Signal Processing: Image Communication 56 (2017) 28–39

Contents lists available at ScienceDirect

Signal Processing: Image Communication

journal homepage: www.elsevier.com/locate/image

Superpixel segmentation: A benchmark

Murong Wang[a], Xiabi Liu[a,*,1], Yixuan Gao[a], Xiao Ma[a], Nouman Q. Soomro[b]

[a] Department of Computer Science, Beijing Institute of Technology University, Beijing, China
[b] Department of Software Engineering, Mehran University of Engineering & Technology, SZAB Campus, Khairpur, Pakistan

ARTICLE INFO

Keywords:
Superpixel
Benchmark
Evaluation

ABSTRACT

Various superpixel approaches have been published recently. These algorithms are assessed using different evaluation metrics and datasets resulting in discrepancy in algorithm comparison. This calls for a benchmark to compare the state-of-the-arts methods and evaluate their pros and cons. We analyze benchmark metrics, datasets and built a superpixel benchmark. We evaluated and integrated top 15 superpixel algorithms, whose code are publicly available, into one code library and, provide a quantitative comparison of these algorithms. We find that some superpixel algorithms perform consistently better than others. Clustering based superpixel algorithms are more efficient than graph-based ones. Furthermore, we also introduced a novel metric to evaluate superpixel regularity, which is a property that superpixels desired. The evaluation results demonstrate the performance and limitations of state-of-the-art algorithms. Our evaluation and observations give deep insight about different algorithms and will help researchers to identify the more feasible superpixel segmentation methods for their different problems.

1. Introduction

The target of superpixel segmentation is to generate a coherent grouping of pixels, which is also known as image oversegmentation [1]. Ren and Malik [2] proposed the concept of superpixel and generated superpixels by using the normalized cuts [3]. Recently, superpixel segmentation has attracted a lot of interest in computer vision as it provides a convenient way to compute image features and reduce the complexity of subsequent image processing tasks. Many superpixel segmentation algorithms have been proposed in recent years [1,4–21]. The value of superpixel lies in its applications in many fields including object recognition [22], image segmentation [23], object tracking [24], video segmentation [25,26], classification [27], and reconstruction [28]. Each method has its own advantages and disadvantages. It is important to explore the properties of these algorithms. According to the work in Ref. [1] the following properties are desired for superpixel segmentation:

1. Accuracy. Accuracy measures how superpixels adhere well to the image boundaries. In order to preserve more information for future processing on superpixels, superpixel segmentation should be as accurate as possible.
2. Regularity. Regularity measures how a superpixel is close to square Superpixels are used to represent the information of the pixels. Regular superpixels can preserve the typical properties of the pixels, such as spatial topology, structure, homogeneity, and isometric information.
3. Efficiency. As a pre-processing step in computer vision, the superpixel is used to reduce computational complexity. It should be fast and memory-efficient to compute and simple to use.

With the development of superpixel algorithms, it is important to build a united benchmark for superpixel evaluation. Thus, in recent years, superpixel segmentation evaluation has been considered by many researchers. Neubert and Protzel [29] built a standardized evaluation for superpixel segmentation. This benchmark evaluated the accuracy and efficiency of superpixel with several metrics, such as boundary recall undersegmentation error, etc. But the benchmark ignored the regularity of superpixel which is a desired property of superpixel. In Ref. [30], five super-voxel algorithms are evaluated in early video processing. It only focused on evaluating a few super-voxel algorithms on video datasets. Furthermore, they extended these works to a supervoxel library and build a benchmark named LIBSVX [31]. Stutz [32] compared 11 superpixels segmentation algorithms with three

* Corresponding author.
E-mail addresses: wangmurong@bit.edu.cn (M. Wang), liuxiabi@bit.edu.cn (X. Liu), gaoyixuan@bit.edu.cn (Y. Gao), marshall0304@bit.edu.cn (X. Ma), noumansoomro@gmail.com (N.Q. Soomro).
[1] This work was supported in part by National Natural Science Foundation of China [grant numbers 60973059, 81171407] and Program for New Century Excellent Talents in University of China [grantNCET 10-0044].

http://dx.doi.org/10.1016/j.image.2017.04.007
Received 11 November 2016; Received in revised form 7 April 2017; Accepted 9 April 2017
Available online 14 April 2017

metrics: boundary recall, undersegmentation error and runtime. He evaluated the accuracy of superpixel segmentation that was proposed in past years but recent works are not included, such as [4,7]. Furthermore, this work only measures three metrics, some most commonly-used metrics are excluded such as compactness and regularity. In order to make research on superpixel more convenient, a code library that integrates common superpixel segmentation algorithms and evaluation metrics is needed.

Considering the situation described above, we reviewed the literature on superpixel segmentation and built a code library that integrates superpixel segmentation algorithms to facilitate evaluation. Superpixels generated by other algorithms can be evaluated by this library. It is also easy to integrate new metrics to this library. Currently, it integrated 15 superpixel segmentation algorithms and 13 metrics including our proposed novel metric for measuring the regularity of superpixel. Based on this benchmark, results of 15 superpixel segmentation methods, after careful evaluation, are presented in this paper which sheds light on future research in superpixel and its applications. The contributions of this work are summarized as follows.

1. **Code library**. We developed a code library by integrating superpixel segmentation algorithms to facilitate performance evaluation.
2. **Novel regularity metric**. We proposed a novel metric to measure superpixel regularity.
3. **Comprehensive evaluation**. We compared 15 main superpixel segmentation algorithms with 13 metrics and presented comprehensive evaluations of these 15 algorithms.

The rest of this paper is organized as follows. Section 2 describes related work. We introduce evaluated algorithms and used datasets in Section 3. Section 4 shows the benchmark setting and presents our superpixel regularity metrics. Evaluation results are discussed in Section 5. We conclude in Section 6.

2. Superpixel segmentation algorithms

In this section, a review of different superpixel segmentation algorithms is presented. According to different classification criteria superpixel segmentation algorithms are divided into different categories. For example, superpixel segmentation can be categorized as constrained or unconstrained based on whether object function considered the compactness or not [7]. Xu et al. [5] classified superpixel algorithms into bottom-up and top-town algorithms based on the way of superpixel generation. Achanta et al. [1] divided superpixel segmentation algorithms into graph-based approaches and gradient ascent approaches on the basis of iterative process.

In this paper, we classified superpixel algorithms as graph-based and clustering-based by keeping in view the segmentation model.

2.1. Graph-based methods

Graph-based approaches [7,14,3,33,15,16,21] treat each pixel as a node in a graph. Similarities between neighboring pixels are defined as edge weights. Superpixels are generated by minimizing a cost function defined over the graph. Representative works of this category include normalized cuts (N-cut) [3], Felzenszwalb and Huttenlocher (FH) [33], Homogeneous Superpixels (HS) [15], Superpixels via Pseudo-Boolean Optimization (PB) [14], Topology Preserved Regular Superpixel (TPS) [16], Lazy Random Walks Superpixel (LRW) [7], and Entropy Rate Superpixels (ERS) [21]. They are discussed below in detail.

N-Cut. The normalized cut (N-cut) algorithm [3] recursively partitions the graph of all pixels in the image using contour and texture cues. The cost function is defined by computing a fraction of the total edge connections to all the nodes in the graph. It produces regular superpixels. However, the boundary adherence of N-Cut is relatively poor. The efficiency of it is also not very satisfactory. It has a complexity of $O(n^{\frac{3}{2}})$, where n is the number of pixels.

FH. Felzenszwalb and Huttenlocher (FH) [33] agglomerates pixels on the graph by defining a predicted for measuring the evidence for a boundary between the two regions using a graph-based representation of the image. Superpixels are generated by finding the minimum spanning tree of the constituent pixels. Dijkstras algorithm is used to compute the shortest paths in the undirected graph defined on these grid positions. Compared to the N-cut, FH adheres well to image boundaries in practice, but the superpixels have irregular sizes and shapes. Its complexity is $O(N\log N)$, and it cannot directly control the number of superpixels.

HS. The main idea of Homogeneous Superpixels (HS) [15] is to compute superpixels by using Markov clustering (MCL) [34] which is a graph-based algorithm using stochastic flow circulation. The graph nodes are corresponding to the pixels in the input image. An adjacency matrix is initialized using a simple similarity measure to define the edge weight. In order to avoid inhomogeneous pixels and reduce computational time of the MCL, compact pruning [15] is proposed. The complexity of HS is Nr^4, where N is the number of pixels and r is the pruning radius.

PB. Superpixels via Pseudo-Boolean (PB) [14] formats superpixel segmentation as a multi-label assigning problem. Initially, the input image is covered by half-overlapping horizontal strips. Each pixel has the chance to be assigned to one of two alternative latent strips. Then, the task is to decide which strips the pixels belong to. It can be formatted as a binary labeling problem on Markov Random Fields (MRFs). The objective function is composed of two Pseudo-Boolean functions which can be optimized by elimination algorithm [35]. It generates superpixels that are regular in both size and shape. The speed of PB is independent of the number of superpixels, which is usually the bottle-neck of traditional superpixel segmentation algorithms.

