Manifold SLIC: A Fast Method to Compute Content-Sensitive Superpixels

Yong-Jin Liu, Cheng-Chi Yu*, Min-Jing Yu*
Tsinghua University, China
{liuyongjin, ycc13, yumj14}@tsinghua.edu.cn

Ying He
Nanyang Technological University, Singapore
YHe@ntu.edu.sg

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 (e.g., with high intensity or color variation) and large superpixels in content-sparse regions. Rather than the conventional SLIC method that clusters pixels in \( \mathbb{R}^5 \), we map the 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 \( \mathcal{M} \). We propose an efficient method to compute restricted centroidal Voronoi tessellation (RCVT) — a uniform tessellation — on \( \mathcal{M} \), which induces the content-sensitive superpixels in \( I \). Unlike other algorithms that characterize content-sensitivity by geodesic distances, manifold SLIC tackles the problem by measuring areas of Voronoi cells on \( \mathcal{M} \), which can be computed at a very low cost. As a result, it runs 10 times faster than the state-of-the-art content-sensitive superpixels algorithms. We evaluate manifold SLIC and seven representative methods on the BSDS500 benchmark and observe that our method outperforms the existing methods.

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 \( 10 \), contour closure \( 8 \), 2.1D sketches \( 21 \), object location \( 6 \), object tracking \( 19 \), stereo 3D reconstruction \( 14 \), and many others.

There are two major classes of algorithms to compute superpixels, namely, the graph-based methods and the clustering-based methods. 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) \( 16 \), the Felzenszwalb-Huttenlocher (FH) method \( 5 \), superpixel lattices (SL) \( 13 \), and the graph-cut-based energy optimization method (GraphCut) \( 17 \). NC generates regular and compact superpixels, but does not adhere to image boundary and has a high computational cost — \( O(N^{1.5}) \) time complexity for an \( N \)-pixel image as observed in \( 9 \). FH preserves 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 superpixel algorithms. However, their produced superpixels conform to grids rather than adhering to image boundaries. GraphCut is elegant and theoretically sound, but it is difficult to use due to many parameters involved in the algorithm.

The clustering-based algorithms group pixels into clusters (i.e., superpixels) and iteratively refine them until some convergence criteria are satisfied. Popular clustering methods are Turbopixels \( 9 \), simple linear iterative clustering (SLIC) \( 1 \) and structure-sensitive superpixels (SSS) \( 18 \). All three 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 \( 9 \) generates a geometric flow for each seed and propagates them using the level set method. The superpixel boundary is the points where two flows meet. Although Turbopixels has a theoretical linear time complexity \( O(N) \), computational results show that it runs slowly on real-world datasets \( 1 \). SLIC \( 1 \) is an adaption of \( K \)-means that clusters pixels in a 5-dimensional Euclidean space combining colors and images. It assigns each pixel to a cluster of the nearest seed and iteratively updates the cluster center using centroidal Voronoi tessellation (CVT). SLIC is conceptually simple, easy to implement, and highly efficient in practice.

To compute content-sensitive superpixels, Wang et al. \( 18 \) adopted the geometric flow method \( 9 \) into a CVT optimization framework. A key difference between SSS and SLIC is that SLIC measures the distance between clusters

\( 1 \)Since in computer vision, “structure” is usually referred to as some high-order structures in the image, in this paper we use the word “content”.

* C. Yu and M. Yu contributed equally to this paper
using Euclidean distance whereas SSS takes geodesic distances into account. As a result, SLIC produces uniform superpixels everywhere, whereas SSS can effectively capture the non-homogenous feature in images, i.e., small superpixels in content-dense regions (e.g., with high intensity or color variation) and large superpixels in content-sparse regions. However, SSS is computationally expensive.

Inspired by the simplicity and high performance of SLIC \cite{Achanta2009SLIC} and the content-aware nature of SSS \cite{Roth2007}, we propose manifold SLIC, an extension of SLIC for fast computation of content-sensitive superpixels. Our key idea is to represent the image $I$ as a 2-manifold $\mathcal{M}$ embedded in the combined color and image space $\mathbb{R}^5$, on which the area elements are a good measure of content density in $I$. We develop an efficient algorithm to compute restricted CVT — a uniform tessellation — on $\mathcal{M}$, which induces the content-sensitive superpixels in $I$. Unlike other algorithms that characterize content-sensitivity by geodesic distances, manifold SLIC addresses the problem by measuring areas of Voronoi cells on $\mathcal{M}$, which can be computed at a very low cost. As a result, it runs 10 times faster than the state-of-the-art content-sensitive superpixels algorithm \cite{Roth2007}. We evaluate manifold SLIC and seven representative methods on the BSDS500 benchmark and observe that our method outperforms the existing methods. See Section 6 for detailed comparison and discussion. Images are provided in high resolution for zoom-in examination.

