SLIC Superpixels Compared to State-of-the-art Superpixel Methods

Radhakrishna Achanta, Appu Shaji, Kevin Smith,
Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk

***Abstract*—Computer vision applications have come to rely increasingly on superpixels in recent years, but it is not always clear what constitutes a good superpixel algorithm. In an effort to understand the benefits and drawbacks of existing methods, we empirically compare five state-of-the-art superpixel algorithms for their ability to adhere to image boundaries, speed, memory efficiency, and their impact on segmentation performance. We then introduce a new superpixel algorithm, *simple linear iterative clustering* (SLIC), which adapts a k-means clustering approach to efficiently generate superpixels. Despite its simplicity, SLIC adheres to boundaries as well as or better than previous methods. At the same time, it is faster and more memory efficient, improves segmentation performance, and is straightforward to extend to supervoxel generation.**

***Index Terms*—Superpixels, segmentation, clustering, k-means.**

I. Introduction

Superpixel algorithms group pixels into perceptually meaningful atomic regions, which can be used to replace the rigid structure of the pixel grid. They capture image redundancy, provide a convenient primitive from which to compute image features, and greatly reduce the complexity of subsequent image processing tasks. They have become key building blocks of many computer vision algorithms, such as top scoring multi-class object segmentation entries to the PASCAL VOC Challenge [9], [29], [11], depth estimation [30], segmentation [16], body model estimation [22], and object localization [9].

There are many approaches to generate superpixels, each with its own advantages and drawbacks that may be better suited to a particular application. For example, if adherence to image boundaries is of paramount importance, the graph-based method of [8] may be an ideal choice. However, if superpixels are to be used to build a graph, a method that produces a more regular lattice, such as [23], is probably a better choice. While it is difficult to define what constitutes an ideal approach for all applications, we believe the following properties are generally desirable:

1) Superpixels should adhere well to image boundaries.
2) When used to reduce computational complexity as a preprocessing step, superpixels should be fast to compute, memory efficient, and simple to use.
3) When used for segmentation purposes, superpixels should both increase the speed and improve the quality of the results.

We therefore performed an empirical comparison of five state-of-the-art superpixel methods [8], [23], [26], [25], [15], evaluating their speed, ability to adhere to image boundaries, and impact on segmentation performance. We also provide a qualitative review of these, and other, superpixel methods. Our conclusion is that no existing method is satisfactory in all regards.

All authors are with the School of Computer and Communication Sciences (IC), École Polytechnique Fédérale de Lausanne (EPFL), Switzerland.
E-mail: firstname.lastname@epfl.ch

Fig. 1: Images segmented using SLIC into superpixels of size 64, 256, and 1024 pixels (approximately).

To address this, we propose a new superpixel algorithm: *simple linear iterative clustering* (SLIC), which adapts k-means clustering to generate superpixels in a manner similar to [30]. While strikingly simple, SLIC is shown to yield state-of-the-art adherence to image boundaries on the Berkeley benchmark [20], and outperforms existing methods when used for segmentation on the PASCAL [7] and MSRC [24] data sets. Furthermore, it is faster and more memory efficient than existing methods. In addition to these quantifiable benefits, SLIC is easy to use, offers flexibility in the compactness and number of the superpixels it generates, is straightforward to extend to higher dimensions, and is freely available[1].

II. Existing superpixel methods

Algorithms for generating superpixels can be broadly categorized as either graph-based or gradient ascent methods. Below, we review popular superpixel methods for each of these categories, including some that were not originally designed specifically to generate superpixels. Table I provides a qualitative and quantitative summary of the reviewed methods, including their relative performance.

A. Graph-based algorithms

Graph-based approaches to superpixel generation treat each pixel as a node in a graph. Edge weights between two nodes are proportional to the similarity between neighboring pixels. Superpixels are created by minimizing a cost function defined over the graph.