TPS. Topology Preserved regular Superpixel (TPS) [16] uses three steps to accomplish superpixel segmentation. First, original seeds are placed into a regular grid. Second, each seed is relocated into the appropriate boundary following maximal edge magnitude constraints. The searching radius is defined as $r = \Gamma\max(\frac{W}{N}, \frac{H}{N})$, where W, H is the image width and height, respectively. The third step is to generate the local optimal path by connecting neighborhood of each relocated seed, vertically and horizontally. Dijkstra's algorithm is used to generate the shortest path in this step. TPS can generate topology preserved regular superpixels which are topology preserved, with a complexity of $O(N\log(N))$. N is the number of superpixels. When the N is less than 300, TPS is the fastest superpixel algorithm.

LRW. Lazy Random Walk (LRW) [7] derives from RW algorithm. The input image is translated into a graph. The vertex of the graph is the image pixel and the edge is defined on Gaussian weighting function. LRW initializes seeds on the graph with the strategy similar as in Ref. [21]. After seed initialization stage, the initial superpixels are iteratively optimized by the new energy function, which is related to the iterative time and the texture measurement. LRW adhere well to the object boundaries.

ERS. Entropy rate superpixel (ERS) [21] formulated superpixel segmentation problem as a maximization problem on a graph and present a novel objective function on the graph topology. The image is mapped to a graph with vertices denoting the pixels and the edge weights denoting the pairwise similarities. The goal of superpixel segmentation is to select a subset of edges from the resulting graph with exactly K connected sub-graphs, where K is the number of superpixels. The complexity of ERS is $O(n)$, where n is the number of pixels.

2.2. Clustering based methods

The clustering-based algorithms [1,4,6,8–13,17–19,36–39] group pixels into clusters (i.e., superpixels) and iteratively refine them until

some convergence criteria are satisfied. Popular clustering based methods include Simple Linear Iterative Clustering (SLIC) [39], Watershed [36], MeanShift [37], QuickShift [38], TurboPixel [10], Manifold SLIC [39], Contour Relaxed Superpixels (CRS) [8], Superpixel Segmentation using Linear Spectral Clustering (LSC) [4], Depth-Adaptive Superpixels (DAS) [18], Compact superpixels (CS) [12], Constant Intensity Superpixels (CIS) [12], SEEDS [17], VCells [19], VCCS [20], Superpixel Lattices [11], Lattices Cut [13],Scene Shape Superpixel (SSP) [40], Co-superpixel [41], Saliency-based superpixel (SBS) [5] and Structure Sensitive Superpixels (SSS) [9].

SLIC. SLIC [1] randomly initializes cluster centers and then the assignment step was used to redefine the location of cluster centers. Next, in the assignment step, each pixel is associated with the nearest cluster center whose search region overlaps its location. Once each pixel has been associated to the nearest cluster center, an update step was used to adjust the cluster centers. The assignment and update steps are repeated iteratively until the error converges. Finally, a post-processing step enforces connectivity by reassigning disjoint pixels to nearby superpixels. Regular superpixels can be obtained after clustering. Superpixel adhere well and efficiently to boundaries. However, the superpixels generated by SLIC cannot capture global image properties. Its complexity is $O(N)$, where N is the number of superpixels.

Watershed. The watershed approach [36] starts from local minimal to produce watersheds, lines that separate catchment basins. The resulting superpixels are highly irregular in size and shape, and also do not exhibit good boundary adherence. The approach of [36] is relatively fast with complexity of $O(N\log N)$, but does not offer control over the amount of superpixels or their compactness. Compact watershed [42] is a modified implementation that considers the compactness constrains of superpixels and can control the superpixel number directly. In our evaluation, we use the implementation of compact watershed.

MeanShift. MeanShift [37], an iterative mode-seeking procedure for locating local maximal of a density function, is applied to find modes in color or intensity feature space of an image. Pixels that converge to the same mode denote the superpixels. MeanShift is a classical image segmentation approach that can be used to produce superpixel. It produces irregular shaped superpixels of no uniform sizes. The complexity of MeanShift is $O(N^2)$, making it relatively slow, and does not offer direct control over the amount, size, or compactness of superpixels.

QuickShift. QuickShift [38] is also a mode-seeking segmentation method. It moves each point in the feature space to the nearest neighbor that increases the Parzen density estimate. It has relatively good boundary adherence with $O(dN^2)$ complexity (d is a small constant, N is the number of pixels). It cannot control the size or the number of superpixels.

TurboPixel. TurboPixel [10] segments an image into a lattice-like structure of compact regions (superpixels) by dilating seeds so as it can to adapt to local image structure. TurboPixel locates a set of seeds using level-set-based geometric flow. The geometric flow relies on local image gradient, aiming to obtain regular superpixels. Unlike watershed [36], TurboPixel [10] superpixels are constrained to have uniform size, compactness, and boundary adherence. The complexity of TurboPixel is $O(N\log N)$ and is claimed to be $O(N)$ in practice.

Manifold SLIC. Manifold SLIC [39] is extended from SLIC to compute content-sensitive superpixels, i.e. small superpixels in content-dense regions and large superpixels in content-sparse regions. In contrast to conventional SLIC that cluster pixels in Lab color space, Manifold SLIC map the image to a 2-dimensional manifold for computing the content density. An efficient method was proposed to compute the Restricted Centroidal Voronoi Tessellation (RCVT) [43]. As a result, it runs 10 times faster than the state-of-the-art content-sensitive superpixel algorithms.

CRS. Contour Relaxed Superpixels (CRS) [8] produces superpixels under the constraint of obtaining maximum homogeneity of the texture inside of each patch, and maximum accordance of the contours with both the image content as well as a Gibbs-Markov random field model. It formulated the superpixel segmentation problem as an estimation task. The energy function that is to be maximized in CRS, has only a very small number of design parameters, depending on the particular statistical model used for the images. They build on the fundamental model used by Mester et al. [11] and transform it into a competitive superpixel approach by introducing a compactness term.

LSC. Linear Spectral Clustering (LSC) [4] maps image pixels to weighted points in ten-dimensional feature space by kernel functions. Then, the seed pixels are sampled uniformly over the whole image. These seeds are used as the search centers and their feature vectors are used as initial weighted means of the corresponding clusters. Each pixel is then assigned to the cluster whose weighted mean is closest to the pixels' vector in the feature space. The weighted mean and search center of each cluster will be updated accordingly. These two steps performed iteratively until the cluster centers are stabilized. Compared with the SLIC, it preserves global properties of images and its superpixels are more regular.

DAS. Depth-adaptive superpixels (DAS) [18] is for RGB-D images which have depth information. It uses the additional depth information to simplify the segmentation task. DAS contains three steps. First, the density of superpixel clusters in the image space is computed from the depth image; second, multi-scale sampling method [44] is used to sample points which will guarantee the blue-noise spectrum property. Last, k-means clustering is used to assign points to superpixels and improves superpixel centers. DAS can generate superpixels in real time.

CS. Compact Superpixels (CS) [12] assumes that the input image is intensively covered by half-overlapping square patches with the same size. Each square patch corresponds to a label. Therefore, hundreds of labels are generated. They then assign each pixel to a unique patch by minimizing an energy function composed of data terms and smoothing terms. The data terms decide each pixel belong to which patch, and the smoothing terms control the boundary of the patch. It uses Potts model wpq from [45] to approximate Euclidean metric between pixels. The patch size is an upper bound of the superpixels. In this method, superpixel sizes tend to be equalized and boundaries are encouraged to be compact by the energy function.

CIS. CS [12] did not explicitly encourage constant intensity superpixels. To generate constant intensity superpixels, Veksler et al. proposed constant intensity superpixels (CIS) [12]. CIS encourages constant intensity inside a superpixel but at the price of relaxing the requirement that superpixels are of roughly equal sizes. It first assigns each patch the color of the pixel at the center. CIS explicit requires that superpixel has constant intensity by adding constraint to the object function. CIS is slower than CS, but the resulting superpixels become coherent in color.

SEEDS. SEEDS [17] starts from an initial superpixel partitioning. Then, superpixels are refined by modifying the boundaries according to the energy function. The function is defined on superpixel boundaries and color histogram and contains two terms: color term and boundary term. The energy function can be solved by the hill-climbing optimization. The running time of SEEDS is related to the iteration time. The authors claimed that SEEDS is able to run in real-time. SEEDS suffers from high shape irregularity and the number of superpixels is difficult to be controlled.

VCells. VCells [19] has two steps; first, the image is divided into small segments with uniform size and shape, and then EWCVT_LNN [43] was used to adjust the boundary of these segments to make the superpixels meet the compactness constraint. EWCVT_LNN is a modified version of EWCVT [43], which has been very successful for general image segmentation problems. The complexity of VCells is $O(K(N_c, N)$, in which K, n_c, N are the number of iterations, superpixels, and pixels, respectively.