2. Preliminaries

Since manifold SLIC is based on SLIC \cite{Achanta2009SLIC} and is also closely related to SSS \cite{Roth2007}, we briefly introduce both algorithms before presenting it.

2.1. SLIC

Let us denote by $I$ the input image with $N$ pixels. For a pixel $p \in I$, SLIC represents its color $c(p)$ in the CIELAB color space, i.e., $c(p) = (l(p), a(p), b(p))$. Given two pixels $p_1 = (u_1, v_1)$ and $p_2 = (u_2, v_2)$, SLIC measures the distance between them using a normalized Euclidean distance in the combined color and image space $\mathbb{R}^5$:

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

where $N_s$ and $N_c$ are two constants, and

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

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

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 the so-called centroidal Voronoi tessellation.

Mathematically speaking, a CVT is defined as follows: Let $\rho : I \rightarrow \mathbb{R}^+$ be a density function defined on $I$. For a Voronoi cell $\mathcal{V}(s_i)$, its center of mass $c_i$ is

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

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., $c_i = s_i$, $1 \leq i \leq K$. Du et al. \cite{Du2006} showed that a CVT is the minimizer of the following energy $E$

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

where $d(x, y)$ measures the distance between $x$ and $y$. They also showed that the solution always exists.

SLIC sets the distance $d(x, s_i) = D(x, s_i)$, as defined in Eqn. (1), and computes CVT using the Lloyd method, which iteratively moves each seed to the corresponding center of mass. We outline SLIC in Algorithm 1. 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 \( s_i \) (lines 6-13) 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 (line 16). Computational results show that SLIC converges in only 10 iterations for real-world images [11]. As a result, SLIC has an \( O(N) \) time complexity, which is independent of the number of superpixels \( K \). In contrast, the conventional \( K \)-means method runs in \( O(KN\text{iter}) \), where \( \text{iter} \) is the number of iterations.

**Algorithm 1 SLIC [11]**

**Input:** An image \( I \) of \( N \) pixels, the desired number of superpixels \( K \), the maximal number of iterations \( \text{iter}_{\text{max}} \) and the convergence threshold \( \varepsilon \).

**Output:** \( K \) superpixels of similar sizes.

1. Initialize seeds \( (s_i)_{i=1}^K \) at regular grids with step length \( S = \sqrt{N/K} \).
2. Move each seed to the lowest gradient position in a \( 3 \times 3 \) neighborhood.
3. Initialize label \( l(p) = -1 \) and distance \( d(p) = \infty \) for each pixel \( p \).
4. Initialize the residual error \( \text{err} = \infty \) and \( \text{iter} = 0 \).
5. **while** \( \text{err} > \varepsilon \) and \( \text{iter} \leq \text{iter}_{\text{max}} \) **do**
6. **for** each seed \( s_i \) **do**
7. **for** each pixel \( p \) in a \( 2S \times 2S \) region around \( s_i \) **do**
8. Compute the distance \( D = D(s_i, p) \) between \( s_i \) and \( p \).
9. **if** \( D < d(p) \) **then**
10. \( d(p) = D; l(p) = i \).
11. **end if**
12. **end for**
14. **end for**
15. **for** each seed \( s_i \) **do**
16. Compute average \( c_i \) of pixels in the cluster of \( s_i \).
17. \( \text{err} += \| c_i - s_i \|^2 \).
18. \( s_i = c_i \).
19. **end for**
20. \( \text{iter} +++. \)
21. **end while**