NC05 – The Normalized cuts algorithm [23] recursively partitions a graph of all pixels in the image using contour and texture cues, globally minimizing a cost function defined on the edges at the partition boundaries. It produces very

[1]Cross-platform executables and source code for SLIC superpixels and supervoxels can be found at http://ivrg.epfl.ch/research/superpixels

TABLE I: Summary of existing superpixel algorithms. The ability of a superpixel method to adhere to boundaries found in the Berkeley data set [20] is measured and ranked according to two standard metrics: under-segmentation error and boundary recall (for ∼500 superpixels). We also report the average time required to segment images using an Intel Dual Core 2.26 GHz processor with 2GB RAM, and the class-averaged segmentation accuracy obtained on the MSRC data set using the method described in [11]. Bold entries indicate best performance in each category. Ability to specify the amount of superpixels, control their compactness, and ability to generate supervoxels is also provided.

	Graph-based					Gradient-ascent-based				
	GS04 [8]	NC05 [23]	SL08 [21]	GCa10[b] [26]	GCb10[b] [26]	WS91 [28]	MS02 [4]	TP09[b] [15]	QS09 [25]	SLIC
Adherence to boundaries										
Under-segmentation error (rank)	0.23	0.22	-	0.22	0.22	-	-	0.24	0.20	**0.19**
Boundary recall (rank)	**0.84**	0.68	-	0.69	0.70	-	-	0.61	0.79	0.82
Segmentation speed										
320×240 image	1.08s[a]	178.15s	-	5.30s	4.12s	-	-	8.10s	4.66s	**0.36s**
2048×1536 image	90.95s[a]	N/A[c]	-	315s	235s	-	-	800s	181s	**14.94s**
Segmentation accuracy (using [11] on MSRC)	74.6%	75.9%	-	-	73.2%	-	-	62.0%	75.1%	**76.9%**
Control over amount of superpixels	No	Yes	Yes	Yes	Yes	No	No	Yes	No	Yes
Control over superpixel compactness	No	No	No	No[d]	No[d]	No	No	No	No	Yes
Supervoxel extension	No	No	No	Yes	Yes	Yes	No	No	No	Yes

[a] Reported time includes parameter search. [b] Considers intensity only, ignores color. [c] NC05 failed to segment 2048×1536 images, producing "out of memory" errors.
[d] Constant-intensity (GCa10) or compact (GCb10) superpixels can be selected.

regular, visually pleasing superpixels. However, the boundary adherence of NC05 is relatively poor and it is the slowest among the methods (particularly for large images), although attempts to speed up the algorithm exist [5]. NC05 has a complexity of $O(N^{\frac{3}{2}})$ [15], where N is the number of pixels.

GS04 – Felzenszwalb and Huttenlocher [8] propose an alternative graph-based approach that has been applied to generate superpixels. It performs an agglomerative clustering of pixels as nodes on a graph, such that each superpixel is the minimum spanning tree of the constituent pixels. GS04 adheres well to image boundaries in practice, but produces superpixels with very irregular sizes and shapes. It is $O(N \log N)$ complex and fast in practice. However, it does not offer an explicit control over the amount of superpixels or their compactness.

SL08 – Moore *et al.* propose a method to generate superpixels that conform to a grid by finding optimal paths, or seams, that split the image into smaller vertical or horizontal regions [21]. Optimal paths are found using a graph cuts method similar to *Seam Carving* [1]. While the complexity of SL08 is $O(N^{\frac{3}{2}} \log N)$ according to the authors, this does not account for the pre-computed boundary maps, which strongly influence the quality and speed of the output.

GCa10 and **GCb10** – In [26], Veksler *et al.* use a global optimization approach similar to the texture synthesis work of [14]. Superpixels are obtained by stitching together overlapping image patches such that each pixel belongs to only one of the overlapping regions. They suggest two variants of their method, one for generating compact superpixels (GCa10) and one for constant-intensity superpixels (GCb10).

B. Gradient-ascent-based algorithms

Starting from a rough initial clustering of pixels, gradient ascent methods iteratively refine the clusters until some convergence criterion is met to form superpixels.

MS02 – In [4], mean shift, an iterative mode-seeking procedure for locating local maxima of a density function, is applied to find modes in the color or intensity feature space of an image. Pixels that converge to the same mode define the superpixels. MS02 is an older approach, producing irregularly shaped superpixels of non-uniform size. It is $O(N^2)$ complex, making it relatively slow and does not offer direct control over the amount, size, or compactness of superpixels.

