_g^2(y, z) dy. \tag{8}$$

Since $\mathcal{V}_g(\Phi(s_i))$ and $\mathcal{M}$ are non-empty compact sets in $\mathbb{R}^n$, by a simple adaption of Lemma 3.4 in [17] and Theorem 1.13 in [19], it can be shown that $\mathcal{E}_i(z)$ is continuous and has at least one global minimum point in $\mathcal{V}_g(\Phi(s_i))$.

Definition 2. *Let $X = \{x_i\}_{i=1}^K$ be a set of points on $\mathcal{M}$. A geodesic Voronoi tessellation $GVT(X,\mathcal{M})$ is a geodesic centroidal Voronoi tessellation (GCVT) if each generator $x_i \in X$ is the nominal mass centroid of its geodesic Voronoi cell.*

Theorem 1. *Let $\mathcal{M}$ be a 2-manifold embedded in $\mathbb{R}^n$ and $K \in \mathbb{Z}_+$ a positive integer. For an arbitrary set of points $X = \{x_i\}_{i=1}^K \in \mathcal{M}$ and an arbitrary tessellation $\{V_i\}_{i=1}^K$ on $\mathcal{M}$, $\bigcup_{i=1}^K V_i = \mathcal{M}$, $V_i \bigcap V_j = \emptyset$, $\forall i \neq j$, define the CVT energy functional as follows:*

$$\mathcal{E}\Big(\{(x_i, V_i)\}_{i=1}^K\Big) = \sum_{i=1}^K \int_{y \in V_i} d_g^2(y, x_i) dy. \tag{9}$$

Then the necessary *condition for $\mathcal{E}$ being minimized is that $\{(x_i, V_i)\}_{i=1}^K$ is a GCVT of $\mathcal{M}$.*

Proof. The proof consists of two parts: (1) if $\{V_i\}_{i=1}^K$ is fixed, then minimization of $\mathcal{E}$ requires that each x_i is the nominal mass centroid of V_i, and (2) if $\{x_i\}_{i=1}^K$ is fixed, then minimization of $\mathcal{E}$ requires that $\{V_i\}_{i=1}^K$ is the geodesic Voronoi tessellation $GVT(X,\mathcal{M})$.

First, by fixing $\{V_i\}_{i=1}^K$ and $\{x_i\}_{i=1}^K$ except for a single point x_j, we have

$$\mathcal{E}(x_j) = \int_{y \in V_j} d_g^2(y, x_j) dy + C,$$

where C is a constant. By Definition 1, to minimize $\mathcal{E}(x_j)$, x_j is the nominal mass centroid of the region V_j.

Second, we fix the positions of points in X and choose a tessellation $\{V_i\}_{i=1}^K$ other than the geodesic Voronoi tessellation $GVT(X,\mathcal{M}) = \cup_{i=1}^K \mathcal{V}_g(x_i)$. Then we compare $\mathcal{E}(\{(x_i, \mathcal{V}_g(x_i))\}_{i=1}^K)$ with $\mathcal{E}(\{(x_i, V_i)\}_{i=1}^K)$. For any point $y \in \mathcal{M}$ belonging to a Voronoi cell $\mathcal{V}_g(x_i)$, we have

$$d_g(y, x_i) \le d_g(y, x_j),\ \ x_j \in X. \tag{10}$$

Since $\{V_i\}_{i=1}^K$ is not a Voronoi tessellation, Eqn. (10) must hold with strict inequality over some measure nonzero set in $\mathcal{M}$. Accordingly,

$$\mathcal{E}(\{(x_i, \mathcal{V}_g(x_i))\}_{i=1}^K) < \mathcal{E}(\{(x_i, V_i)\}_{i=1}^K).$$

Thus $\mathcal{E}$ is minimized when $\{V_i\}_{i=1}^K$ are chosen to be $\{\mathcal{V}_g(x_i)\}_{i=1}^K$. □

In Section 5, the GCVT on $\mathcal{M}$ is computed by the Lloyd method [18], which iteratively moves each generator to the corresponding nominal mass centroid:

1) Select an initial set of $X_j = \{x_i^j\}_{i=1}^K$, $j = 0$, on $\mathcal{M}$;
2) Construct $GVT(X_j, \mathcal{M})$;
3) Compute the nominal mass centroid of each Voronoi cell $\mathcal{V}_g(x_i^j)$ in $GVT(X_j, \mathcal{M})$ and place all centroids into a new set X_{j+1};
4) If X_{j+1} satisfies termination conditions, then stop; otherwise, let $j \leftarrow j + 1$ and return to Step 2.

The convergence of the Lloyd algorithm derives directly from the proof of Theorem 1

$$\begin{aligned}&\mathcal{E}(X_{j+1}, GVT(X_{j+1}, \mathcal{M})) \leq \mathcal{E}(X_{j+1}, GVT(X_j, \mathcal{M}))\\ &\leq \mathcal{E}(X_j, GVT(X_j, \mathcal{M})).\end{aligned} \tag{11}$$

Compared with the constrained CVT (CCVT) [20] and the restricted CVT (RCVT) [13], our proposed nominal mass centroids and GCVT use the geodesic distance on $\mathcal{M}$ that guarantee the single connectivity of each Voronoi cell.

5 Computing Geodesic CVT

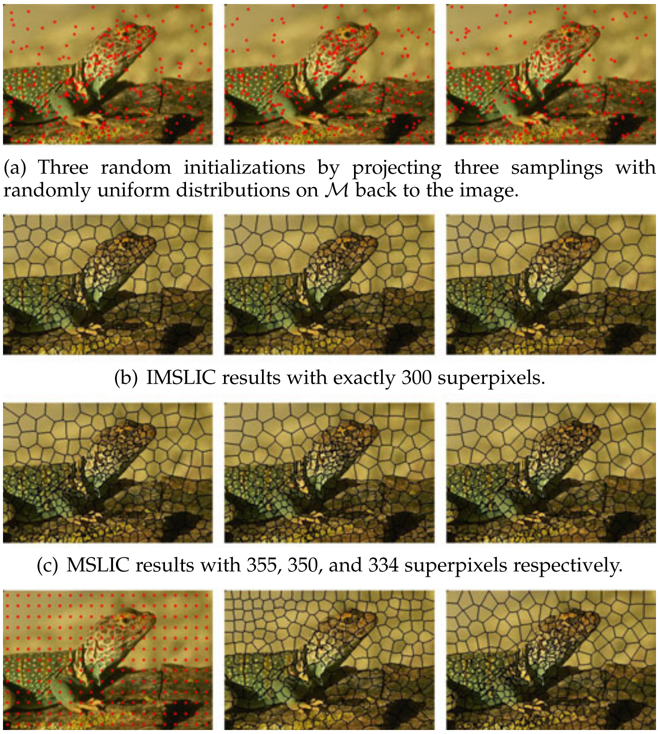
It is expensive to compute *exact* GVTs on curved manifolds [21], [22], [23]. To quickly compute GCVTs on image manifold $\mathcal{M}$, we propose four key ingredients: (1) a good initialization of generators on $\mathcal{M}$ (Section 5.1), (2) a local adaptive search strategy for Voronoi cells (Section 5.2), (3) discrete geodesics on an image graph (Section 5.3), and (4) a fast approximate solution to nominal mass centroid (Section 5.4). We refer to our method as *intrinsic MSLIC* (IMSLIC); its pseudo-code is presented in Algorithm 1.