2.2. SSS

SLIC is conceptually simple and highly efficient. However, due to the constant CVT density function \( \rho(x) \equiv 1, \forall x \in I \), it produces uniform superpixels, i.e., superpixels have similar sizes everywhere. Since many real-world images may contain complicated contents, it is highly desired to develop content-sensitive superpixels, i.e., small superpixels in content-dense regions and large superpixels in content-sparse regions. Wang et al. [12] computed content-sensitive superpixels using a geometric flow [3] into the CVT optimization framework. In contrast to SLIC that uses Euclidean distances, SSS adopts geodesic distances defined as follows:

\[
d(p_1, p_2) = \min_{\gamma_{p_1, p_2}} \int_0^1 D(P_{p_1, p_2}(t))\|\dot{P}_{p_1, p_2}(t)\|dt
\]

where \( P_{p_1, p_2}(t) \) is a parameterized path connecting pixels \( p_1 \) (\( t = 0 \)) and \( p_2 \) (\( t = 1 \)). The weight function \( D(z) \) penalizes image boundaries [2].

\[
D(z) = e^{\frac{\text{edge}(z)}{v}}, \text{ edge}(z) = \frac{\|\nabla I\|}{G_\sigma * \|\nabla I\| + \gamma}
\]

where \( v \) is a scaling parameter, \( \text{edge}(z) \) is a normalized edge measurement, \( G_\sigma \) is the Gaussian functions with the standard deviation \( \sigma \), and \( \gamma \) is a small constant to alleviate the affect of weak intensity boundary.

Compared with Euclidean distances in Eqn. (1), geodesic distances in Eqn. (4) penalize image boundary, hereby are an effective measure of content-sensitivity. Computational results show that Wang et al.’s method [12] can compute high-quality content-sensitive superpixels that exhibit high boundary adherence. However, since it is computationally expensive to measure geodesic distances, their method is among the slowest algorithms in our experiments.

3. Overview

Manifold SLIC extends SLIC to compute content-sensitive superpixels, meanwhile it inherits all the favorable features of SLIC, such as simplicity and high performance.

Similar to SLIC [11], we represent the color in the CIELAB color space and denote by \( \mathbf{c}(u, v) = (l(u, v), a(u, v), b(u, v)) \) the color at the pixel \((u, v)\) in the image \( I \). Then we define a stretching map \( \Phi : I \rightarrow \mathbb{R}^5 \) to send pixels to a 2-manifold \( \mathcal{M} \) embedded in the 5-dimensional combined image and color space,

\[
\Phi(u, v) = (\lambda_1 u, \lambda_2 c, \lambda_1 l, \lambda_1 a, \lambda_2 b),
\]

where \( \lambda_1 \) and \( \lambda_2 \) are global stretching factors. We set \( \lambda_1 = \frac{1}{\sqrt{N}} \) and \( \lambda_2 = \frac{1}{N} \) and adopt the SLIC metric \( D \) in Eqn. (1) to measure the distance between points on manifold \( \mathcal{M} \), i.e.,

\[
D(\Phi(p_1), \Phi(p_2)) = \sqrt{\lambda_1^2 d_1^2 + \lambda_2^2 d_2^2 + d_3^2}.
\]

For a pixel \( p = (u, v) \), we denote by \( \Box_p \) the unit square \( 1 \times 1 \) centered at \( p \). Let \( p_1, p_2, p_3, p_4 \) be the four corners of
Figure 2. Measuring area on the 2-manifold \( \mathcal{M} \) embedded in \( \mathbb{R}^2 \).

Figure 3. Overview of manifold SLIC on a synthetic greyscale image \( I \). 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 the content density in \( I \) (see (a)). Initially, we distribute \( K (= 16) \) seeds \( \{s_i\}_{i=1}^K \) uniformly in \( I \), leading to a Voronoi diagram on \( \mathcal{M} \) (see (b)). We then improve the Voronoi diagram by computing restricted centroidal Voronoi tessellation (RCVT) \( \{\mathcal{V}(\Phi(s_i))\}_{i=1}^K \) on \( \mathcal{M} \) — a uniform tessellation on \( \mathcal{M} \) where the Voronoi cells are of similar sizes. Finally, the RCVT induces the content-sensitive superpixels \( \{\Phi^{-1}(\mathcal{V}(\Phi(s_i)))\}_{i=1}^K \) in the image domain (see (c)).