QS08 – Quick shift [25] also uses a mode-seeking segmentation scheme. It initializes the segmentation using a medoid shift procedure. It then moves each point in the feature space to the nearest neighbor that increases the Parzen density estimate. While it has relatively good boundary adherence, QS08 is quite slow, with an $O(dN^2)$ complexity (d is a small constant [25]). QS08 does not allow for explicit control over the size or number of superpixels. Previous works have used QS08 for object localization [9] and motion segmentation [2].

WS91 – The watershed approach [28] performs a gradient ascent starting from local minima to produce *watersheds*, lines that separate catchment basins. The resulting superpixels are often highly irregular in size and shape, and do not exhibit good boundary adherence. The approach of [28] is relatively fast ($O(N \log N)$ complexity), but does not offer control over the amount of superpixels or their compactness.

TP09 – The Turbopixel method progressively dilates a set of seed locations using level-set based geometric flow [15]. The geometric flow relies on local image gradients, aiming to regularly distribute superpixels on the image plane. Unlike WS91, TP09 superpixels are constrained to have uniform size, compactness, and boundary adherence. TP09 relies on algorithms of varying complexity, but in practice, as the authors claim, has approximately $O(N)$ behaviour [15]. However, it is among the slowest algorithms examined and exhibits relatively poor boundary adherence.

III. SLIC SUPERPIXELS

We propose a new method for generating superpixels which is faster than existing methods, more memory efficient,

exhibits state-of-the-art boundary adherence, and improves the performance of segmentation algorithms. *Simple linear iterative clustering* (SLIC) is an adaptation of k-means for superpixel generation, with two important distinctions:

1) The number of distance calculations in the optimization is dramatically reduced by limiting the search space to a region proportional to the superpixel size. This reduces the complexity to be linear in the number of pixels N – and independent of the number of superpixels k.
2) A weighted distance measure combines color and spatial proximity, while simultaneously providing control over the size and compactness of the superpixels.

SLIC is similar to the approach used as a preprocessing step for depth estimation described in [30], which was not fully explored in the context of superpixel generation.

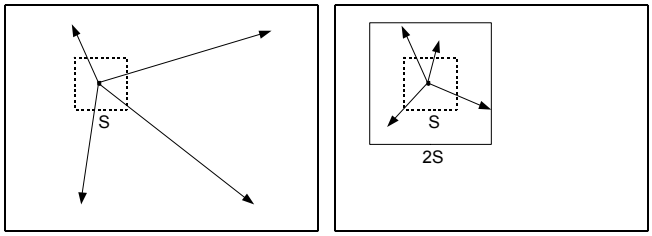
A. Algorithm

SLIC is simple to use and understand. By default, the only parameter of the algorithm is k, the desired number of approximately equally-sized superpixels.[2] For color images in the CIELAB color space, the clustering procedure begins with an initialization step where k initial cluster centers $C_i = [l_i \ a_i \ b_i \ x_i \ y_i]^{\mathrm{T}}$ are sampled on a regular grid spaced S pixels apart. To produce roughly equally sized superpixels, the grid interval is $S = \sqrt{N/k}$. The centers are moved to seed locations corresponding to the lowest gradient position in a 3×3 neighborhood. This is done to avoid centering a superpixel on an edge, and to reduce the chance of seeding a superpixel with a noisy pixel.

Next, in the assignment step, each pixel i is associated with the nearest cluster center whose search region overlaps its location, as depicted in Fig. 2. This is the key to speeding up our algorithm because limiting the size of the search region significantly reduces the number of distance calculations, and results in a significant speed advantage over conventional k-means clustering where each pixel must be compared with all cluster centers. This is only possible through the introduction of a distance measure D, which determines the nearest cluster center for each pixel, as discussed in Section III-B. Since the expected spatial extent of a superpixel is a region of approximate size $S \times S$, the search for similar pixels is done in a region $2S \times 2S$ around the superpixel center.

Once each pixel has been associated to the nearest cluster center, an update step adjusts the cluster centers to be the mean $[l \ a \ b \ x \ y]^{\mathrm{T}}$ vector of all the pixels belonging to the cluster. The L_2 norm is used to compute a residual error E between the new cluster center locations and previous cluster center locations. The assignment and update steps can be repeated iteratively until the error converges, but we have found that 10 iterations suffices for most images, and report all results in this paper using this criteria. Finally, a post-processing step enforces connectivity by re-assigning disjoint pixels to nearby superpixels. The entire algorithm is summarized in Algorithm 1.