Algorithm 1. Intrinsic MSLIC

```
Input: An image $I$ of $N$ pixels, the desired number of superpixels $K$, the maximal number of iterations $iter_{max}$ and the convergence threshold $\varepsilon$.
Output: $K$ content-sensitive superpixels.
Randomly initialize seeds $\{s_i\}_{i=1}^K$ with uniform distribution (Section 5.1).
Initialize label $l(p) = -1$ and distance $d(p) = \infty$ for each pixel $p$.
Initialize the residual error $err = \infty$ and $iter = 0$.
Compute $Area(\Phi(\Box_p))$ for each pixel $p \in I$ (Section 3).
Compute a local search range $\Xi = \frac{4\sum_{i=1}^K Area(\Phi(\Box_{p_i}))}{K}$.
while $err > \varepsilon$ and $iter \leq iter_{max}$ do
    for each seed $s_i$ do
        Compute $Area(\Phi(\Omega(s_i)))$ of a $2S \times 2S$ region $\Omega(s_i)$ centered at $s_i$.
        Compute a scaling factor $\lambda_i = \sqrt{\frac{\Xi}{Area(\Phi(\Omega(s_i)))}}$.
    end for
    for each seed $s_i$ do
        for each pixel $p$ in a $2\lambda_i S \times 2\lambda_i S$ region centered at $s_i$ do
            Compute the geodesic distance $D = d_g(\Phi(s_i), \Phi(p))$ (Section 5.3).
            if $D < d(p)$ then
                $d(p) = D$; $l(p) = i$.
            end if
        end for
    end for
    $err = 0$.
    for each seed $s_i$ do
        Compute $Area(\mathcal{V}_{\mathcal{M}}(\Phi(s_i)))$.
        Compute the nominal mass centroid $m_i$ of $\mathcal{V}_g(\Phi(s_i))$ (Section 5.4) and set $c_i = P(m_i)$ using Eqn. (12).
        $err$ += $d_p(\Phi(c_i), \Phi(s_i))$.
        $s_i \leftarrow c_i$.
    end for
    $iter$ ++.
end while
```

5.1 Initialization

The Lloyd algorithm specified by Eqn. (11) is a local optimization method, and a good initialization is important to achieve good performance. The goal of our content-sensitive superpixels is to compute a uniform tessellation on $\mathcal{M}$. Then a good initialization is a set of generators $\{\Phi(s_i)\}_{i=1}^K$ that are uniformly distributed on $\mathcal{M}$.


(a) Three random initializations by projecting three samplings with randomly uniform distributions on $\mathcal{M}$ back to the image.

(b) IMSLIC results with exactly 300 superpixels.

(c) MSLIC results with 355, 350, and 334 superpixels respectively.

(d) Superpixels by IMSLIC (middle) and MSLIC (right) starting at a uniform initialization on I (left). IMSLIC and MSLIC generate 300 and 366 superpixels respectively.

Fig. 6. Comparison of different initializations with $K = 300$. The commonly used quality measures on these superpixels resulting from different initializations are summarized in Table 2.

To quickly generate a good initialization, we randomly generate K samples on $\mathcal{M}$ with uniform distribution. First we generate an array A with its indices related to the number of pixels in I, i.e., $A[i]$ corresponds to the pixel $p_i \in I$. Each element in A stores the accumulated pixel areas, i.e., $A[i-1] = \sum_{j=1}^i Area(\Phi(\Box_{p_j}))$. Second, we apply a random number generator using a computer's real time clock as the seed to sample between 0 and $A[N-1]$. Finally, each generated random number x offers a random sample $\Phi(p_j)$ on $\mathcal{M}$, where the index j satisfies $A[j-1] < x \leq A[j]$. If a point $\Phi(p_j)$ is sampled twice, another new point is randomly sampled.

When projecting randomly and uniformly distributed samples back to the image by the inverse map $\Phi^{-1} : \mathcal{M} \rightarrow I$, they are dense in content-dense regions and sparse in other regions (Fig. 6a). Starting with the proposed initialization, IMSLIC (Algorithm 1) can generate better superpixels than MSLIC [13] (Figs. 6b, 6c, and 6d) in terms of the commonly used quality measures (Table 2); see Section 6 for detailed discussions.

5.2 A Local Adaptive Search Strategy

The bottleneck of the Lloyd method is constructing the Voronoi tessellation $GVT(G, \mathcal{M})$ for the set of seeds $G = \{\Phi(s_i)\}_{i=1}^K$. Inspired by the local search strategy in SLIC, we develop a *local* method to approximate $GVT(G, \mathcal{M})$ in $O(N)$ time.

TABLE 2
The Commonly Used Quality Measures (See Section 6 for Details on USE, ASA and BR) for the Superpixels Shown in Fig. 6. The Number of Output Superpixels Is Denoted as #output_sp. Only IMSLIC Can Produce Exactly the Same Number of Superpixels Required by the User. Starting with Any of Three Random Initializations, IMSLIC Always Outperforms MSLIC. IMSLIC with Random Initializations Also Outperforms IMSLIC and MSLIC with Uniform Initializations

Metric		USE	ASA	BR	#output_sp
Random	IMSLIC	0.045	0.993	0.919	300
initialization 1	MSLIC	0.066	0.982	0.828	355
Random	IMSLIC	0.041	0.993	0.926	300
initialization 2	MSLIC	0.061	0.986	0.874	350
Random	IMSLIC	0.049	0.993	0.924	300
initialization 3	MSLIC	0.056	0.989	0.876	334
Uniform	IMSLIC	0.065	0.982	0.817	300
initialization 3	MSLIC	0.060	0.984	0.864	366

Observe that GCVT tends to generate uniform geodesic Voronoi cells (GVCs) on $\mathcal{M}$. Therefore, the area of each GVC is roughly $\frac{Area(\mathcal{M})}{K}$. To speed up the construction of a GVC, we limit the size of the search area for each generator to $\Xi = \frac{4Area(\mathcal{M})}{K}$. To quickly find the limited search areas, for each GVC on $\mathcal{M}$, we project its nominal mass centroid back to the image plane by

$$P(m) = (u, v), \tag{12}$$

where $m = (u, v, \frac{\lambda_2}{\lambda_1} l, \frac{\lambda_2}{\lambda_1} a, \frac{\lambda_2}{\lambda_1} b) \in \mathbb{R}^5$. We use a $2S \times 2S$ region Ω centered at $P(m)$ in the image I as the initial guess, where $S = \sqrt{N/K}$. Then we adjust the local region Ω by calculating an adaptive scaling factor $\lambda = \sqrt{\Xi / Area(\Phi(\Omega))}$. Finally we approximate the limited searching area by the set $\{\Phi(p) | p \in \Omega' \subset I\}$, where Ω' is the $2\lambda S \times 2\lambda S$ region centered at $P(m)$ in image I.

Now we show that Algorithm 1 runs in $O(N)$ time, which is independent of the number of superpixels K. The key is to show that the scaling factor λ_i for each GVC is bounded by a constant $c_{\max} = \max\{Area(\Phi(\Box_{p_j})), p_j \in I\}$, which is determined by the colour variation of the image I. To see this, note that λ_i reaches the maximum when $Area(\Phi(\Omega_i))$ is minimum. $Area(\Phi(\Omega_i))$ is minimized when $\Phi(\Omega_i)$ is flat (has the same colour), i.e., $Area(\Phi(\Omega_i)) = 4S^2 = 4N/K$. Therefore, $\lambda_i = \sqrt{\Xi / Area(\Phi(\Omega_i))} \leq \sqrt{Area(\mathcal{M})/N} \leq \sqrt{c_{\max}}$.

In each local search region, there are at most $2\lambda_i S \times 2\lambda_i S \leq 4c_{\max} N/K$ pixels. Then the total number of visited pixels in the K search regions is bounded by $4c_{\max} N$. The discrete geodesic algorithm in Section 5.3 assigns geodesic distances from all samples $\Phi(p_i)$, $\forall p_i \in I$, to their nearest generators in $O(N)$ amortized time. Putting it all together, Algorithm 1 takes linear amortized time $O(N)$.

5.3 Discrete Geodesics

To compute the geodesic distance on the image manifold $\mathcal{M}$, we build a weighted image graph $\mathcal{G}(P, E)$ as follows. The node set P includes all pixels in the image I. An edge $e = (p_i, p_j)$ is in E if p_i and p_j are 8-connected neighbouring pixels in I. The weight for each edge e is $l(e) = \|\Phi(p_i) - \Phi(p_j)\|_2$, where $\|\cdot\|_2$ is the Euclidean distance in $\mathbb{R}^5$.