Consider \( \square_p \), each of which is determined by the average of the four neighboring pixels. The square \( \square_p \) consists of two triangles, i.e., \( \square_p = \triangle p_1 p_2 p_3 \cup \triangle p_3 p_4 p_1 \). Then we approximate the area of the curved triangle \( \Phi(\triangle p_1 p_2 p_3) \) by the area of the planar triangle \( \triangle \Phi(p_1) \Phi(p_2) \Phi(p_3) \),

\[
\text{Area}(\Phi(\triangle p_1 p_2 p_3)) \approx \frac{1}{2} \|\Phi(p_2) - \Phi(p_1)\| \|\Phi(p_2) - \Phi(p_3)\| \sin \theta,
\]

where \( \theta \) is the angle between vectors \( \Phi(p_2) - \Phi(p_1) \) and \( \Phi(p_2) - \Phi(p_3) \). Note that the length \( \|\Phi(x) - \Phi(y)\| \) is measured using the metric \( D \) in Eqn. (7). 

Algorithm 2 Manifold SLIC

**Input:** An image \( I \) of \( N \) pixels, the desired number of superpixels \( K \), the maximal number of iterations \( \text{iter}_{\text{max}} \) and the convergence threshold \( \varepsilon \).

**Output:** \( K \) content-sensitive superpixels.

1. Initialize seeds \( \{s_i\}_{i=1}^K \) at regular grids with step length \( S = \sqrt{N/K} \).
2. Move each seed to the lowest gradient position in a \( 3 \times 3 \) neighborhood.
3. Initialize label \( l(p) = -1 \) and distance \( d(p) = \infty \) for each pixel \( p \).
4. Initialize the residual error \( \text{err} = \infty \) and \( \text{iter} = 0 \).
5. Compute \( \text{Area}(\Phi(\square_p)) \) for each pixel \( p \in I \).
6. Compute a local search range \( \Xi = \frac{4\sum_{i=1}^K \text{Area}(\Phi(\square_{p_i}))}{K} \).
7. while \( \text{err} > \varepsilon \) and \( \text{iter} \leq \text{iter}_{\text{max}} \) do
   8. for each seed \( s_i \) do
      9. Compute \( \text{Area}(\Phi(\Omega(s_i))) \) of a \( 2S \times 2S \) region \( \Omega(s_i) \) centered at \( s_i \).
   10. Compute a scaling factor \( \lambda_i = \sqrt{\frac{\Xi}{\text{Area}(\Phi(\Omega(s_i)))}} \).
   11. if \( \text{iter} > 0 \) and \( \lambda_i < \tau \) and \( \text{Area}(\mathcal{V}_M(\Phi(s_i))) > \Xi/4 \) then
      12. Split Voronoi cell \( \mathcal{V}_M(\Phi(s_i)) \). (Section 5.1)
   13. end if
   14. end for
   15. Merge the Voronoi cells whose areas are small and whose seeds are close to each other. (Section 5.2)
   16. for each seed \( s_i \) do
      17. for each pixel \( p \) in a \( 2\lambda_i S \times 2\lambda_i S \) region centered at \( s_i \) do
         18. Compute the distance \( D = D(\Phi(s_i), \Phi(p)) \) using Eqn. (7).
         19. if \( D < d(p) \) then
            20. \( d(p) = D; l(p) = i \).
         end if
      end for
   end for
   21. end if
   22. end for
   23. end for
   24. err = 0.
   25. for each seed \( s_i \) do
      26. Compute \( \text{Area}(\mathcal{V}_M(\Phi(s_i))) \).
      27. Compute the mass center \( m_i \) of \( \mathcal{V}_M(\Phi(s_i)) \) using Eqn. (11) and set \( c_i = P(m_i) \) using Eqn. (19).
      28. \( s_i = c_i \).
   end for
   29. \text{iter} ++.
30. end while


Figure 4. Towards this goal, we propose two simple yet effective techniques: computing restricted CVT (Section 4) and locally updating bad-shaped Voronoi cells (Section 5). We also outline the manifold SLIC in Algorithm 2 and highlight its differences to SLIC in blue.
4. Computing Restricted CVT

The quality of CVT plays a critical role in the content-sensitive superpixels. It is expensive to compute exact CVTs on curved manifolds \[11, 12, 20\], we compute the restricted centroidal Voronoi tessellation (RCVT), which approximates the exact CVT well for \(K\) in a proper range, e.g., \(K > 200\) in images of resolution \(512 \times 512\).