VCCS. VCCS [20] is the first algorithm that uses voxel relationships to produce superpixels which are fully consistent with the spatial

geometry of the scene in three-dimensional space. Firstly, the adjacency graph was constructed for the voxel. Then, a number of seed points are chosen as cluster centers in the 3D space. After initialization, VCCS is generated by iteratively adjusting the cluster centers according to the distance in the Fast Point Feature Histograms (FPFH) space [46] until the cluster centers stabilize. The complexity of VCCS is $O(kN)$.

Superpixel Lattices. Superpixel Lattices [11] is a greedy superpixel algorithm that maintains the regular topology of the grid graph of pixels. It generates superpixels by detecting vertical or horizontal strips. Superpixel Lattices formats superpixel segmentation as finding paths in a boundary cost map. It is a 2D array that contains the probability of a semantically meaningful boundary between two pixels. The construction of the superpixel lattice is incremental. Initially, the image is divided into two segments: vertical and horizontal. An additional vertical or horizontal path is added at each subsequent step. The segmentation goal is to find minimum weighted vertical and horizontal paths whose boundary cost is the lowest. S-t min-cut method [47] is used to produce paths in this algorithm. The results of Superpixel-Lattices do not adhere well to the image boundary. It has a complexity $O(N^3 S \log NS)$, where N is the strips length and S is the strip width.

Lattice Cut. Lattice cut [13] starts with a regular grid, and then iteratively merge superpixels. It associates a label with each pixel to indicate which superpixel it belongs to. The problem is formulated as assigning the unknown labels in a Markov random field (MRF) model. Graph-cuts [48,49] is used to find the MAP (Maximum a Posterior) solution. Lattice-cut can exploit both region and edge information and construct a globally optimal solution in either the horizontal or vertical direction using a single graph cut.

SSP. Scene Shape Superpixel (SSP) [40] incorporates prior information into superpixel segmentation. It is motivated by superpixel lattices [11] which performs badly in the non-uniform sampling image. SSP is sensitive for this class of images. It suits for road scene images where objects in the center of the image tend to be more distant and smaller than those at the edge. First, it learns the boundary distribution prior which is used to improve the superpixel segmentation. Then non-uniform minimum cost path algorithm is used to find the shortest path. The running time of SSP is related to the iteration times in the learning step.

Co-superpixel. Co-superpixel [41] generates superpixel via the graph matching. Compared with other superpixel segmentation methods which generate superpixels for a single image, Co-superpixel generates superpixels for a pair of images which contain the same object instances or similar objects. It first generates superpixel for each image, then, these superpixels are merged. The merging process can be divided into two steps. In the first step, these superpixels obtained by over-segmentation in the pair of images are merged based on the appearance similarities with their adjacent superpixels. After this, Co-Superpixels are merged by matching cost and adjacent superpixel appearance similarity.

SBS. Saliency-based superpixel (SBS) [5] obtains superpixels through a merging strategy based on saliency values of image regions. First, superpixels are generated by other superpixel segmentation, then adjacent superpixels are merged according to the saliency of the superpixels. The merging strategy can obtain more meaningful and compact superpixels and decrease the number of superpixels.

SSS. Structure Sensitive Superpixels (SSS) [9] generates superpixels by two steps. First, it puts some seeds roughly in a lattice structure on the image. The seeds serve as initial estimates of the superpixel centers. Then, the location of seeds is refined according to superpixels distribution and magnitudes. The centers are relocated or split by certain criteria related by shape or size. Additional superpixels are created by splitting existing ones until the densities of superpixels satisfied certain conditions. The complexity of SSS is $O(NM_I)$, where N is the number of superpixels, M_I is the total number of iterations.

We summarize the aforementioned superpixel algorithms in Table 1.

Table 1
Superpixel Methods.

Methods	Group	Complexity
Watershed	Gradient-based	$O(N\log N)$
N-cut	Graph-based	$O(N^{\frac{3}{2}})$
MeanShift	Gradient-based	$O(N^2)$
FH	Graph-based	$O(N\log N)$
QuickShift	Gradient-based	$O(dN^2)$
Lattices	Cues based	$O(N^{\frac{3}{2}}\log N)$
TurboPixels	Gradient-based	$O(N)$
SSP	Cues based	$O(N^{\frac{3}{2}}\log N)$
Lattice Cut	Graph-based	$O(N*M)$
CS	Gradient-based	–
CIS	Gradient-based	–
ERS	Gradient-based	–
HS	Graph-based	$O(Nr^4)$
SPB	Graph-based	$O(Nr^4)$
SLIC	Gradient-based	$O(N)$
TPS	Graph-based	$O(N\log N)$
SEEDS	Gradient-based	–
VCells	Graph-based	$O(K\sqrt{n_c}N)$
DAS	Gradient-based	$O(N)$
CRS	Gradient-based	$O(N)$
VCCS	Graph-based	$O(kN)$
SSS	Graph-based	$O(N)$
SBS	Cues based	$O(N)$
LRW	Graph-based	–
LSC	Gradient-based	$O(N)$

3. Evaluated algorithms and code library

A comprehensive benchmark is needed to explore the merits and problems of superpixel algorithms described in the last section. In this section, we give an overview of the evaluated algorithms in this paper. We also introduced the code library. It integrated 15 superpixel segmentation algorithms whose source codes are publicly available. The library also contains 13 computing metrics and programs for plotting results. It can make future research on superpixels and their applications more convenient.

3.1. Evaluated algorithms

For fair evaluation, we evaluated the superpixel segmentation algorithms whose original source or binary codes are publicly available. Therefore, 15 superpixel segmentation methods are chosen, which are SLIC [39], LSC [4], ERS [21], SEEDS [17], PB [14], CRS [8], LRW [7], TPS [16], CS [12,19], CIS [12], N-cut [3], Superpixel Lattices [11], VCells [19], Turbopixel [10], and Watershed [36]. Fig. 1 shows the qualitative results of these methods.

3.2. Code library

We build a code library for evaluating superpixel segmentation algorithms. This library is open and portable. It aims at facilitating fast prototyping and reproducible research for superpixels. It includes 15 state -of-the-the-art superpixel segmentation algorithms such as SLIC, LSC, ERS, SEEDS, and so on. The library provides MATLAB interfaces for each algorithms. This library also coupled with a principled evaluation benchmark based on quantitative criteria for good superpixels. To sum up, our library composed of the following three modules:

1. Superpixel segmentation algorithms. We integrated 15 superpixel segmentation algorithms into the library, whose source codes are publicly available. Each algorithm provides a MATLAB interfaces. With this interface, users can generate superpixels and set algorithms parameters easily.