In Algorithm 1 (line 13), for each generator $\Phi(s)$, we need to compute geodesic distances from all samples in the limited search area $\Phi(\Omega'(s)) \subset \mathcal{M}$ to $\Phi(s)$. Let $\mathcal{G}_s(P_s, E_s)$ be an induced subgraph of $\mathcal{G}$, in which P_s contains all pixels in $\Omega'(s)$ and $E_s = \{(p_i, p_j) \in E : p_i, p_j \in P_s\}$. We can apply Dijkstra's algorithm on the weighted subgraph $\mathcal{G}_s$ to compute the shortest paths from $\Phi(s)$ to all pixels in P_s, which serve as discrete geodesics on $\mathcal{M}$.

In each local search region, there are at most $2\lambda S \times 2\lambda S \leq 4c_{\max} N/K$ pixels. Dijkstra's algorithm maintains a priority queue Q and runs in $O(\frac{N}{K} \log(\frac{N}{K}))$ time. In particular, by extracting the pixel with the smallest label in Q at each iteration, Dijkstras algorithm is known as a *label setting* method, since the pixels extracted from Q are permanently labelled and never re-enter Q. In IMSLIC, we adopt a *label correcting* method [24] that maintains a bucket data structure B. Each bucket in B is implemented as a first-in-first-out queue, such that the selection of the to-be-processed pixel takes only $O(1)$ time, at the expense of multiple entrances of nodes in B. Among many label correcting schemes [25], [26], we observed that the threshold scheme (Section 2.4.4 in [26]) performs very well in our practice and significantly outperforms the original Dijkstras algorithm on real-world sparse graphs. The threshold scheme that we implemented runs in amortized time $O(N/K)$ [26]. Therefore, in Algorithm 1, the discrete geodesic computation for all K local search regions takes linear amortized time $O(N)$.

5.4 A Fast Approximate Solution to Nominal Mass Centroid Computation

Given a GVC $\mathcal{V}_g(\Phi(s_i)) \subset \mathcal{M}$, it is generally difficult and expensive to compute its nominal mass centroid m_i as in Definition 1. We propose a simple and fast method to estimate it as follows.

Let $\mathcal{G}_v = (P_v, E_v)$ be a subgraph of $\mathcal{G}(P, E)$ induced by $\mathcal{V}_g(\Phi(s_i))$, i.e., P_v includes all pixels in $\{p_j : \Phi(p_j) \in \mathcal{V}_g(\Phi(s_i))\}$, and $E_v = \{(p_i, p_j) \in E : p_i, p_j \in P_v\}$. Our idea is to embed the subgraph $\mathcal{G}_v$ in $\mathbb{R}^2$ and ensure that the length of each embedded edge is as close to its weight as possible. Since discrete geodesics are computed as the shortest paths in $\mathcal{G}_v$, if the length of each embedded edge is preserved, the geodesic distance is also preserved. To realize the idea with low computational overhead, we use the landmark MDS (LMDS) algorithm [27]. Let $p_i = s_i$, and p_j and p_k are two arbitrary pixels in $\mathcal{G}_v$, $i \neq j \neq k$. Without loss of generality, assume $d_g(\Phi(p_i), \Phi(p_j)) \geq d_g(\Phi(p_j), \Phi(p_k)) \geq d_g(\Phi(p_i), \Phi(p_k))$. To make our embedding robust, we require that $d_g(\Phi(p_i), \Phi(p_k)) + d_g(\Phi(p_j), \Phi(p_k)) > 1.2 d_g(\Phi(p_i), \Phi(p_j))$. Otherwise, we arbitrarily choose another pixel until this property holds. The three samples $\Phi(p_i)$, $\Phi(p_j)$ and $\Phi(p_k)$ are the *landmarks* in the general position.

LMDS [27] operates in two steps:

- First, three landmarks are embedded into $\mathbb{R}^2$. Let

$$\Delta_3 = \begin{pmatrix} 0 & d_g^2(i,j) & d_g^2(i,k) \\ d_g^2(j,i) & 0 & d_g^2(j,k) \\ d_g^2(k,i) & d_g^2(k,j) & 0 \end{pmatrix},$$

where $d_g^2(x, y) = d_g^2(\Phi(p_x), \Phi(p_y))$,

$$H_3 = \begin{pmatrix} -1/3 & 2/3 & 2/3 \\ 2/3 & -1/3 & 2/3 \\ 2/3 & 2/3 & -1/3 \end{pmatrix},$$

and $B_3 = -H_3\Delta_3 H_3/2$. Denote the eigenvalues and corresponding eigenvectors (written as column vectors) of B_3 as $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$ and $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ respectively. The optimal embedding vectors of the three landmarks are the rows of the matrix $(\sqrt{\lambda_1}\mathbf{v}_1 \quad \sqrt{\lambda_2}\mathbf{v}_2 \quad \sqrt{\lambda_3}\mathbf{v}_3)$.

- Second, we embed the remaining pixels in $\mathcal{G}_v$ into $\mathbb{R}^2$. Let p_l be a pixel in $\mathcal{G}_v$, $l \neq i, j, k$, and let $\Delta_k = \left(d_g^2(l,i) \quad d_g^2(l,j) \quad d_g^2(l,k)\right)^T$ be a column vector. The optimal embedding vector of p_l is computed by $\left(\mathbf{v}_1/\sqrt{\lambda_1} \quad \mathbf{v}_2/\sqrt{\lambda_2} \quad \mathbf{v}_3/\sqrt{\lambda_3}\right)^T(\overline{\Delta}_3 - \Delta_k)/2$, where $\overline{\Delta}_3$ is the column mean of Δ_3.

If the square geodesic distances between $\Phi(p_l)$ and the three landmarks are known, the embedding position of a pixel p_l can be simply computed by multiplying a 3×3 matrix by a 3×1 column vector. Since p_i is the seed s_i, the geodesic distance between each $\Phi(p_l)$ and the generator $\Phi(s_i)$ is already known.[2] We only need to run the threshold algorithm twice starting at $\Phi(p_j)$ and $\Phi(p_k)$.

After $\mathcal{G}_v$ is embedded into $\mathbb{R}^2$, the nominal mass centroid m_i is approximated by a point on $\mathcal{M}$ whose embedded point is the planar mass centroid of all embedded pixels in $\mathcal{G}_v$.

6 Experimental Results

We implemented IMSLIC in C++ and tested it on a PC with an Intel E5-2650 CPU (2.60 GHz) and 64 GB RAM. The executable of IMSLIC is available.[3] We compared IMSLIC with eleven representative methods: NC [6], FH [7], SL [8], GraphCut [9], TurboPixels [10], VCell [11], GGM [15], SEEDS [28], SLIC [12], SSS [14] and MSLIC [13]. The comparison was performed on two datasets: the Berkeley Segmentation Dataset (BSDS500) [29] and the NYU Depth Dataset (NYUV2) [30], where each image has a ground-truth segmentation. Except for SSS, which was implemented by us, we used the source code of all other methods provided by the authors. We also examined the segmentation effect using different superpixels in an image contour closure application [16]. Quantitative and qualitative results showed that IMSLIC outperforms these representative methods in terms of commonly used quality measures for superpixels (Sections 6.1 and 6.2) and the F-measure in the application (Section 6.3).

6.1 BSDS500 Results

BSDS500 consists of 500 natural images with a resolution of 482×321. We evaluate the methods by averaging the commonly used performance metrics on all 500 images. The metrics include adherence to boundaries, achievable segmentation accuracy, and run time. Because IMSLIC adopts a random initialization, we report the average results of 100 initializations.