4.1. Restricted Voronoi Diagram and RCVT

Let \(S = \{s_i \mid s_i \in I, 1 \leq i \leq K\}\) be the set of seeds in the image \(I\) and \(G = \{\Phi(s_i)\}_{i=1}^K\) the corresponding generators on the manifold \(\mathcal{M}\). The Euclidean Voronoi cell of a generator \(\Phi(s_i)\), denoted by \(\mathcal{V}_G\), is

\[
\mathcal{V}_G(\Phi(s_i)) = \{x \in \mathbb{R}^n \mid ||x - \Phi(s_i)||_2 \leq ||x - \Phi(s_j)||_2, \forall j \neq i, \Phi(s_j) \in G\}.
\]

Restricting \(\mathcal{V}_G\) to the manifold \(\mathcal{M}\), we obtain the restricted Voronoi cell \(\mathcal{V}_M(\Phi(s_i)) \subseteq \mathcal{M} \cap \mathcal{V}_G(\Phi(s_i))\). The restricted Voronoi diagram \(RV\mathcal{D}(\mathcal{M}, \mathcal{M})\) is the collection of restricted Voronoi cells satisfying

\[
RV\mathcal{D}(\mathcal{M}, \mathcal{M}) = \{\mathcal{V}_M(\Phi(s_i)) \neq \emptyset | \mathcal{V}_G(\Phi(s_i)) \in \mathcal{G}\} \tag{10}
\]

Edelsbrunner and Shah \[4\] showed that \(RV\mathcal{D}(\mathcal{M}, \mathcal{M})\) is a finite closed covering of \(\mathcal{M}\). For a restricted Voronoi cell \(\mathcal{V}_M(\Phi(s_i))\), its mass centroid \(m_i\) is

\[
m_i = \frac{\int_{r \in \mathcal{V}_M(\Phi(s_i))} rdr}{\int_{r \in \mathcal{V}_M(\Phi(s_i))} dr} \tag{11}
\]

Then we define restricted CVT as follows:

**Definition 1.** Let \(X = \{x_i \mid x_i \in \mathbb{R}^n, 1 \leq i \leq K\}\) be a set of points in \(\mathbb{R}^n\) and \(\mathcal{M}\) be a 2-manifold embedded in \(\mathbb{R}^n\). A restricted Voronoi diagram \(RV\mathcal{D}(X, \mathcal{M})\) is a restricted CVT (or RCVT) if each generator \(x_i \in X\) is the mass centroid of its restricted 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 \(X\) of points \(\{x_i\}_{i=1}^K\) in \(\mathbb{R}^n\) and an arbitrary tessellation \(\{V_i\}_{i=1}^K\) on \(\mathcal{M}\), \(\bigcup_{i=1}^K V_i = \mathcal{M}\), \(V_i \cap V_j = \emptyset, \forall i \neq j\), define the CVT energy functional as follows:

\[
\mathcal{E}(\{x_i, V_i\}_{i=1}^K) = \sum_{i=1}^K \int_{r \in V_i} d^2(r, x_i) dr \tag{12}
\]

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

First, examining the first variation of \(\mathcal{E}\) with respect to a single point, say \(x_i\), we have

\[
\mathcal{E}(x_i + \epsilon v) - \mathcal{E}(x_i) = \int_{r \in V_i} \{d^2(r, x_i + \epsilon v) - d^2(r, x_i)\} dr \tag{13}
\]

where \(v \in \mathbb{R}^n\) and \(\epsilon \in \mathbb{R}\) are arbitrary. Note that in Eqn. \[(13)\], the fixed variables in the argument of \(\mathcal{E}\) are not listed. Dividing the left hand side by \(\epsilon\) and taking \(\epsilon \to 0\), we have

\[
x_i = \frac{\int_{r \in V_i} rdr}{\int_{r \in V_i} 1 dr} \tag{14}
\]

implying that \(x_i\) is the mass centroid of the region \(V_i\).