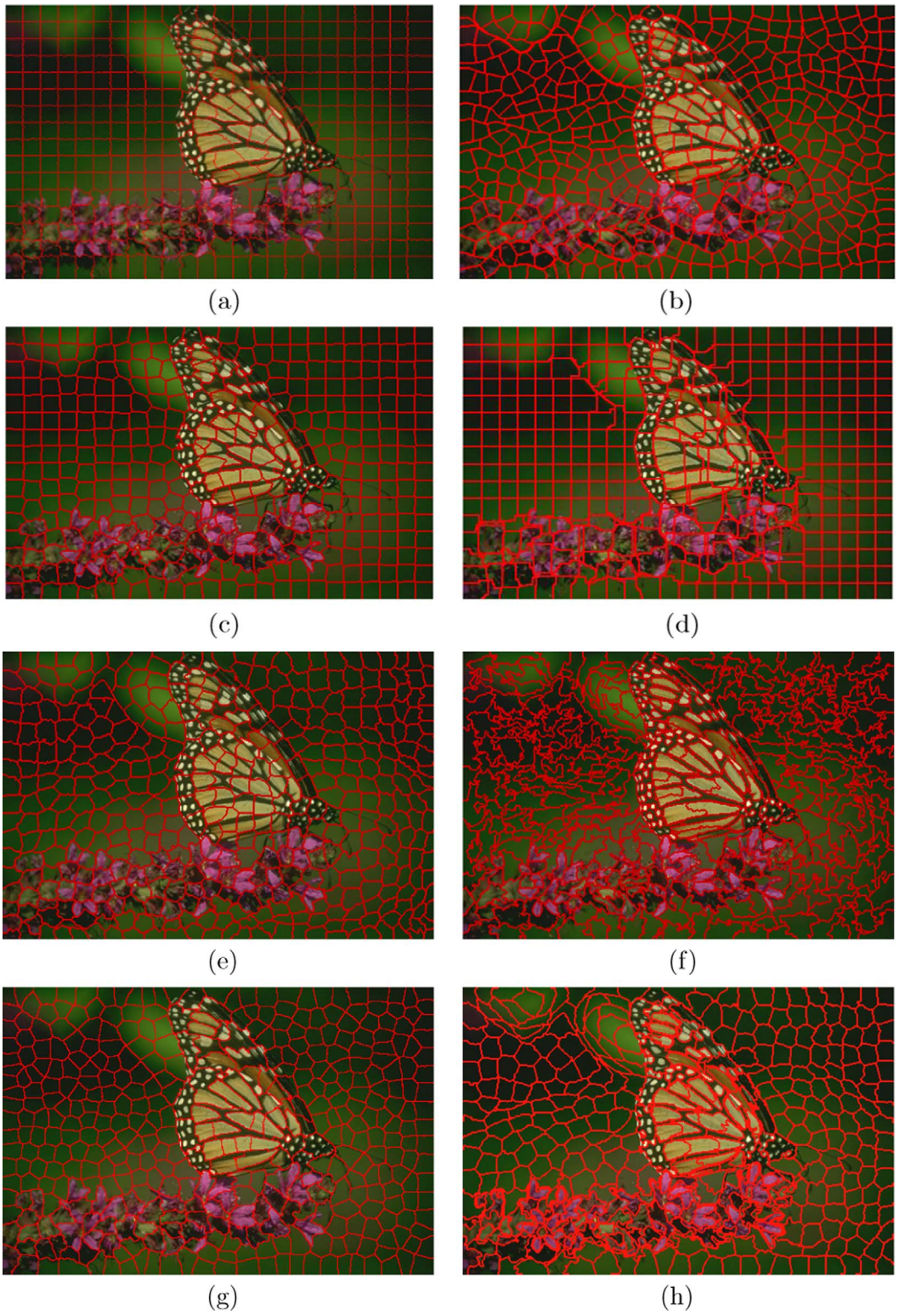

Fig. 1(a). Examples of Superpixel Segmentation Results: (a) Compact Watershed, (b) N-cut, (c) SLIC, (d) TPS, (e) CRS, (f)ERS, (g) LRW, (h) LSC, (i) SEEDS, (j) TurboPixel (k)VCells, (l) Superpixel Lattices, (m) PB, (n) CS, (o) CIS.

2. Superpixel benchmark. We built a benchmark for superpixels, which includes 13 metrics for measuring the superpixel quality. We implement 12 metrics which are always used to evaluate superpixel in past literature computing programs as a part of this library and proposed a novel superpixel metric for superpixel regularity. This is useful for future evaluation of superpixel algorithms.
3. Evaluation of results, computation and plotting. In order to make superpixel evaluation more convenient and easily understandable, the code library contains visualization module of results.

The design principle of this library is easy to be used and extended. So we provide user-friendly interface to set parameters of all these

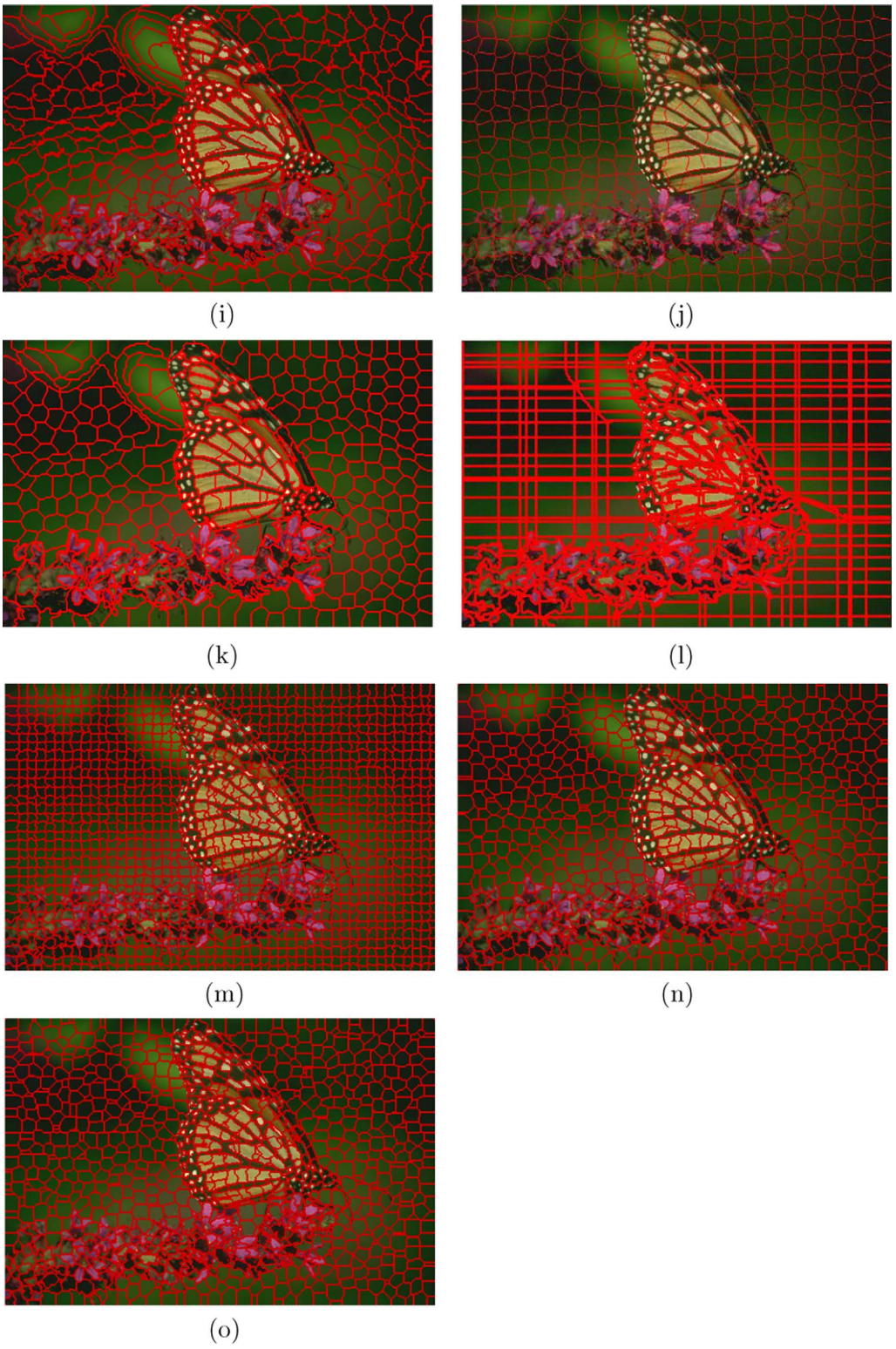

Fig. 1(b). Examples of Superpixel Segmentation Results: (a) Compact Watershed, (b) N-cut, (c) SLIC, (d) TPS, (e) CRS, (f)ERS, (g) LRW, (h) LSC, (i) SEEDS, (j) TurboPixel (k)VCells, (l) Superpixel Lattices, (m) PB, (n) CS, (o) CIS.

methods with much easiness. Moreover, this library is more flexible for adding novel superpixel algorithms and any other metrics into it for superpixel evaluation. This library has been shared on our home page,[2] where the library manual is given to provide more details of the library.

[2] http://www.iscbit.org/source/superpixelbenchmark.zip

4. Benchmark setting

4.1. Dataset

Most of the superpixel algorithms are evaluated on the Berkeley Image Segmentation Dataset [23]. It is consisted of 500 natural images of size 482×321, and split into 200 training, 100 validation and 200

test images. The given ground truth segmentations, at least five per image, have been obtained from different persons and reflect the difficult nature of image segmentation.

4.2. Evaluation criteria

During the past years, several metrics have been used to measure superpixel segmentation methods, such as boundary recall, under-segmentation error, and compactness. All the metrics focus on segmentation accuracy, superpixels compactness, regularity, coherence and efficiency. According to the characters of these metrics, we divided these 13 metrics into three groups. The first group is named segmentation quality evaluation, which considers precision-recall, F-measure [50], variation of information probabilistic rand index, and segmentation covering [23]. The second group is superpixel quality metrics which include undersegmentation error, sum-of-square-error, achievable segmentation accuracy, compactness, explained variation, and regularity index. The last group is computing efficiency measured by the runtime.

More formally, let $S = \{S_1, S_2, \ldots, S_N\}$ be the result of superpixel segmentation for image I, G be the ground truth segmentation of I, then we can define these metrics as follows.

4.3. Segmentation quality metrics

Segmentation Quality Metrics focuses on measuring algorithms the segmentation accuracy. In this paper, we use the metrics proposed in Ref. [23] with Berkeley Image Segmentation Dataset to measure superpixel segmentation quality. Accuracy contains two aspects. One is how the segmentation results adhere to the boundary and the other is the variation of pixels in the segmentations. The first can be measured by Precision-Recall, the second can be measured by Variation of Information (VI), Probabilistic Rand Index (PRI) and Segmentation Covering (SC).

4.3.1. Precision-recall

Precision-recall [2] is a standard boundary-detection and segmentation evaluation criterion. It is widely used in superpixel literature [1,18,19,39] to measure the segmentation quality. It measures the precision-recall curve which captures the trade-off between precision and accuracy. Let S be a superpixel segmentation, G a ground truth segmentation and r a tolerance parameter controlling the allowed deviation from ground truth, then we define:

$TP(S, G)$(True Positives): Boundary pixels in G for which there is a boundary pixel in S in range r.

$FN(S, G)$ (False Negatives): Boundary pixels in G for which there is no boundary pixel in S range r.

$FP(S, G)$ (False Positives): Boundary pixels in S for which there is no boundary pixel in G range r.

In evaluation, r is set to 0.0075 times the image diagonal, which is the default value used by the Berkeley Segmentation Benchmark.