Adherence to Boundaries. As dense over-segmentation of images, superpixels should well preserve the boundary of ground-truth segmentations. Under segmentation error and boundary recall are the standard measures for boundary adherence [10], [12], [14]. The former measures the tightness of superpixels that overlap with a ground-truth segmentation, and the latter measures what fraction of the ground truth edges fall within at least two pixels of a superpixel boundary. Denote a ground-truth segmentation of an image as $G = \{g_1, g_2, \cdots, g_l\}$, and a superpixel over-segmentation as $S = \{s_1, s_2, \cdots, s_r\}$. The under-segmentation error E_U of a segment g_i is defined as

$$E_U = \frac{\sum_{\{s_j \in S: Area(s_j \cap g_i) > B_j\}} Area(s_j) - Area(g_i)}{Area(g_i)},$$

where $Area(x)$ is the image area (in pixels) of a segment x, and we follow [12] to set B_j to 5 percent of $Area(s_j)$. E_U of an image is the average of all segments in G. Boundary recall is defined as the faction of the ground-truth edges falling within a two-pixel distance from at least one superpixel boundary. A high boundary recall means that very few true edges are missed. As shown in Figs. 7a and 7b, superpixels generated by IMSLIC, VCells and GGM have the lowest under-segmentation error (for $K > 300$) and the highest boundary recall ratio, demonstrating the ability of these three methods to adhere to image boundaries.

Achievable Segmentation Accuracy (ASA). Superpixels can be used as a preprocessing step for the subsequent segmentation algorithms. ASA, defined as the highest accuracy in all possible segmentations that use superpixels as input, is the upper bound of accuracy of a segmentation [14]. The ASA of an image I is defined as

$$ASA(I) = \frac{\sum_{s_j \in S} \max_{g_i \in G}\{Area(s_j \cap g_i)\}}{\sum_{g_k \in G} Area(g_k)}.$$

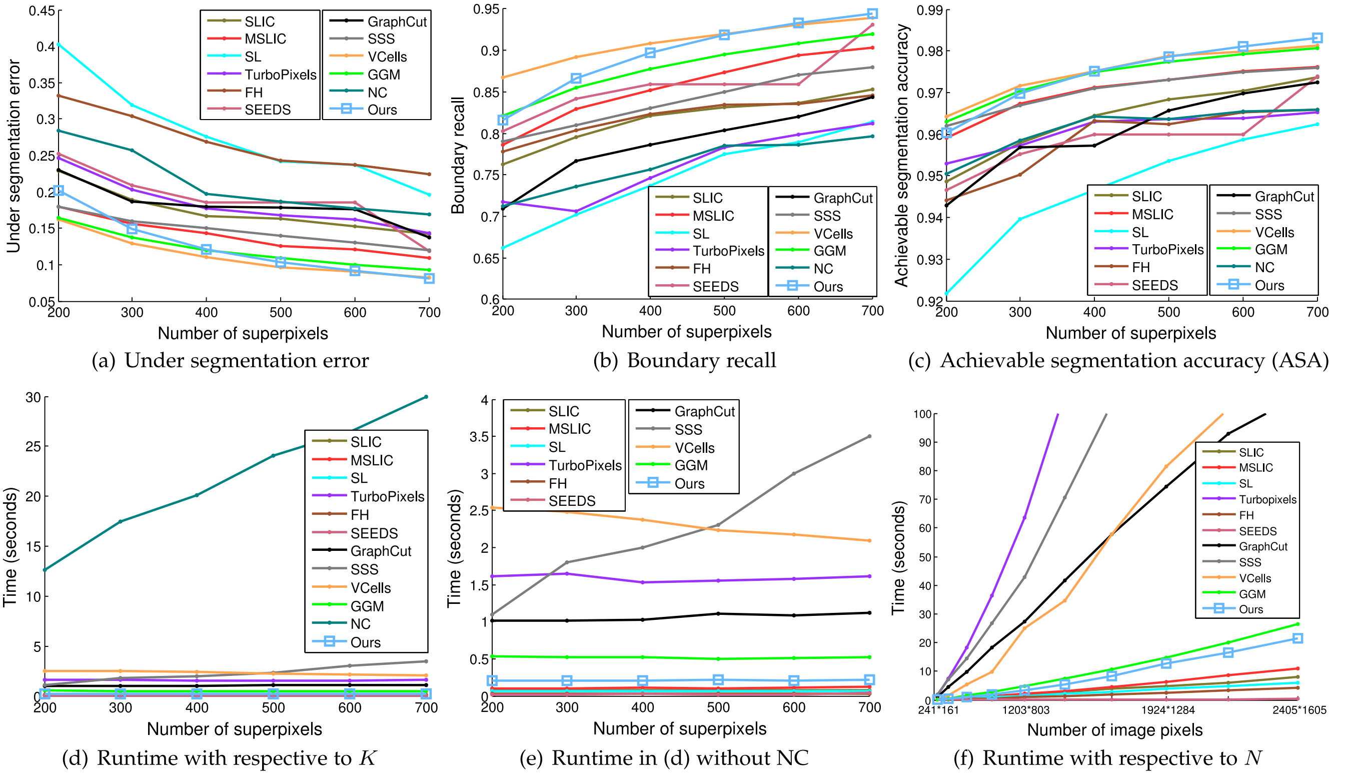
In general, the more superpixels there are, the better ASA one can obtain. As shown in Fig. 7c, the superpixels produced by IMSLIC, VCells and GGM have the best ASA values for $K \geq 300$.

Runtime Performance. Similarly to SLIC and MSLIC, IMSLIC has $O(N)$ time complexity, which is independent of the number of superpixels K. Fig. 7d plots the runtime curves for all testing methods, showing that NC is much slower than the other methods. Therefore we exclude NC for further comparison. Figs. 7e and 7f plot the runtime curves for the remaining methods with respect to superpixel number and image size, respectively. We observe that SEEDS, FH, SLIC, SL and MSLIC are the fastest algorithms (MSLIC is 2 times faster than IMSLIC) and IMSLIC is significantly faster than the other methods. For example, IMSLIC runs 10 times faster than SSS for $K \in [400, 700]$, 10 times faster than VCells and 2 times faster than GGM for all K.

Random versus Uniform Initialization in IMSLIC. A good initialization is important for IMSLIC. In Section 5.1, we propose a random strategy that considers content sensitivity in initialization. Fig. 6a illustrates three random initializations and Table 2 shows that random initialization produces consistently better results than those of uniform initialization. We further compare random and uniform

[2] These geodesic distances have already been computed to determine the GVC $\mathcal{V}_g(\Phi(s_i))$ (line 13 in Algorithm 1).

[3] http://47.89.51.189/liuyj/



(a) Under segmentation error (b) Boundary recall (c) Achievable segmentation accuracy (ASA)

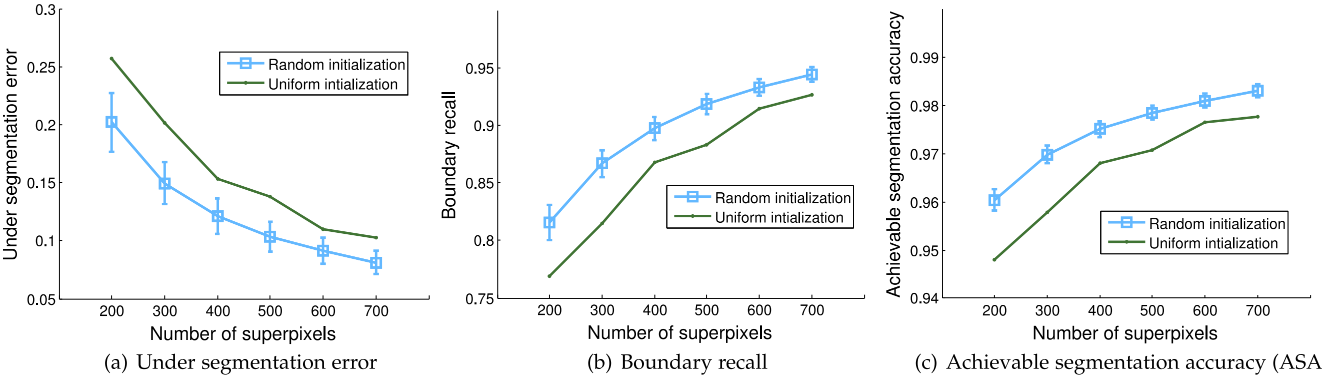
(d) Runtime with respective to K (e) Runtime in (d) without NC (f) Runtime with respective to N

Fig. 7. Evaluation of eleven representative algorithms and our method (IMSLIC) on the BSDS500 benchmark for $K \in [200, 700]$. Superpixels produced by IMSLIC, VCells, and GGM have the lowest under-segmentation error (for $K > 300$), the highest boundary recall ratio (for all K) and the best ASA values (for $K > 300$). SEEDS, FH, SLIC, SL, and MSLIC are the five fastest algorithms and IMSLIC is significantly faster than the other six algorithms. IMSLIC achieves a good balance of under-segmentation error, boundary recall, ASA and run time.

initializations in IMSLIC on the entire BSDS500 dataset. For each image i, we run IMSLIC with 100 random initializations. For each of the three measures (i.e., under segmentation error, boundary recall, and ASA), we compute the metric value on each result and compute the mean E_i and the standard deviation σ_i of the 100 results. Then we compute the mean of E_i, i.e., $E = \frac{1}{500}\sum_{i=1}^{500} E_i$, and the mean of standard deviations $E_\sigma = \frac{1}{500}\sum_{i=1}^{500} \sigma_i$ of 500 images in BSDS500. The results are presented in Fig. 8, which demonstrate that (1) random initialization is consistently better than uniform initialization for $K \in [200, 700]$ on the entire BSDS500 dataset, and (2) the variance from random initializations is small, implying that IMSLIC is numerically stable.