[2]Optionally, the compactness of the superpixels can be controlled by adjusting m, which is discussed in Section III-B.



(a) standard k-means searches the entire image

(b) SLIC searches a limited region

Fig. 2: Reducing the superpixel search regions. The complexity of SLIC is linear in the number of pixels in the image $O(N)$, while the conventional k-means algorithm is $O(kNI)$ where I is the number of iterations. This is achieved by limiting the search space of each cluster center in the assignment step. *(a)* In the conventional k-means algorithm, distances are computed from each cluster center to every pixel in the image. *(b)* SLIC only computes distances from each cluster center to pixels within a $2S \times 2S$ region. Note that the expected superpixel size is only $S \times S$, indicated by the smaller square. This approach not only reduces distance computations but also makes SLIC's complexity independent of the number of superpixels.

Algorithm 1 SLIC superpixel segmentation

/ Initialization */*
Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S.
Move cluster centers to the lowest gradient position in a 3×3 neighborhood.
Set label $l(i) = -1$ for each pixel i.
Set distance $d(i) = \infty$ for each pixel i.
repeat
  / Assignment */*
  for each cluster center C_k **do**
    for each pixel i in a $2S \times 2S$ region around C_k **do**
      Compute the distance D between C_k and i.
      if $D < d(i)$ **then**
        set $d(i) = D$
        set $l(i) = k$
      end if
    end for
  end for
  / Update */*
  Compute new cluster centers.
  Compute residual error E.
until $E \leq$ threshold

B. Distance measure

SLIC superpixels correspond to clusters in the $labxy$ color–image plane space. This presents a problem in defining the distance measure D, which may not be immediately obvious. D computes the distance between a pixel i and cluster center C_k in Algorithm 1. A pixel's color is represented in the CIELAB color space $[l \ a \ b]^{\mathrm{T}}$, whose range of possible values is known. The pixel's position position $[x \ y]^{\mathrm{T}}$, on the other hand, may take a range of values that varies according to the size of the image.

Simply defining D to be the five-dimenensional Euclidean distance in $labxy$ space will cause inconsistencies in clustering

behavior for different superpixel sizes. For large superpixels, spatial distances outweigh color proximity, giving more relative importance to spatial proximity than color. This produces compact superpixels that do not adhere well to image boundaries. For smaller superpixels, the converse is true.

To combine the two distances into a single measure, it is necessary to normalize color proximity and spatial proximity by their respective maximum distances within a cluster, N_s and N_c. Doing so, D' is written

$$\begin{aligned} d_c &= \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2} \\ d_s &= \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \\ D' &= \sqrt{\left(\frac{d_c}{N_c}\right)^2 + \left(\frac{d_s}{N_s}\right)^2}. \end{aligned} \quad (1)$$

The maximum spatial distance expected within a given cluster should correspond to the sampling interval, $N_S = S = \sqrt{(N/K)}$. Determining the maximum color distance N_c is not so straightforward, as color distances can vary significantly from cluster to cluster and image to image. This problem can be avoided by fixing N_c to a constant m so that Eq. 1 becomes

$$D' = \sqrt{\left(\frac{d_c}{m}\right)^2 + \left(\frac{d_s}{S}\right)^2}, \quad (2)$$

which simplifies to the distance measure we use in practice

$$D = \sqrt{{d_c}^2 + \left(\frac{d_s}{S}\right)^2 m^2}. \quad (3)$$

By defining D in this manner, m also allows us to weigh the relative importance between color similarity and spatial proximity. When m is large, spatial proximity is more important and the resulting superpixels are more compact (i.e. they have a lower area to perimeter ratio). When m is small, the resulting superpixels adhere more tightly to image boundaries, but have less regular size and shape. When using the CIELAB color space, m can be in the range $[1, 40]$.