The recall is the fraction of all boundary pixels within the ground truth segmentation G which are correctly detected within the superpixel S that is

$$Recall\ (S, G) = \frac{TP(S, G)}{TP(S, G) + FN(S, G)} \tag{1}$$

The precision is the fraction of all boundary pixels within the ground truth segmentation S not within the ground truth G that is

$$Precision\ (S, G) = \frac{TP(S, G)}{TP(S, G) + FP(S, G)} \tag{2}$$

F-measure is defined on recall and precision with a relative cost α.

$$F_{measure} = \frac{Precision*Recall}{\alpha Recall + (1 - \alpha)*Precision} \tag{3}$$

4.3.2. Variation of Information

The Variation of Information [23] metric measures the distance between two segmentations with their average conditional entropy given by:

$$VI\ (S_i, S_j) = H(S_i) + H(S_j) - 2I(S_i, S_j), \tag{4}$$

where H and I represent the entropies and mutual information between two segmentations S_i,S_j, respectively.

4.3.3. Probabilistic Rand Index

Rand Index [2] was introduced for general clustering evaluation. It operates by comparing the compatibility of assignments between pairs of elements in the clusters. The Rand Index between test superpixel S and G is given by the sum of the number of pairs of pixels that have the same label in S and G and those that have different labels in both segmentations, divided by the total number of pairs of pixels. Given a set of ground-truth segmentations $\{G_k\}$, Probabilistic Rand Index (PRI) amounts to averaging the Rand Index among different ground-truth segmentations:

$$PRI\ (S, \{G_k\}) = \frac{1}{T}\sum_{i<j}[c_{ij}p_{ij} + (1 - c_{ij})(1 - p_{ij})] \tag{5}$$

where c_{ij} indicates that pixels i and j have the same label and p_{ij} its probability.

4.3.4. Segmentation Covering (SC)

The overlap between two segmentation S_i and S_j defined as:

$$O(R, R') = \frac{|R \cap R'|}{|R \cup R'|} \tag{6}$$

We define the covering of a segmentation S by a ground G as:

$$C\ (G \to S) = \frac{1}{N}\sum_{R \in S}|R| \max_{R' \in S'} O\ (R, R') \tag{7}$$

where N denotes the total number of pixels in the image. Similarly, the covering of a machine segmentation S by a family of ground-truth segmentations $\{G_1, G_2, \ldots, G_N\}$ is defined by first covering S separately with each human segmentation G, and then averaging over the different humans.

4.4. Superpixel quality metric

Superpixels are usually used as a pre-processing step in image processing. Compared to the traditional pixel representation of the image, the superpixel representation greatly reduces the number of, image primitives and improves the representative efficiency. In order to make superpixel representation more effective, superpixels should preserve image information as much as possible. According to research in Ref. [51], coherence, compactness and regularity is the most important properties for good superpixels. The most commonly used metrics for the superpixel quality include undersegmentation error, achievable segmentation accuracy, compactness and explained variation, regularity index.

4.4.1. Undersegmentation error

The Undersegmentation Error describes the leakage or bleeding of a superpixel with respect to a specific ground truth segment. We define undersegmentation error according to Xu and Corso [30]. Given a ground-truth segmentation $\{g_1, \ldots, g_K\}$ and a superpixel segmentation $\{S_1, cdots, S_N\}$, we quantify the undersegmentation error for segment for with the fraction:

$$UE(g_i) = \frac{[\sum_{\{s_j | s_j \cap g_i \neq \emptyset\}} - Area(g_i)] - Area(g_i)}{Area(g_i)} \tag{8}$$

where g_i is the ground truth segmentation and s_i is the superpixel. It

measures how each superpixel overlap with only one object. To evaluate the undersegmentation performance of a given algorithm, we simply average the above fraction across all ground truth segments and all images.

4.4.2. Sum-of-Squared Error(SSE)

Sum of Squared Error (SSE) is the sum of the squared differences between each member in the cluster and the cluster mean. It is used to measure the variation within a cluster. SSE is defined as:

$$SSE(S) = \frac{1}{N}\sum_{S_j \in S}\sum_{x_n \in S_j} d(x_n, S_j)^2 \tag{9}$$

Where $d(x_n, S_j)$ is euclidean distance in color space.

$$d(x_n, S_j) = \| I(x_n) - I(S_j)\|_2 \tag{10}$$

Stutz [32] used SSE to measure the quality of the superpixel segmentation. By coloring each pixel with the mean color of corresponding superpixels, the superpixel segmentation is interpreted as the reconstruction of the original image and the SSE measures the reconstruction error.

4.4.3. Achievable segmentation accuracy (ASA)

Superpixels always are used as pre-processing step, we want the performance of subsequent steps to be as far as possible unaffected. We would like to quantify the accuracy achievable by subsequent steps, as for example image segmentation. To compute ASA, we label each superpixel with the label of ground truth segmentation which has the largest overlap. The maximum fraction of correctly labeled pixels in the ground-truth is the achievable segmentation accuracy:

$$ASA_g(S) = \frac{\sum_k \max_i |S_k \cap G_i|}{\sum_i |G_i|} \tag{11}$$

Therefore, ASA represents an upper bound on the accuracy achievable by a subsequent segmentation step.

4.4.4. Compactness

Compactness (CO) is defined based on the isoperimetric quotient [52]. For example, let A_S be the area and L_S be the perimeter of a superpixel S. The circle whose perimeter equal to the perimeter of superpixel, then its radius $r = \frac{L_S}{2\pi}$ Let A_c be the area of the circle. Then the isoperimetric quotient is

$$Q_S = \frac{A_S}{A_C} = \frac{4\pi A_S}{L_S^2}. \tag{12}$$

Based on the isoperimetric quotient [52], the compactness (CO) is

$$CO = \sum_{S \in Q} Q_S \frac{|S|}{|I|}, \tag{13}$$

where S is the set of superpixels of image I, $|S|$ is the size of the set. The compactness is a desirable property for the regular boundaries.

4.4.5. Explained variation

Explained Variation quantifies how well the color variation within the image is captured by the superpixel segmentation. Explained Variation in Ref. [30] is used to evaluate an overall difference between the original pixels and the superpixels as:

$$R^2 = \frac{\sum_{k=1}^{S} k(\mu_k - \mu)^2}{\sum_{i \in I}(x_i - \mu)^2} \tag{14}$$

Where x_i, μ_k and μ denote the actual pixel value, the mean value of superpixel k and the global pixel mean, respectively. The information carried by an image is primarily defined by variation in color or intensity. Thus, Explained Variation measures the fraction of information captured by the superpixel segmentation.

4.4.6. Regularity index

Let $S = \{S_1, S_2, \ldots, S_n\}$ be the superpixel segmentation of image I, then the superpixel area standard deviation (SASD) is defined as

$$S_\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(A_i - A_\mu)^2} \tag{15}$$

where A_i is the area of the i-th superpixel and A_μ is the expectation of superpixels area. Considering that most of these superpixel segmentation algorithms are intended to get more squared superpixels,so, we use the area of the square around the superpixel to replace the circumcircle and define the shape regularity of a superpixel as:

$$S_i(R) = \frac{area(S_i)}{area(Square)} \tag{16}$$

where the $area(S_i)$ is the area of superpixel S_i, and $area(Square)$ is the area of external square, whose edge is the longest of chord of the superpixel. Then the meaning regularity of superpixels' set is

$$S(R) = \frac{1}{N}\sum_{i=1}^{N} S_i(R) \tag{17}$$

then, we get regularity index as:

$$\rho = \frac{S_\sigma^2}{S(R)} = \frac{\sum_{i=1}^{N}(A_i - A_\mu)}{\sum_{i=1}^{N} S_i(R)} \tag{18}$$

5. Evaluation results

In this section, we performed a quantitative comparison of 15 superpixel segmentation algorithms. Superpixels are used to reduce the complexity of subsequent image processing tasks, thus the number of superpixels is an important parameter of superpixels segmentation algorithms. Most of superpixel segmentation algorithms provide parameters to control the number of superpixels. There are three ways for superpixel algorithms to control the number of superpixels.

1. Directly controlled. The number of superpixels is controlled by setting parameters, such as in evaluated SLIC and ERS.
2. Controlled by setting the maximum width and height of superpixels. This category includes PB and Superpixel Lattices.
3. Controlled by setting the maximum or minimum size of superpixels, such as in FH, CIS and CS.

Except the number of superpixels, there are also other parameters in these algorithms, we us the default parameters in the source code provided by the authors.

All the algorithms are evaluated on 500 images in the Berkeley Image Segmentation Dataset. We use the mean value of these metrics to represent the results of these methods.

5.1. Segmentation quality

Fig. 2 shows the evaluation results of boundary recall, precision, and F-measure. From the results in Fig. 2, the following observations are obtained.