Among the three content-sensitive methods, SSS, MSLIC and IMSLIC, IMSLIC exhibits the best performance despite being 2 times slower than MSLIC. Due to the inferior performance of SSS, we did not compare it further in Sections 6.2 and 6.3.

GGM, VCells and IMSLIC exhibit the best overall performance and achieve a good balance among the metrics USE, BR, ASA and run time. However, VCells is based on an edge-weighted centroidal Voronoi tessellation (EWCVT) [31] that clusters pixels in a colour space and does not consider pixels' connectivity. Therefore, the pixels that have the same label (corresponding to a cluster) can have many disjoint components. Although VCells further applies a Looking-Nearest-Neighbor (LNN) strategy to reduce the probability of segment breaking, the case in which one label has multiple disjoint components still exists. In a postprocess, VCells simply extracts all of the connected components



(a) Under segmentation error (b) Boundary recall (c) Achievable segmentation accuracy (ASA)

Fig. 8. Comparison of random and uniform initializations in IMSLIC on the BSDS500 benchmark for $K \in [200, 700]$. The average results of 100 random initializations are reported and the bars indicate ±1 the mean of standard deviations (see text for details).

TABLE 3
The Statistical Data Pertaining to the Exact Numbers of Superpixels Produced by VCells on the Images in BSDS500

User-specified number of superpixels	Actual numbers of superpixels generated by VCells on BSDS500		
	min num	mean value	max num
200	201	548.824	3,283
300	301	707.190	4,360
400	400	859.724	5,249
500	501	1,003.608	5,974
600	599	1,140.548	6,285
700	696	1,284.734	6,686

in superpixels and relabels them. Therefore, VCells may produce more superpixels than that specified by the user. For example, for the image shown in Fig. 1f, VCells outputs 352 superpixels whereas the user inputs $K = 300$. We summarize the statistical data pertaining to the exact numbers of superpixels produced by VCells on all images in BSDS500 in Table 3. The results show that VCells frequently outputs a much larger number of superpixels. In the application involving image contour closure (Section 6.3), the characteristic of multiple disjoint components in one label makes VCells the worst method for generating superpixels.

GGM [15] uses an objective function that separates spatial compactness and colour homogeneity. Although it exhibits good performance on the metrics USE, BR and ASA, the superpixels produced by GGM have strongly curved boundaries. To demonstrate this attribute, we introduce the compactness metric, which is another important measure of superpixels [32]. The compactness metric is defined as

$$C = \sum_{s \in S} Q_s \frac{|s|}{|I|}, \qquad \text{where} \quad Q_s = \frac{4\pi A_s}{L_s^2}, \tag{13}$$

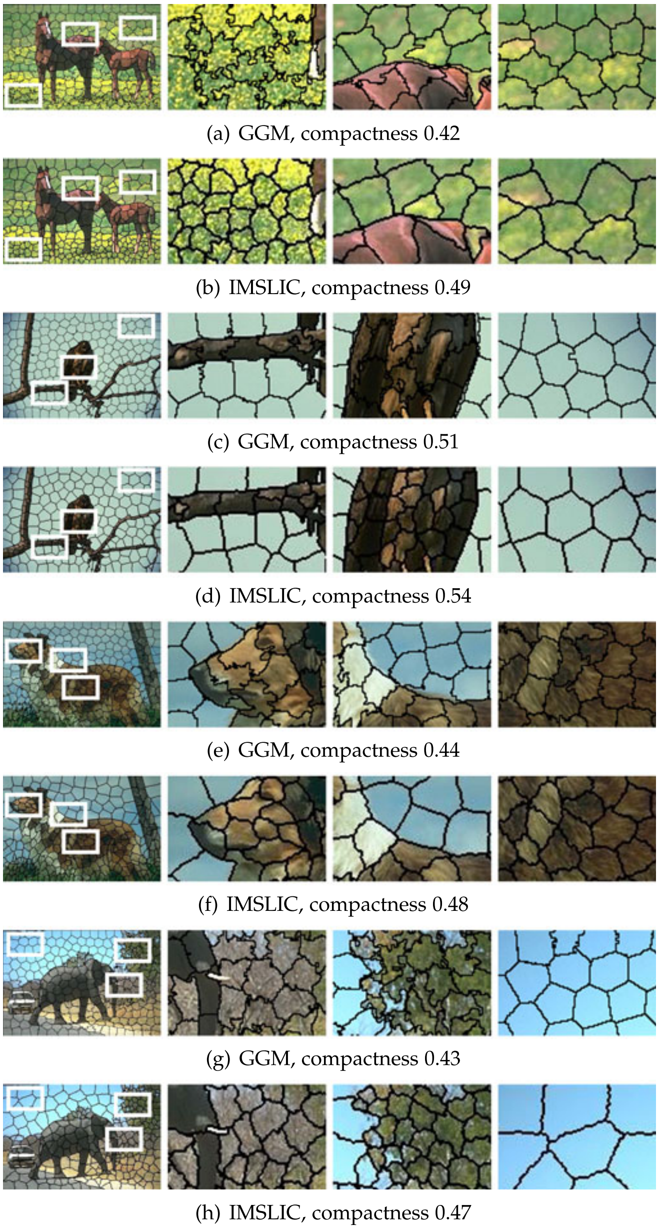
$|s|$ and $|I|$ are the number of pixels in superpixel s and image I respectively, and A_s and L_s are the area and perimeter of s, respectively. $Q_s = 1$ if s is circular and $Q_s(< 1)$ decreases for less compact shapes. $C \leq 1$, and the larger the value of C for a superpixel over-segmentation S is, the more compact S is. Fig. 10a shows the compactness values of four representative methods (GGM, VCells, MSLIC and IMSLIC) on BSDS500. Fig. 10b shows the results on the NYUV2 dataset presented in Section 6.2. The results show that IMSLIC and MSLIC have the best compactness values. Fig. 9 shows quantitative results that compare the compactness of GGM and IMSLIC. These examples demonstrate that

- superpixels from IMSLIC are content-sensitive whereas those from GGM are nearly uniform, and
- superpixels from IMSLIC have a more regular boundary and thus better compactness than those from GGM.

As discussed in Section 6.3, IMSLIC superpixels exhibit better segmentation performance than GGM superpixels, because compact superpixels are more spatially coherent than non-compact ones.

6.2 NYUV2 Results

NYUV2 consists of 1,449 indoor-scene images with a resolution of 640×480. For each colour image, a depth image is also provided. After aligning undistorted depth images with colour images, we further crop the images to a resolution of 625×468 to remove small unlabelled boundary regions.