Equation 3 can be adapted for grayscale images by setting $d_c = \sqrt{(l_j - l_i)^2}$. It can also be extended to handle 3D supervoxels, as depicted in Figure 3, by including the depth dimension to the spatial proximity term of Eq. 3

$$d_s = \sqrt{(x_i - x_j)^2 + (y_i - y_j) + (z_i - z_j)^2}. \quad (4)$$

C. Post-processing

Like some other superpixel algorithms [8], SLIC does not explicitly enforce connectivity. At the end of the clustering procedure, some "orphaned" pixels that do not belong to the same connected component as their cluster center may remain. To correct for this, such pixels are assigned the label of the nearest cluster center using a connected components algorithm.

D. Complexity

By localizing the search in the clustering procedure, SLIC avoids performing thousands of redundant distance calculations. In practice, a pixel falls in the neighborhood of less than eight cluster centers, meaning that SLIC is $O(N)$ complex. In contrast, the trivial upper bound for the classical k-means algorithm is $O(k^N)$ [17], and the practical time complexity

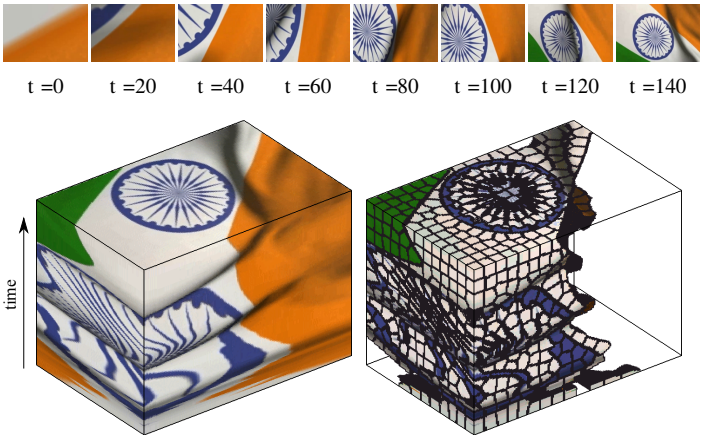


Fig. 3: SLIC supervoxels computed for a video sequence. *(top)* frames from a short video sequence of a flag waving. *(bottom left)* A volume containing the video. The last frame appears at the top of the volume. *(bottom right)* A supervoxel segmentation of the video. Supervoxels with orange cluster centers are removed for display purposes.

is $O(NkI)$ [6], where I is the number of iterations required for convergence. While schemes to reduce the complexity of k-means have been proposed using prime number length sampling [27], random sampling [13], local cluster swapping [12], and by setting lower and upper bounds [6], these methods are very general in nature. SLIC is specifically tailored to the problem of superpixel clustering. Finally, unlike most superpixel methods and the aforementioned approaches to speed up k-means, the complexity of SLIC is linear in the number of pixels, irrespective of k.

IV. Comparison with State-of-the-Art

We performed a quantitative comparison of SLIC and five state-of-the-art superpixel methods using publicly available source code. These algorithms include GS04[3], NC05[4], TP09[5], QS09[6], and two versions of the algorithm proposed in [26], GCa10 and GCb10[7]. Examples of superpixel segmentations produced by each method appear in Fig. 7.

A. Adherence to boundaries

Arguably, the most important property of a superpixel method is its ability to adhere to image boundaries. Boundary recall and under-segmentation error are standard measures for boundary adherence [15], [26]. In Fig. 4(a) and (b), SLIC, GS04, NC05, TP09, QS09, and GC10, are compared using these measures on the Berkeley database [20]. In addition, a baseline performance obtained by segmenting the image into uniform squares is denoted as "Squares". The Berkeley data set contains three-hundred 321×481 images, and approximately 10 human-annotated ground truth segmentations corresponding to each image.

Boundary recall measures what fraction of the ground truth edges fall within at least two pixels of a superpixel boundary. The boundary recall of each method is plotted in Fig. 4(a)

[3] http://people.cs.uchicago.edu/~pff/segment/
[4] http://www.cs.sfu.ca/~mori/research/superpixels/
[5] http://www.cs.toronto.edu/~babalex/turbopixels_supplementary.tar.gz
[6] http://www.vlfeat.org/download.html
[7] http://www.csd.uwo.ca/faculty/olga/Code/superpixels1pt1.zip