1. LSC, ERS, and SEEDS have the highest boundary recall, followed by N-cut. VCells and CIS have the lowest boundary recall. Both VCells and CIS add compactness constraint to the objectness [53] function. This sacrificed the boundary recall of segmentation.
2. Among the precision plots of all algorithms, the precision of N-cut is the highest followed by LRW. They both are graph-based methods. As shown in the Fig. 2(b), blue lines are the graph-based methods, while the red are the clustering based methods. Graph-based methods perform better than clustering-based methods. The reason

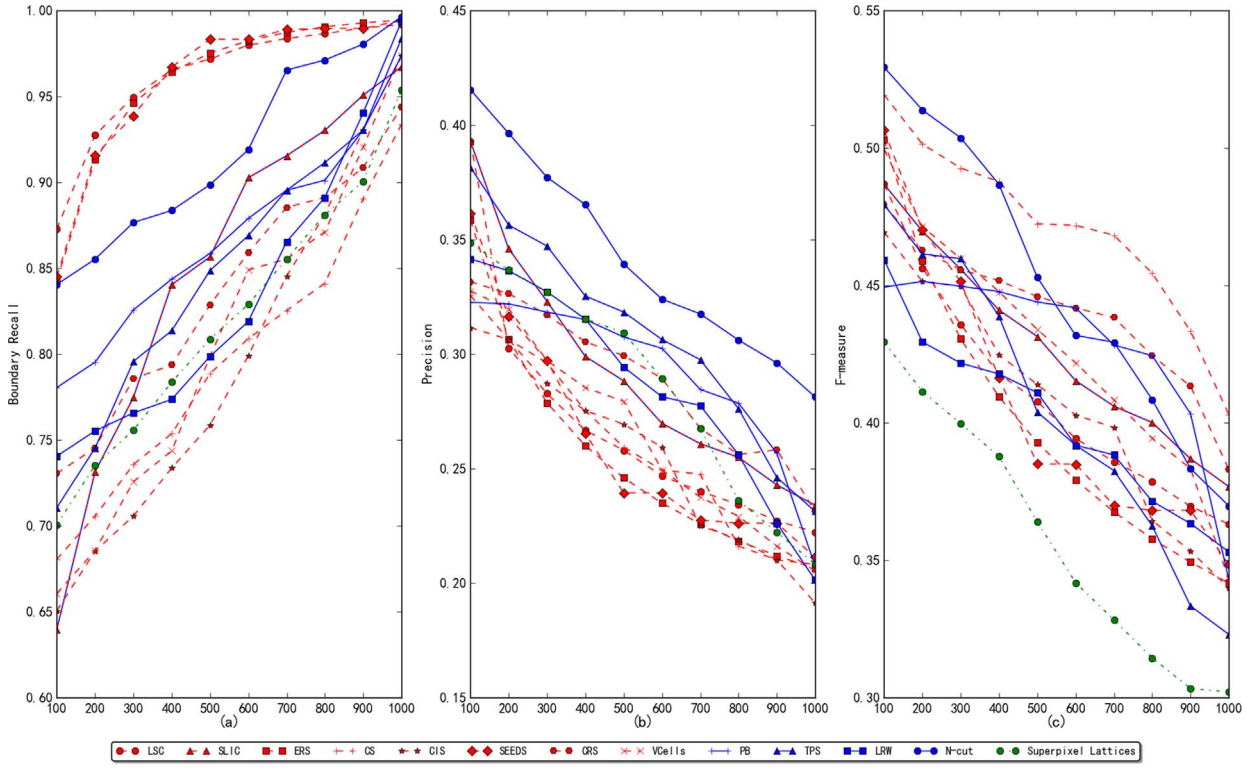

Fig. 2. Evaluation using precision and recall: (a) Recall Comparison, (b) Precision Comparison, (c) F-measure Comparison.

is that graphical methods use a more sophisticated strategy such as lazy random walk to search the boundary between the pixels.

3. SLIC and LSC are both clustering based methods, but the segmentation quality of LSC is better than SLIC. LSC uses a kernel function then explicitly mapping of pixel values and coordinates into a high dimensional feature space. Similarity between pixels are measured by Euclidean distance in the dimensional feature space. This improves the segmentation accuracy.
4. CS and CIS are based on energy minimization framework and optimized with graph cuts. CS performs better than CIS. CIS explicitly encourage constant intensity inside superpixel but at the price of relaxing the requirement that superpixels are of roughly equal sizes. It improved the compactness of superpixel but sacrificed the accuracy of segmentation.

The evaluation results of VI, SC and PRI are shown in Fig. 3. As shown there, LSC, ERS and Superpixel Lattices perform best in SC, followed by VCells, CS and CIS, whereas TPS performed worst. In all algorithms, SC decreases with decreasing size of superpixels. The larger the size of superpixels, the higher SC will be. Superpixel Lattices perform best in SC, because Superpixel generates regular superpixels by adding vertical and horizontal path to the image. The SC will get higher as number of regular superpixels increases.

TPS, SEEDS, and VCells have the highest VI. Higher VI, which means superpixels have smaller coherent sensitiveness. TPS, SEEDS, and VCells consider the compactness property of superpixel which improves the regularity of superpixels but on the cost of sacrificed coherent.

These methods performed differently on PRI. PRI increases with the increment of superpixel number in CS, CIS, CRS, N-Cut, VCells, PB, Superpixel Lattices and LRW But in TPS, LSC, SLIC, ERS and SEEDS, PRI decreased when the number of superpixel increases.

5.2. Superpixel quality

The performance of all the superpixel quality are presented in Fig. 4.

According to the results shown in Fig. 4, the following conclusions can be obtained:

1. All the algorithms ASA is larger than 0.8 and increased with the number of superpixels. And ASA will be closed to 1, if number of superpixels exceeds than 400. ASA has small differences in all these different. ASA can be ignored if number of superpixels are larger than 1000. PB performs worst when the number of superpixels is less than 900. PB generates superpixels by labeling pixels with the strips label. The pseudo boolean optimization caused ASA unstable, that the average value of ASA less than other algorithms.
2. PB performed best in UE, whereas clustering based method perform bad in UE, because clustering based methods only consider the similarity of the pixels in one cluster but the non-similarity between clusters, such as SLIC and LSC.
3. Superpixel Lattices has the highest SSE, followed by CS and CIS. CRS is the only algorithm whose SSE increase with the number of superpixels, because CRS limited the homogeneity of superpixels. When the number of superpixels is increased, in order to satisfy homogeneity constrain then SSE increased.
4. Clustering based algorithms perform better than graph based methods on EV. Moreover, results in Fig. 4 suggested SLIC and LSC algorithms suits for supervoxel in video segmentation. It means that they generate better supervoxel.
5. Whenever the number of superpixels is larger than 500, SLIC performed best on regularity and superpixel regularity contains two aspects: size regularity and shape regularity. SLIC and LSC limit the size of superpixel, hence improves the size regularity. TPS, SEEDS, PB, Superpixel Lattices, CS and CIS all try to get square superpixels. These methods perform well on shape regularity.
6. SLIC has the highest compactness when superpixel number is larger than 500 and SEEDS performs worst in compactness. Because compact measures how the area of the superpixels is closed to the isoperimetric circle. Superpixels which limited maximum area with higher compactness. SLIC and CRS also provide parameter to control superpixel compactness.

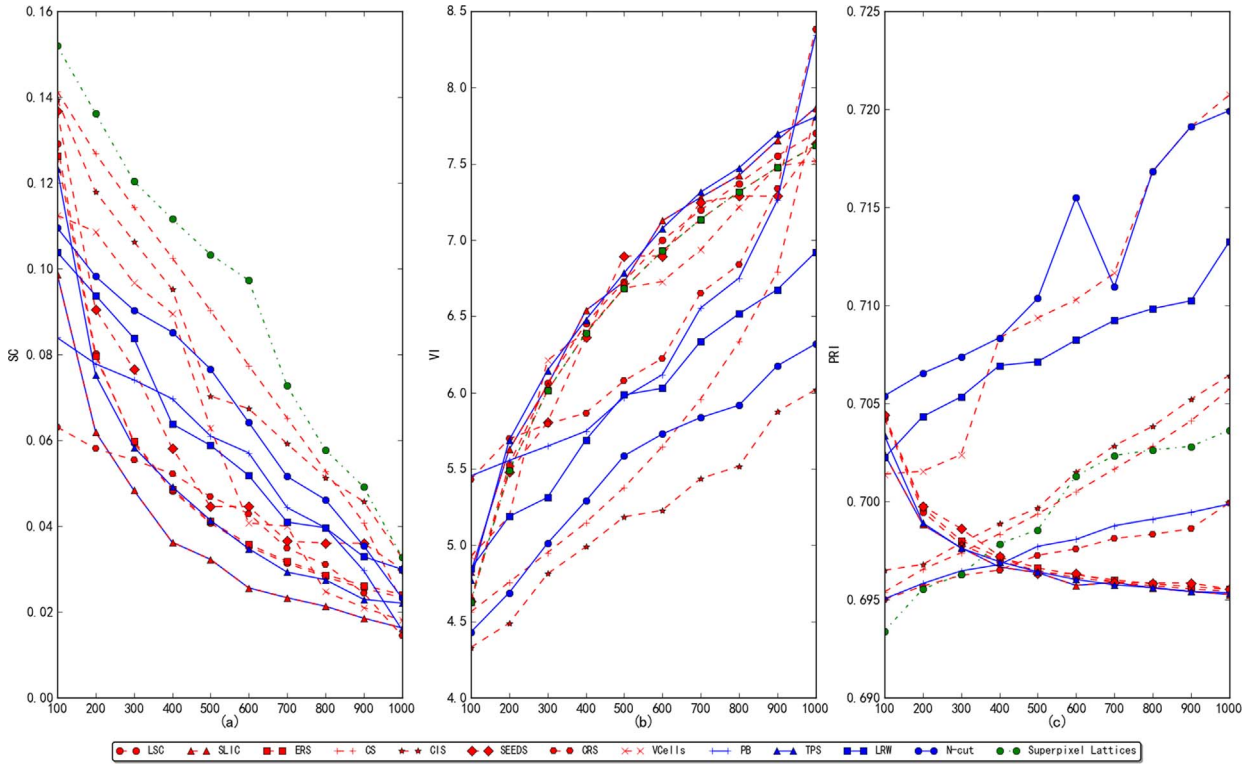

Fig. 3. Segmentation Quality Evaluation Results: (a) SC Comparison, (b) VI Comparison, (c) PRI Comparison.