(a) GGM, compactness 0.42

(b) IMSLIC, compactness 0.49

(c) GGM, compactness 0.51

(d) IMSLIC, compactness 0.54

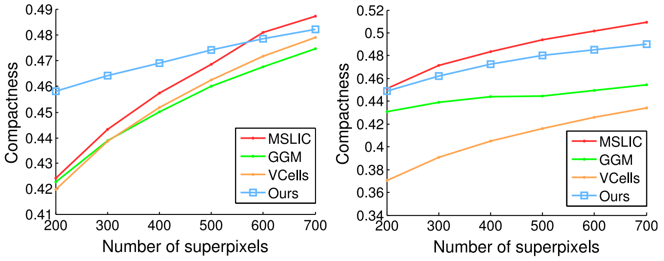
(e) GGM, compactness 0.44

(f) IMSLIC, compactness 0.48

(g) GGM, compactness 0.43

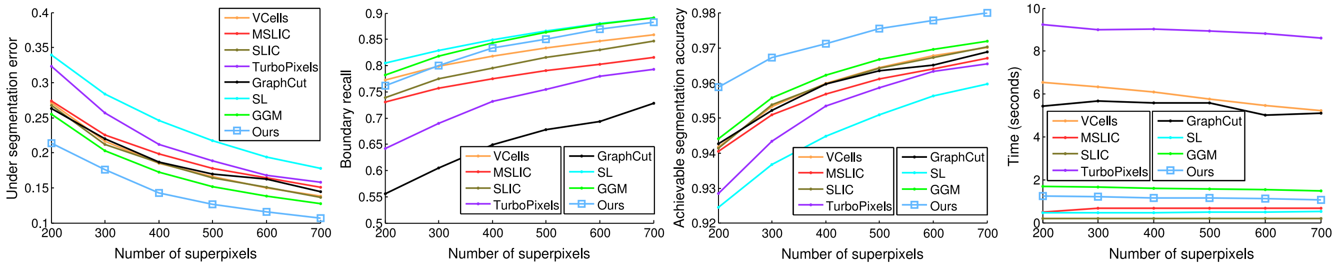
(h) IMSLIC, compactness 0.47

Fig. 9. Quantitative comparison of compactness of GGM and IMSLIC. The results of IMSLIC exhibit content sensitivity and a regular superpixel shape, leading to good compactness. Superpixels in GGM show strongly curved boundaries, leading to bad compactness.



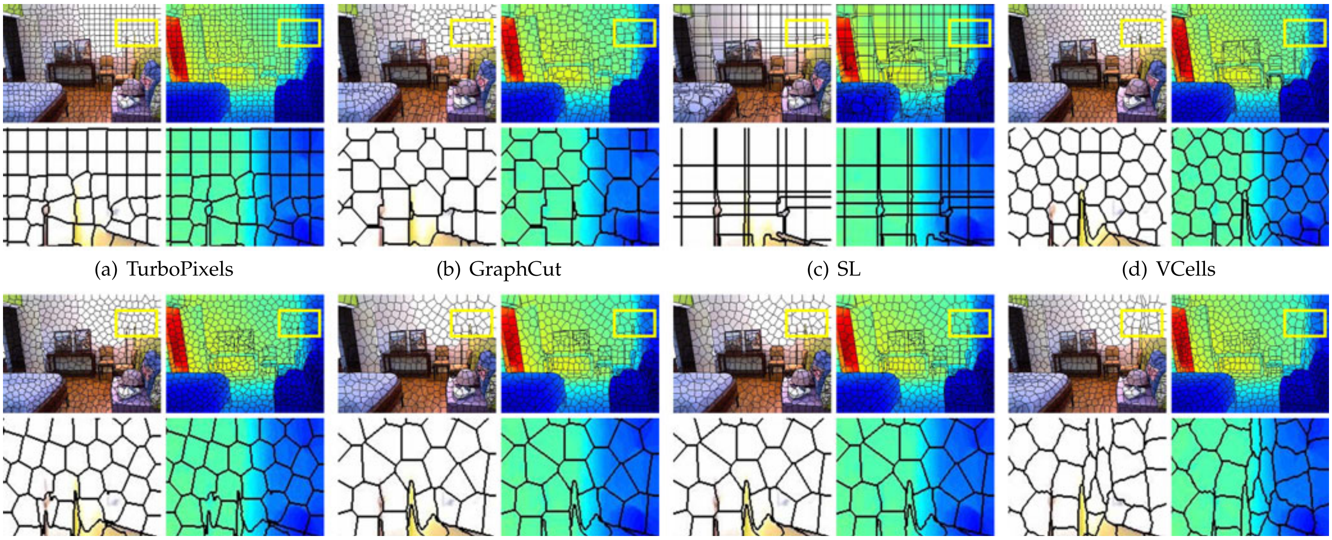
(a) BSDS500 (b) NYUV2

Fig. 10. The compactness metric of four representative methods on the BSDS500 and NYUV2 benchmarks for $K \in [200, 700]$.



(a) Under segmentation error (b) Boundary recall (c) ASA (d) Runtime with respective to K

Fig. 11. Evaluation of seven representative algorithms and our method (IMSLIC) on the NYUV2 benchmark for $K \in [200, 700]$. Superpixels produced by IMSLIC have the least under segmentation error (USE) and the highest achievable segmentation accuracy (ASA) ratio. SL, GGM, and IMSLIC have the highest boundary recall ratio, however, IMSLIC is 1.5 times faster than GGM.



(a) TurboPixels (b) GraphCut (c) SL (d) VCells

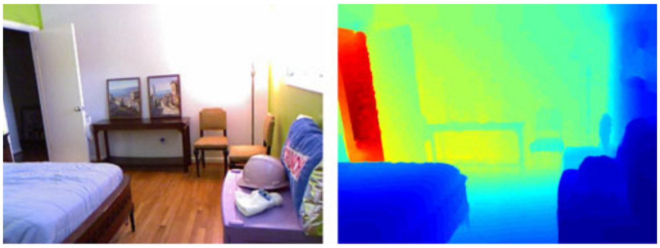
(e) GGM (f) SLIC (g) MSLIC (h) Ours

Fig. 12. Superpixel results of eight representative algorithms on the RGBD image shown in Fig. 13. TurboPixels, GraphCut, SL, VCells and GGM do not use the depth information, and therefore, their superpixels cannot capture the depth change in the upper-right corner (outlined by yellow rectangle). SLIC, MSLIC, and IMSLIC can use the depth information. However, due to the Euclidean metric used in SLIC, and MSLIC, their superpixels still cannot capture the sharp depth change. Only our method, IMSLIC, can successfully generate content-and-depth-sensitive superpixels in the upper-right corner.

By mapping an RGBD image to a 2-manifold in $\mathbb{R}^6 = (\lambda_2 l, \lambda_2 a, \lambda_2 b, \lambda_1 d, \lambda_1 u, \lambda_1 v)$, where (l, a, b) is the CIELAB colour, d is the depth, (u, v) is the pixel location, and λ_1 and λ_2 are the factors defined in Eqn. (4), SLIC, MSLIC and IMSLIC are naturally extended to include the depth dimension in the metric (1) [12]. Although segmentation algorithms that are specially designed for RGBD images exist (e.g., [33]), SLIC, MSLIC, and IMSLIC are general for both RGB and RGBD images.