Second, we fix the points in \(X\) and choose a tessellation \(\{V_i\}_{i=1}^K\) other than the restricted Voronoi tessellation \(RV\mathcal{D}(\{x_i\}_{i=1}^K, \mathcal{M})\). Then we compare \(\mathcal{E}(\{(x_i, V_i(x_i))\}_{i=1}^K)\) with \(\mathcal{E}(\{(x_i, V_i)\}_{i=1}^K)\). For any point \(r \in \mathcal{M}\) belonging to a restricted Voronoi cell \(\mathcal{V}_M(x_i)\), we have

\[
d(r, x_i) \leq d(r, x_j), x_j \in X \tag{15}
\]

Since \(\{V_i\}_{i=1}^K\) is not a Voronoi tessellation, the inequality \[(15)\] must hold strictly over some measure nonzero set in \(\mathcal{M}\). Accordingly,

\[
\mathcal{E}(\{(x_i, V_M(x_i))\}_{i=1}^K) < \mathcal{E}(\{(x_i, V_i)\}_{i=1}^K) \tag{16}
\]

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

Given the initial generators \(\{x_i\}_{i=1}^K, x_i \in \mathcal{M}\), manifold SLIC adopts the Lloyd-like method to iteratively compute RCVT (lines 7-32 in Algorithm 2). Lines 8-14 check if there are bad-shaped Voronoi cells. If so, split these cells. Line 15 checks those Voronoi cells with small areas. If the generators of two neighboring Voronoi cells are close to each other, we merge them into a single Voronoi cell. Lines 16-23 compute the restricted Voronoi tessellation \(RV\mathcal{D}(X, \mathcal{M})\). Lines 25-30 move the generator of each restricted Voronoi cell to its mass center. The algorithm terminates when the generators \(s_i\) do not move or the iteration number reaches a user-specified threshold.

Denote by \(X_i\) the set \(\{x_i\}_{i=1}^K\) at each iteration \(i\) in Algorithm 2. The Lloyd algorithm is a fixed point iteration of a Lloyd map \(T:\)

\[
X_n = T(X_{n-1}), \text{ for } n \geq 1 \tag{17}
\]

The convergence of the Lloyd algorithm directly obtained from the proof of Theorem 1.

\[
\mathcal{E}(X_{n+1}, RV\mathcal{D}(X_{n+1}, \mathcal{M})) \leq \mathcal{E}(X_{n+1}, RV\mathcal{D}(X_n, \mathcal{M})) \\
\leq \mathcal{E}(X_n, RV\mathcal{D}(X_n, \mathcal{M})) \tag{18}
\]
The difference between the proposed RCVT and the constrained CVT (CCVT) is that the center of mass of a Voronoi cell in CCVT is always on the manifold \( \mathcal{M} \), whereas RCVT does not have such a requirement. This seemingly minor change is indeed critical to improve the runtime performance significantly.

### 4.2. A Simple Yet Efficient Method

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

Observe that RCVT tends to generate uniform restricted Voronoi cells (RVC) on \( \mathcal{M} \). Therefore, the area of each RVC is roughly \( \frac{\text{Area}(M)}{K} \). To speed up the construction of a RVC, we limit the size of searching area for each generator to be \( \frac{\text{Area}(M)}{K} \). To quickly find the limited search areas, for each RVC on \( \mathcal{M} \), we project its mass centroid back to the image plane by

\[
P(m) = (u, v),
\]

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

Now we show that Algorithm 2 runs in \( O(N) \) time, which is independent of the number of superpixels \( K \). The key is to show \( \lambda_i \) is bounded by a constant \( \epsilon_{\text{max}} = \max \{ \text{Area}(\Phi(\Omega(s_i))), p_j \in I \} \) that is determined by the color variation of the image \( I \). To see this, note that \( \lambda_i \) reaches maximum when \( \text{Area}(\Phi(\Omega(s_i))) \) is minimum. \( \text{Area}(\Phi(\Omega(s_i))) \) is minimum when \( \Phi(\Omega(s_i)) \) is flat (having the same color), i.e.,

\[
\text{Area}(\Phi(\Omega(s_i))) = 4S^2 = 4N/K.
\]

Therefore \( \lambda_i = \frac{\sqrt{\Xi/\text{Area}(\Phi(\Omega(s_i)))}}{\sqrt{\text{Area}(\mathcal{M})/N}} \leq \epsilon_{\text{max}} \).

In each local search region, there are at most \( 2\lambda_i S \times 2\lambda_i S \leq 4\epsilon_{\text{max}} N/K \) pixels. Then the total number of pixels visited in all \( K \) local regions is bounded by \( 4\epsilon_{\text{max}} N \). Therefore, Algorithm 2 takes linear time \( O(N) \).