7. RI is a new metric measuring both size and shape regularity. As shown in Fig. 4. SLIC and LSC perform well on compactness. But RI results show that SLIC performed better than LSC.

5.3. Runtime

Fig. 5 shows the comparison of runtime carried out on an AMD Phenom(tm) II X4 B97 3.20 GHz processor with 8GB RAM. As shown in Fig. 5, PB is the fastest algorithm followed by SEEDS and SLIC when the number of superpixels is large than 300. When the number of superpixels is less than 300, then TPS is the fastest algorithm. The reason is that TPS generate superpixels by finding paths between initial seed points. When the seed points are small, this procedure is very fast. Parallel SLIC algorithms [54] can generate superpixel in real time. Most of the running time is dependent on the number of superpixels, and the running time of N-cut and LRW increased sharply by increasing the

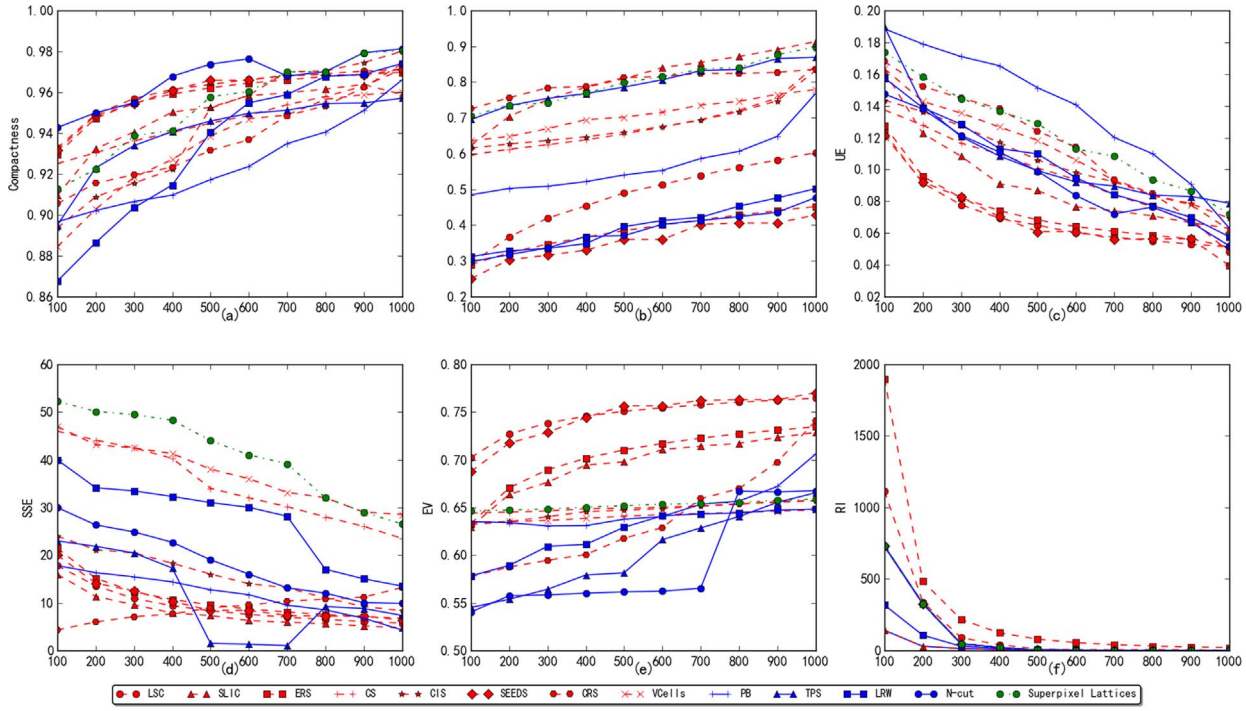

Fig. 4. Superpixel Quality Comparison: (a) ASA Comparison, (b) Compactness Comparison, (c) UE Comparison, (d) SSE Comparison, (e) EV comparison RI Comparison.

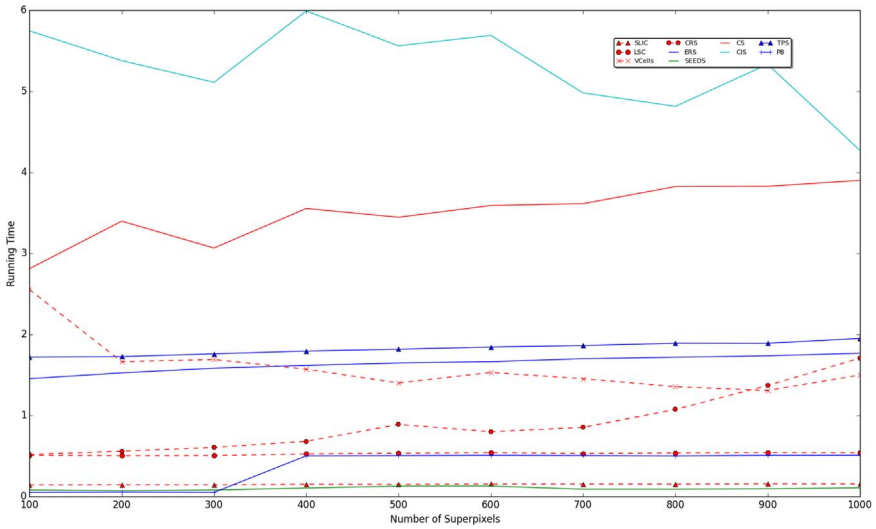

Fig. 5. Runtime.

number of superpixels. Most of the clustering based algorithms are faster and more efficient than graph-based ones such as TPS, SEEDS and SLIC. In practice, they perform badly on high resolution image which usually needs to segment into thousands of superpixels.

6. Conclusions

This paper built a benchmark framework for the evaluation of superpixel segmentation. The benchmark evaluates 15 superpixel segmentation methods from three aspects with 13 metrics and are integrated into one library. This library is easily extendable to novel superpixel segmentation algorithms and metrics. We also proposed a novel metric to measure superpixels' regularity.

Based on the evaluated results presented above, we conclude that graph based superpixel algorithms work better than the clustering based algorithms on segmentation accuracy. Furthermore, Superpixel algorithms which add compact constraint to objectness [53] function produce more compact, coherent and regular superpixels. All of the algorithms evaluated in this paper are hard to apply to the real-time scene.

It is worthwhile to study the trade-off between speed and accuracy of superpixel segmentation. Running time of several algorithms depends mostly on iteration time during optimization, such as LRW, SEEDS, Lattice Cut. Most of these algorithms generate superpixels slowly when the number of superpixels is larger than 1000. So, more efficient superpixel algorithms are desired.