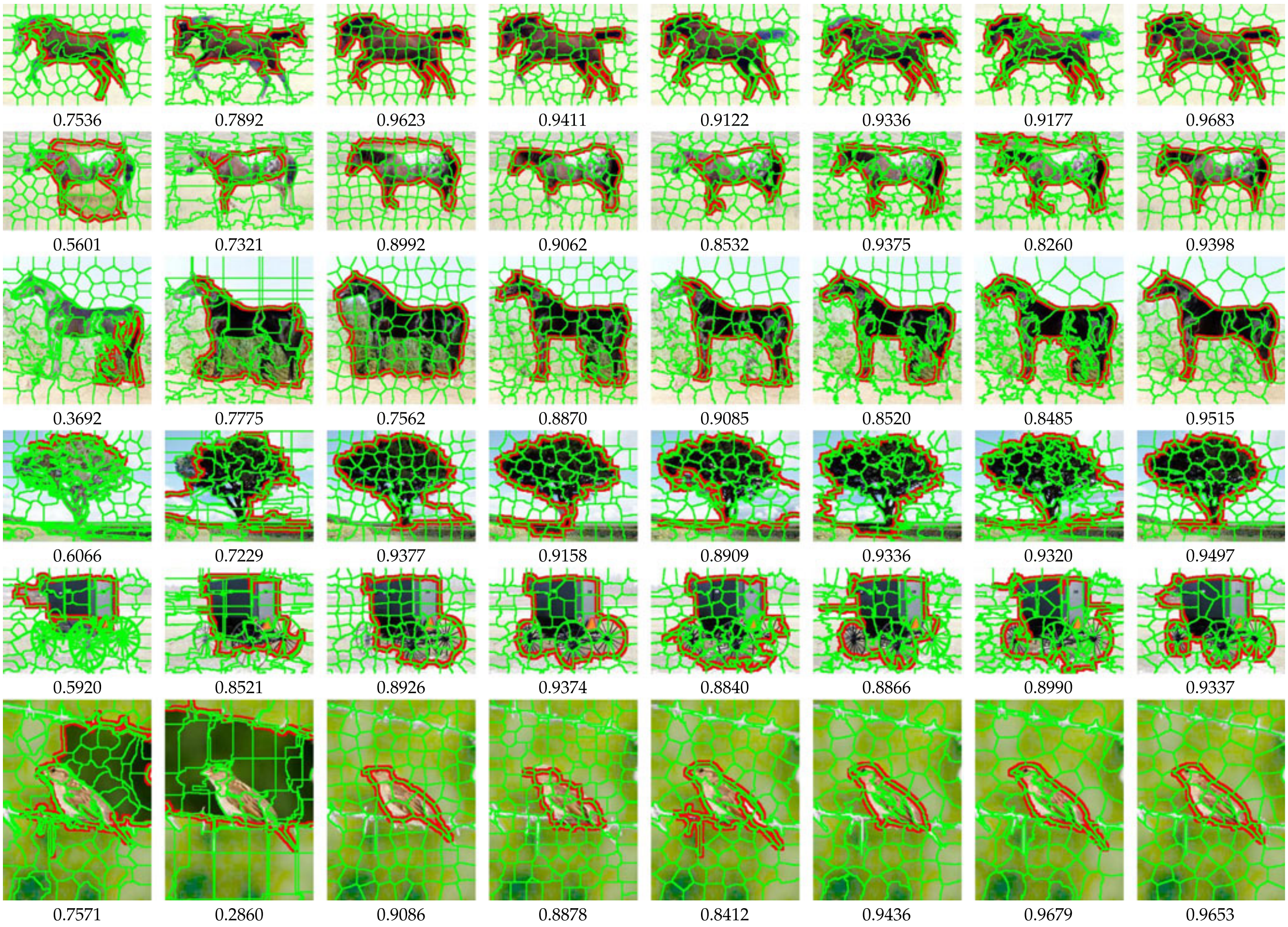
Eight superpixel methods, including three (SLIC, MSLIC and IMSLIC) that can incorporate the depth information and five (VCells, TurboPixels, GraphCut, SL and GGM) that only operate with colour images, are compared on the NYUV2 dataset. As shown in Figs. 11a and 11c, IMSLIC exhibits the best performance on the USE and ASA metrics. Fig. 11b shows that GGM, SL and IMSLIC exhibit the best performance on the BR metric. Fig. 11d shows that SLIC, SL, MSLIC and IMSLIC are the four fastest methods. MSLIC is 2 times faster than IMSLIC, whereas IMSLIC is 1.5, 5, 5.5 and 8 times faster than GGM, GraphCut, VCells and TurboPixels, respectively. Note that our method shows consistently better compactness than GGM in NYUV2 (Fig. 10b). The consistently good performance of IMSLIC on BSDS500 and NYUV2 shows that it has good scalability on different datasets.

Fig. 13 shows one example of an RGBD image in the NYUV2 dataset. Note that in this example, the upper-right corner outlined by a yellow rectangle in Fig. 12 is nearly uniform in colours (Fig. 13a), but there is a sharp change in depth values (Fig. 13b). The superpixels produced by those methods based on only colour information cannot reflect this sharp depth change (Figs. 12a, 12b, 12c, 12d, and 12e). Due to the Euclidean metric, SLIC and MSLIC still cannot reflect this sharp depth change in superpixels (Figs. 12f, and 12g). Taking advantage of intrinsic geodesic distance, only



(a) Color image (b) Depth image

Fig. 13. An RGBD image in NYUV2. In the upper-right corner of the scene, the colour is almost uniformly white; however, there is a sharp change in depth. Superpixels generated by eight representative methods on this image are illustrated in Fig. 12.



(a) VCells (b) SL (c) TurboPixels (d) GraphCut (e) GGM (f) SLIC (g) MSLIC (h) Ours

Fig. 14. Contour closure results on three examples in WHD (first three rows) and three examples in WSD (last three rows) using the superpixels generated by eight algorithms. The optimal closure contours are shown in red, and the boundaries of superpixels are shown in green. The F measure value for each closure contour is shown below each image; the range of the F-measure values is $[0, 1]$, and larger values indicate better results.

our method IMSLIC can capture this sharp depth change (Fig. 12h).

Comparison of MSLIC and IMSLIC. Both MSLIC and IMSLIC generate uniform tessellations on a 2-manifold $\mathcal{M}$ and output content-sensitive superpixels. By using geodesic metric, each Voronoi cell generated by IMSLIC is guaranteed to be a singly connected region, and thus, IMSLIC guarantees to output the exact number of superpixels designated by the user. By comparison, due to the Euclidean metric, one Voronoi cell generated by MSLIC may consist of several disjoined components, and thus, a post-process of split-and-merge has to be applied with some additional heuristic parameters. Because IMSLIC does not require any heuristic-based postprocessing, it is easy to use and shows consistent performance on different image datasets.

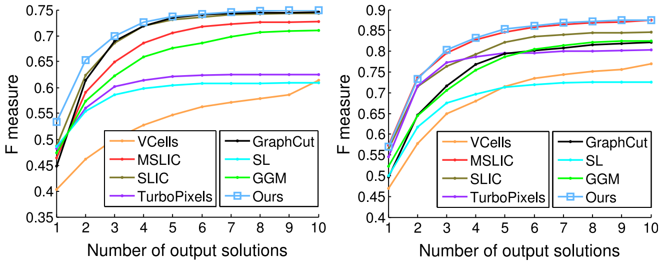
6.3 Application to Image Contour Closure

Superpixels are an intermediate structure, designed to reduce the complexity of subsequent image processing tasks such as segmentation. We evaluate the performance of various superpixels in an application of foreground segmentation [16] in which contour closure is detected for separating a figure from the background. Observing that searching the entire image space of all pixels is intractable; Levinshtein et al. [16] proposed a novel framework that transforms the problem of finding cycles of contour fragments into that of finding subsets of superpixels such that the boundary of union of these superpixels has strong edge support in the image.

We use the source code of the framework developed by Levinshtein et al. provided on the authors' website to compare eight representative superpixel methods. Two datasets with ground-truth segmentations, the Weizmann Horse Database (WHD) [34] and the Weizmann Segmentation Database (WSD) [35], were used to conduct a quantitative assessment. The average F-measure values for these two datasets are summarized in Fig. 15, in which values vary over the range $[0, 1]$ and larger values indicate better results. We also illustrate six qualitative results (three in WHD and three in WSD) in Fig. 14. Both the quantitative and qualitative results show that IMSLIC exhibits the best foreground segmentation performance.

7 Conclusion

In this paper, we proposed the intrinsic MSLIC (IMSLIC), which computes content-sensitive superpixels by constructing an intrinsic GCVT on an image manifold. To efficiently compute the GCVT at a low cost, we proposed a simple and fast approximation to a closed-form solution. Compared



(a) WHD (b) WSD

Fig. 15. F-measure values with respect to the number of output solutions (from 1 to 10) in the framework developed by Levinshtein et al. [16] on the WHD (a) and WSD (b) datasets. The number of superpixels is fixed to 100, and F-measure values vary over the range $[0, 1]$; larger values indicate better results.

with existing methods, IMSLIC has the following merits: (1) IMSLIC better characterizes image edges by applying an intrinsic geodesic metric, (2) IMSLIC has fewer parameters and is easy to use on different image datasets, and (3) IMSLIC can output the exact number of superpixels designated by the user. We compared IMSLIC with eleven representative methods on the BSDS500 dataset and seven representative methods on the NYUV2 dataset. Both quantitative and qualitative results showed that IMSLIC outperforms other methods by achieving a good balance among commonly used quality measures including compactness, adherence to boundaries, achievable segmentation accuracy and run time. We also studied different superpixels in an application of image contour closure, and the results obtained for two datasets WHD and WSD demonstrated that IMSLIC achieves the best foreground segmentation performance among eight superpixel methods.