### 5. Locally Updating Voronoi Cells

#### 5.1. The Splitting Operator

The splitting operator is used to produce more seeds in content-rich regions. In line 12 of Algorithm 2 a seed \( s_i \) is replaced by four new seeds if the following conditions hold:

- The local search region \( \Omega' = 2\lambda_i S \times 2\lambda_i S \) around \( s_i \) contains very rich content information. This condition is determined by \( \text{Area}(\Phi(\Omega'(s_i))) \) in \( \mathcal{M} \). If it is sufficiently large, i.e., the adaptive scaling factor \( \lambda = \sqrt{\Xi/\text{Area}(\Phi(\Omega'(s_i)))) \) is less than a small threshold (\( \tau = 0.5 \) in Algorithm 2), we consider that this condition holds.

- To prevent endless loop, we can only split a Voronoi cell whose area is larger than \( \Xi/4 \).

If \( s_i \) is the to-be-split seed, we partition the local adaptive local search region \( \Omega' = 2\lambda_i S \times 2\lambda_i S \) into four identical sub-regions and generate four new seeds at the centers of sub-regions. For each new seed, we assign the adaptive scaling factor to be \( \lambda_i/2 \). Due to this regular 1-4 splitting strategy, we set the threshold \( \tau = 0.5 \) in Algorithm 2 (\( \tau = 0.5 \) means that \( \text{Area}(\Phi(s_i)) \) is four times larger than the area \( \Xi \) and after splitting, the expected area of each subregion is still no less than \( \Xi \)). See Figure 4. The positions of the new seeds and their associated adaptive scaling factors will be further optimized in the next iteration.

#### 5.2. The Merging Operator

The splitting operator produces more seeds than the user specified number \( K \). To meet the user’s requirement, Algorithm 2 adopts the merging operator to reduce the number of seeds in content-sparse regions.

We consider the following two cases. First, if the area of \( \mathcal{V}_\mathcal{M}(\Phi(s_i)) \) of a seed is very small (\( \text{Area}(\mathcal{V}_\mathcal{M}(\Phi(s_i))) < \text{Area}(\mathcal{M})/8K \) in our experiment), we randomly choose a neighbor seed \( s_j \) and merge \( s_i \) and \( s_j \). Second, if the sum of areas \( \text{Area}(\mathcal{V}_\mathcal{M}(\Phi(s_i))) + \text{Area}(\mathcal{V}_\mathcal{M}(\Phi(s_j))) \) of two neighbor seeds \( s_i \) and \( s_j \) is smaller than a threshold \( \tau_{\text{area}} \), they are merged into one. We choose the value of \( \tau_{\text{area}} \) to prevent the merged seed being split again by the splitting operator. Denote by \( s_k \) the merged seed of \( s_i \) and \( s_j \) and let \( \text{Area}(\mathcal{V}_\mathcal{M}(\Phi(s_k))) = \text{Area}(\mathcal{V}_\mathcal{M}(\Phi(s_i))) + \text{Area}(\mathcal{V}_\mathcal{M}(\Phi(s_j))) \).
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 \cite{9, 1, 18}. 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. A high boundary recall means that very few true edges are missed. As shown in Figure 5(a) and (b), superpixels generated by manifold SLIC have the least under segmentation error and the highest boundary recall ratio, demonstrating its ability to adhere to image boundaries.

Content Sensitivity. The supplementary material shows typical results of manifold SLIC, in which one can clearly see that the computed superpixels are content sensitive, i.e., they are small in content dense regions and large in content sparse regions. The content sensitive feature is due to the fact that regions of high color variance have larger areas on $\mathcal{M}$.

6. Experimental Results

We implemented manifold SLIC in C++ and tested it on a PC with an Intel I7-860 CPU (2.80GHz) and 8GB RAM. We compared manifold SLIC with several representative methods, including NC \cite{16}, FH \cite{5}, SL \cite{15}, GraphCut \cite{17}, Turbopixels \cite{9}, SLIC \cite{11} and SSS \cite{18}, on the BSDS500 benchmark \cite{13}, where each image has a ground truth segmentation. Following \cite{18}, we evaluated the algorithms on 200 randomly selected images of resolution $481 \times 321$.

Figure 5. Evaluation of representative algorithms and manifold SLIC on the BSDS500 benchmark for $K \in [200, 700]$. Superpixels produced by manifold SLIC have the least under segmentation error and the highest boundary recall ratio, indicating that our method adheres to image boundaries very well. Both SSS and manifold SLIC lead to similar ASA, however, our method significantly outperforms SSS in terms of runtime performance.