Superpixels produced by other algorithms can be quickly and easily evaluated using the library described in this paper. It is our hope that this benchmark and code library will be useful for researchers working on superpixel segmentation and applications.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Susstrunk, Slic superpixels compared to state-of-the-art superpixel methods, IEEE Trans. Pattern Anal. Mach. Intell. 34 (11) (2012) 2274–2282.
[2] X. Ren, J. Malik, Learning a classification model for segmentation, in: Computer Vision, 2003. Proceedings. in: Proceedings of the Ninth IEEE International Conference on, IEEE, 2003, pp. 10–17.
[3] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.
[4] Z. Li, J. Chen, Superpixel segmentation using linear spectral clustering, in: Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on, IEEE, 2015, pp. 1356–1363.
[5] L. Xu, L. Zeng, Z. Wang, Saliency-based superpixels, Signal, Image Video Process. 8 (1) (2014) 181–190.
[6] H. Fu, X. Cao, D. Tang, Y. Han, D. Xu, Regularity preserved superpixels and supervoxels, IEEE Trans. Multimed. 16 (4) (2014) 1165–1175.
[7] J. Shen, Y. Du, W. Wang, X. Li, Lazy random walks for superpixel segmentation, IEEE Trans. Image Process. 23 (4) (2014) 1451–1462.
[8] C. Conrad, M. Mertz, R. Mester, Contour-relaxed superpixels, in: Energy Minimization Methods in Computer Vision and Pattern Recognition, Springer, 2013, pp. 280–293.
[9] P. Wang, G. Zeng, R. Gan, J. Wang, H. Zha, Structure-sensitive superpixels via geodesic distance, Int. J. Comput. Vision. 103 (1) (2013) 1–21.
[10] A. Levinshtein, A. Stere, K.N. Kutulakos, D.J. Fleet, S.J. Dickinson, K. Siddiqi, Turbopixels: fast superpixels using geometric flows, IEEE Trans. Pattern Anal. Mach. Intell. 31 (12) (2009) 2290–2297.
[11] A.P. Moore, J. Prince, J. Warrell, U. Mohammed, G. Jones, Superpixel lattices, in: Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, IEEE, 2008, pp. 1–8.
[12] O. Veksler, Y. Boykov, P. Mehrani, Superpixels and supervoxels in an energy optimization framework, in: Computer Vision–ECCV 2010, Springer, 2010, pp. 211–224.
[13] A.P. Moore, S.J. Prince, J. Warrell, lattice cut-constructing superpixels using layer constraints, in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE, 2010, pp. 2117–2124.
[14] Y. Zhang, R. Hartley, J. Mashford, S. Burn, Superpixels via pseudo-boolean optimization, in: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 1387–1394.
[15] F. Perbet, A. Maki, Homogeneous superpixels from random walks., in: MVA, 2011, pp. 26–30.
[16] D. Tang, H. Fu, X. Cao, Topology preserved regular superpixel, in: Multimedia and Expo (ICME), 2012 IEEE International Conference on, IEEE, 2012, pp. 765–768.
[17] M. Van den Bergh, X. Boix, G. Roig, B. de Capitani, L. Van Gool, Seeds: Superpixels extracted via energy-driven sampling, in: Computer Vision–ECCV 2012, Springer, 2012, pp. 13–26.
[18] D. Weikersdorfer, D. Gossow, M. Beetz, Depth-adaptive superpixels, in: Pattern Recognition (ICPR), 2012 Proceedings of the 21st International Conference on, IEEE, 2012, pp. 2087–2090.
[19] J. Wang, X. Wang, Vcells: simple and efficient superpixels using edge-weighted centroidal voronoi tessellations, IEEE Trans. Pattern Anal. Mach. Intell. 34 (6) (2012) 1241–1247.
[20] J. Papon, A. Abramov, M. Schoeler, F. Worgotter, Voxel cloud connectivity segmentation-supervoxels for point clouds, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 2027–2034.
[21] M.-Y. Liu, O. Tuzel, S. Ramalingam, R. Chellappa, Entropy rate superpixel segmentation, in: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE, 2011, pp. 2097–2104.
[22] D. Giordano, F. Murabito, S. Palazzo, C. Spampinato, Superpixel-based video object segmentation using perceptual organization and location prior, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 4814–4822.
[23] P. Arbelaez, M. Maire, C. Fowlkes, J. Malik, Contour detection and hierarchical image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 33 (5) (2011) 898–916.
[24] F. Yang, H. Lu, M.-H. Yang, Robust superpixel tracking, IEEE Trans. Image Process. 23 (4) (2014) 1639–1651.
[25] Z. Tian, N. Zheng, J. Xue, X. Lan, C. Li, G. Zhou, Video object segmentation with shape cue based on spatiotemporal superpixel neighbourhood, IET Comput. Vision. 8 (1) (2014) 16–25.
[26] F. Galasso, R. Cipolla, B. Schiele, Video segmentation with superpixels, in: Computer Vision–ACCV 2012, Springer, 2012, pp. 760–774.
[27] J. Cheng, J. Liu, Y. Xu, F. Yin, D.W.K. Wong, B.-H. Lee, C. Cheung, T. Aung, T.Y. Wong, Superpixel classification for initialization in model based optic disc segmentation, in: Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE, IEEE, 2012, pp. 1450–1453.

[28] J. Cheng, J. Liu, Y. Xu, F. Yin, D.W.K. Wong, N.-M. Tan, D. Tao, C.-Y. Cheng, T. Aung, T.Y. Wong, Superpixel classification based optic disc and optic cup segmentation for glaucoma screening, IEEE Trans. Med. Imaging 32 (6) (2013) 1019–1032.
[29] P. Neubert, P. Protzel, Superpixel benchmark and comparison, in: Proceedings Forum Bildverarbeitung, 2012, pp. 1–12.
[30] C. Xu, J.J. Corso, Evaluation of super-voxel methods for early video processing, in: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE, 2012, pp. 1202–1209.
[31] C. Xu, J.J. Corso, Libsvx: a supervoxel library and benchmark for early video processing, Int. J. Comput. Vision. 119 (3) (2016) 272–290.
[32] D. Stutz, Superpixel segmentation: An evaluation, in: Pattern Recognition, Springer, 2015, pp. 555–562.
[33] P.F. Felzenszwalb, D.P. Huttenlocher, Efficient graph-based image segmentation, Int. J. Comput. Vision. 59 (2) (2004) 167–181.
[34] S.M. Van Dongen, Graph clustering by flow simulation.
[35] P. Carr, R. Hartley, Minimizing energy functions on 4-connected lattices using elimination, in: 2009 IEEE Proceedings of the 12th International Conference on Computer Vision, IEEE, 2009, pp. 2042–2049.
[36] L. Vincent, P. Soille, Watersheds in digital spaces: an efficient algorithm based on immersion simulations, IEEE Trans. Pattern Anal. Mach. Intell. 6 (1991) 583–598.
[37] D. Comaniciu, P. Meer, Mean shift: a robust approach toward feature space analysis, IEEE Trans. Pattern Anal. Mach. Intell. 24 (5) (2002) 603–619.
[38] A. Vedaldi, S. Soatto, Quick shift and kernel methods for mode seeking, in: Computer vision–ECCV 2008, Springer, 2008, pp. 705–718.
[39] Y.-J. Liu, C.-C. Yu, M.-J. Yu, Y. He, Manifold slic: A fast method to compute content-sensitive superpixels, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 651–659.
[40] A.P. Moore, S.J. Prince, J. Warrell, U. Mohammed, G. Jones, Scene shape priors for superpixel segmentation, in: Computer Vision, 2009 IEEE Proceedings of the 12th International Conference on, IEEE, 2009, pp. 771–778.
[41] Y. Xie, L. Xu, Z. Wang, Automated co-superpixel generation via graph matching, Signal, Image Video Process. 8 (4) (2014) 753–763.
[42] P. Neubert, P. Protzel, Compact watershed and preemptive slic: On improving trade-offs of superpixel segmentation algorithms., in: ICPR, 2014, pp. 996–1001.
[43] J. Wang, L. Ju, X. Wang, An edge-weighted centroidal voronoi tessellation model for image segmentation, IEEE Trans. Image Process. 18 (8) (2009) 1844–1858.
[44] R. Fattal, Blue-noise point sampling using kernel density model, in: ACM Transactions on Graphics (TOG), Vol. 30, ACM, 2011, p. 48.
[45] Y. Boykov, G. Funka-Lea, Graph cuts and efficient nd image segmentation, Int. J. Comput. Vision. 70 (2) (2006) 109–131.
[46] R.B. Rusu, N. Blodow, M. Beetz, Fast point feature histograms (fpfh) for 3d registration, in: Robotics and Automation, 2009. ICRA'09. IEEE International Conference on, IEEE, 2009, pp. 3212–3217.
[47] A. Goldberg, R. Tarjan, Solving minimum-cost flow problems by successive approximation, in: Proceedings of the nineteenth annual ACM symposium on Theory of computing, ACM, 1987, pp. 7–18.
[48] Y.Y. Boykov, M.-P. Jolly, Interactive graph cuts for optimal boundary & region segmentation of objects in nd images, in: Computer Vision, 2001. ICCV 2001. Proceedings. in: Proceedings of the Eighth IEEE International Conference on, vol. 1, IEEE, 2001, pp. 105–112.
[49] C. Rother, V. Kolmogorov, A. Blake, Grabcut: Interactive foreground extraction using iterated graph cuts, in: ACM transactions on graphics (TOG), vol. 23, ACM, 2004, pp. 309–314.
[50] D.R. Martin, C.C. Fowlkes, J. Malik, Learning to detect natural image boundaries using local brightness, color, and texture cues, IEEE Trans. Pattern Anal. Mach. Intell. 26 (5) (2004) 530–549.
[51] A. Hanbury, How do superpixels affect image segmentation?, in: Iberoamerican Congress on Pattern Recognition, Springer, 2008, pp. 178–186.
[52] W.M. Rand, Objective criteria for the evaluation of clustering methods, J. Am. Stat. Assoc. 66 (336) (1971) 846–850.
[53] B. Alexe, T. Deselaers, V. Ferrari, Measuring the objectness of image windows, IEEE Trans. Pattern Anal. Mach. Intell. 34 (11) (2012) 2189–2202.
[54] C.Y. Ren, I. Reid, gslic: a real-time implementation of slic superpixel segmentation, University of Oxford, Department of Engineering, Technical Report.