IMSLIC can be extended to video superpixels (a.k.a. supervoxels) by considering spatio-temporal coherence in video frames. In particular, a propagation scheme was proposed in [14] that extends structure-sensitive superpxels to video superpixels. Following this scheme, we can compute the superpixels using IMSLIC with random initialization in the first frame. Instead of re-initializations at the subsequent frames, we can apply an optical or a SIFT flow to find pixel correspondences between adjacent frames. Then the centers of superpixels (i.e., the generators of Voronoi cells) in a previous frame can be used as initial seeds in the next frame. With these initial seeds, IMSLIC can converge fast to obtain good temporally consistent superpixels in later frames. We will study this extension with detailed performance analysis in a future work.

ACKNOWLEDGMENTS

The author thanks the editor and reviewers for their comments that help improve this paper. This work was supported by the National Key Research and Development Plan (2016YFB1001202), Royal Society-Newton Advanced Fellowship and the Natural Science Foundation of China (61521002, 61432003, 61661130156).

REFERENCES

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

[2] A. Levinshtein, C. Sminchisescu, and S. Dickinson, "Optimal contour closure by superpixel grouping," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 480–493.

[3] B. Fulkerson, A. Vedaldi, and S. Soatto, "Class segmentation and object localization with superpixel neighborhoods," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009, pp. 670–677.

[4] S. Wang, H. Lu, F. Yang, and M.-H. Yang, "Superpixel tracking," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 1323–1330.

[5] B. Mičušík and J. Kočecká, "Multi-view superpixel stereo in urban environments," *Int. J. Comput. Vis.*, vol. 89, no. 1, pp. 106–119, 2010.

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

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

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

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

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

[11] J. Wang and X. Wang, "VCells: Simple and efficient superpixels using edge-weighted centroidal Voronoi tessellations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 6, pp. 1241–1247, Jun. 2012.

[12] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2074–2282, Nov. 2012.

[13] Y.-J. Liu, C. Yu, M. Yu, and Y. He, "Manifold SLIC: A fast method to compute content-sensitive superpixels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 651–659.

[14] P. Wang, G. Zeng, R. Gan, J. Wang, and H. Zha, "Structure-sensitive superpixels via geodesic distance," *Int. J. Comput. Vis.*, vol. 103, no. 1, pp. 1–21, 2013.

[15] Y. Cai and X. Guo, "Anisotropic superpixel generation based on mahalanobis distance," *Comput. Graph. Forum*, vol. 35, no. 7, pp. 199–207, 2016.

[16] A. Levinshtein, C. Sminchisescu, and S. Dickinson, "Optimal image and video closure by superpixel grouping," *Int. J. Comput. Vis.*, vol. 100, no. 1, pp. 99–119, 2012.

[17] Q. Du, V. Faber, and M. Gunzburger, "Centroidal Voronoi tessellations: Applications and algorithms," *SIAM Rev.*, vol. 41, no. 4, 1999, Art. no. 637676.

[18] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.

[19] R. Horst, P. M. Pardalos, and N. V. Thoai, *Introduction to Global Optimization*, 2nd edition. Dordrecht, The Netherlands: Kluwer Academic Publishers, 2000.

[20] Q. Du, M. D. Gunzburger, and L. Ju, "Constrained centroidal Voronoi tessellations for surfaces," *SIAM J. Scientific Comput.*, vol. 24, no. 5, pp. 1488–1506, 2003.

[21] Y.-J. Liu, "Semi-continuity of skeletons in 2-manifold and discrete Voronoi approximation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1938–1944, Sep. 2015.

[22] Y.-J. Liu, Z.-Q. Chen, and K. Tang, "Construction of iso-contours, bisectors and Voronoi diagrams on triangulated surfaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1502–1517, Aug. 2011.

[23] Y.-J. Liu, C.-X. Xu, R. Yi, D. Fan, and Y. He, "Manifold differential evolution (MDE): A global optimization method for geodesic centroidal Voronoi tessellations on meshes," *ACM Trans. Graph.*, vol. 35, no. 6, pp. 243:1–243:10, 2016.

[24] D. P. Bertsekas, "A simple and fast label correcting algorithm for shortest paths," *Netw.*, vol. 23, no. 8, pp. 703–709, 1993.

[25] D. P. Bertsekas, F. Guerriero, and R. Musmanno, "Parallel asynchronous label-correcting methods for shortest paths," *J. Optimization Theory Appl.*, vol. 88, no. 2, pp. 297–320, 1996.

[26] D. P. Bertsekas, *Network Optimization: Continuous and Discrete models*. Belmont, MA, USA: Athena Scientific Belmont, 1998.

[27] V. de Silva and J. B. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction," in *Proc. Advances Neural Inf. Process. Syst.*, 2002, pp. 705–712.

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

[29] D. 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 *Proc. 8th Int. Conf. Comput. Vis.*, July 2001, pp. 416–423.

[30] N. Silberman, P. Kohli, D. Hoiem, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 746–760.

[31] J. Wang, L. Ju, and X. Wang, "Image segmentation using local variation and edge-weighted centroidal Voronoi tessellations," *IEEE Trans. Image Process.*, vol. 20, no. 11, pp. 3242–3256, Nov. 2011.

[32] A. Schick, M. Fischer, and R. Stiefelhagen, "An evaluation of the compactness of superpixels," *Pattern Recog. Lett.*, vol. 43, no. 1, pp. 71–80, 2014.

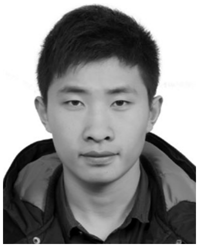
[33] D. Weikersdorfer, D. Gossow, and M. Beetz, "Depth-adaptive superpixels," in *Proc. Int. Conf. Pattern Recog.*, 2012, pp. 2087–2090.

[34] E. Borenstein and S. Ullman, "Class-specific, top-down segmentation," in *Proc. 7th Eur. Conf. Comput. Vis.*, 2002, pp. 109–124.

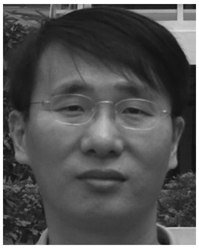
[35] S. Alpert, M. Galun, R. Basri, and A. Brandt, "Image segmentation by probabilistic bottom-up aggregation and cue integration," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog.*, 2007, pp. 1–8.

Yong-Jin Liu received the BEng degree from Tianjin University, China, in 1998, and the PhD degree from the Hong Kong University of Science and Technology, Hong Kong, China, in 2004. He is currently an associate professor with the TNList, Department of Computer Science and Technology, Tsinghua University, China. His research interests include computational geometry, computer graphics and computer-aided design. He is a senior member of the IEEE and a member of the ACM.

Minjing Yu received the BEng degree from Wuhan University, China, in 2014. She is currently working toward the PhD degree at TNList, Department of Computer Science and Technology, Tsinghua University, China. Her research interests include computer vision, computer graphics and cognitive computation.



Bing-Jun Li received the BEng degree from Beijing University of Posts and Telecommunications, China, in 2015. Currently, he is working toward the master degree at the Department of Computer Science and Technology, Tsinghua University, China. His research interests include computer vision and image processing.



Ying He received the BS and MS degrees in electrical engineering from Tsinghua University and the PhD degree in computer science from Stony Brook University. He is an associate professor with the School of Computer Engineering, Nanyang Technological University, Singapore. He is interested in the problems that require geometric computing and analysis. For more information, visit http://www.ntu.edu.sg/home/yhe