Area($\mathcal{V}_M(\Phi(s_j))$). We use Area($\mathcal{V}_M(\Phi(s_k))$) as a low-cost approximation of Area($\mathcal{V}_M(\Phi(s_k))$). We choose $\tau_{area} = \Xi/5$, i.e., Area($\mathcal{V}_M(\Phi(s_k))$) < $\Xi/5$, where $\Xi/5$ characterizes content-sparse areas by noting that in Algorithm \cite{2} a seed $s_i$ is split only if Area($\mathcal{V}_M(\Phi(s_i))$) > $\Xi/4$.

After merging two neighboring seeds $s_i$ and $s_j$, we compute the position of the new seed $s_k$ as

$$s_k = P(\bar{g}_k),$$

where $P$ is the projection operator defined in Eqn. \cite{19} and

$$\bar{g}_k = \frac{\Phi(s_i) \cdot \text{Area}(\mathcal{V}_M(\Phi(s_i))) + \Phi(s_j) \cdot \text{Area}(\mathcal{V}_M(\Phi(s_j)))}{\text{Area}(\mathcal{V}_M(\Phi(s_i))) + \text{Area}(\mathcal{V}_M(\Phi(s_j)))}$$

We show the pseudo codes of the splitting and merging operators and an illustrative example in the supplementary material.
Algorithm 2 reduces to a simple Lloyd algorithm which con-
in cells of small areas. Without these two operators, Al-
of large areas and the merging operator reduces the seeds
areas on \( \mathcal{M} \). See Figure 6.

However, the performance of boundary recall drops slightly.
segmentation error and achievable segmentation accuracy.
ALL+LS+S&M), we observe the improvement of under
pared with randomly selected images (RS+LS+S&M vs.
10 times faster than SSS for
of runtime performance. For example, manifold SLIC runs
10 times faster than SSS for \( K \in [400, 700] \).

Experiments on the Entire BSDS500 Dataset. Compared
with randomly selected images (RS+LS+S&M vs.
ALL+LS+S&M), we observe the improvement of under
segmentation error and achievable segmentation accuracy. However, the performance of boundary recall drops slightly. See Figure 6

Effect on the Local Search. We propose a local method in Section 4.2 to approximate \( RVD(G, \mathcal{M}) \) in \( O(N) \) time. In Figure 6 we compare the approximate solution of our local method and the exact solution obtained by searching on the entire manifold (compare ALL+LS+S&M with ALL+GS+S&M). Judging the results on three metrics, we observe almost no difference between the exact solution and the approximate one.

The splitting operator adds more seeds in cells of large areas and the merging operator reduces the seeds in cells of small areas. Without these two operators, Algorithm 2 reduces to a simple Lloyd algorithm which
verges only to a local minimum. Figure 6 compares the performance of Algorithm 2 with and without these two operators (ALL+LS+S&M vs, ALL+LS(without S&M)) using three metrics. The results demonstrate that the two operators play an important role to jump out of the local minimum, leading to a good approximation of the global minimization to the CVT energy defined in Eqn. (12). It is also worth noting that the splitting and merging operators are designed for our manifold setting and cannot be used in SLIC, because starting from initial seeds, superpixels in each iteration in SLIC have similar areas in the image plane.

7. Conclusion

Manifold SLIC extends the conventional SLIC method to compute content-sensitive superpixels. It maps the input image \( I \) to a 2-dimensional manifold \( \mathcal{M} \) embedded in the combined image and color space \( \mathbb{R}^5 \), whose area elements reflect the density of image content. We propose a simple yet efficient method to compute restricted CVT on \( \mathcal{M} \), which induces the content-sensitive superpixels in \( I \). Computation results on the Berkeley benchmark show that our method outperforms the existing superpixel methods in terms of under segmentation error, boundary recall and achievable segmentation accuracy. Manifold SLIC is also 10 times faster than the state-of-the-art content-sensitive superpixel algorithm. Our future work is to apply GPU parallel computing [14] to further improve the performance of manifold SLIC for high-resolution images.

Acknowledgment

This work was supported by Royal Society-Newton Advanced Fellowship, the Natural Science Foundation of China (61521002, 61432003, 61322206), TNLst Foundation, Tsinghua University Initiative Scientific Research Program, RG23/15 and MOE2013-T2-2-011.
References