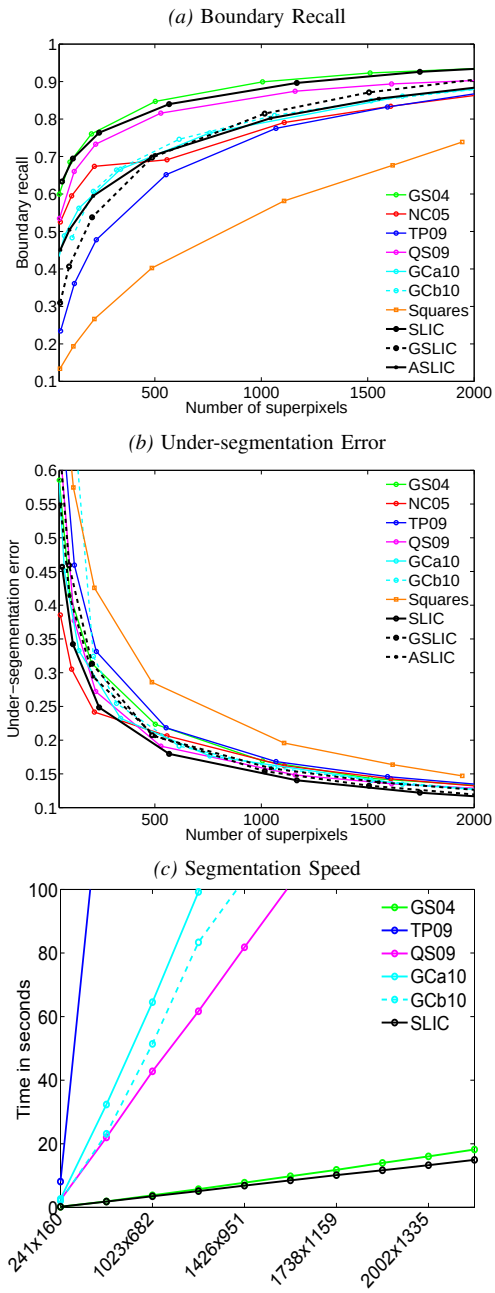


Fig. 4: Boundary adherence and segmentation speed. *(a)* Boundary recall measures the fraction of the ground truth edges that fall within at least two pixels of a superpixel boundary. While GS04 demonstrates the best boundary recall, reducing m from the default value increases the boundary recall of SLIC over that of GS04. *(b)* Under-segmentation error measures the amount of superpixel "leak" for a given ground truth region. SLIC outperforms the other methods, showing the lowest under-segmentation error for most of the useful operating regime. *(c)* Time required to generate superpixels for images of increasing size. SLIC is the fastest superpixel method, followed closely by GS04, and then a significant gap. NC05 is not plotted due to its particularly slow speed.

for increasing numbers of superpixels. A high boundary recall indicates that very few true edges were missed. Superpixels generated by SLIC and GS04 demonstrated the best boundary recall performance. If we reduce SLIC's compactness m from its default value of 10, SLIC shows superior performance to GS04.

Under-segmentation error, shown in Fig. 4(b), is another measure of boundary adherence. Given a region from the ground truth segmentation g_i and the set of superpixels required to cover it, $s_j | s_j \bigcap g_i$, it measures how many pixels from s_j "leak" across the boundary of g_i. If $|.|$ is the size of a segment in pixels, M is the number of ground truth segments, and B is a minimum number of pixels in s_j overlapping g_i, under-segmentation error is expressed as

$$U = \frac{1}{N}\left[\sum_{i=1}^{M}\left(\sum_{s_j | s_j \bigcap g_i > B} |s_j|\right) - N\right]. \quad (5)$$

B is set to 5% of $|s_j|$ in our experiments to account for ambiguities in the ground truth. Superpixels that do not tightly fit the ground truth result in a high value of U.

B. Computational and memory efficiency

Superpixels are often used to replace the pixel-grid to help speed up other algorithms. Thus, it is important that superpixels can be generated efficiently in the first place. In Fig. 4(c), we compare the time required for the various superpixel methods to segment images of increasing size on an Intel Dual Core 2.26 GHz processor with 2GB RAM. SLIC, with its $O(N)$ complexity, is the fastest superpixel method, and its advantage increases with the size of the image. While GS04 is competitive with $O(N)\log N$ complexity, the remaining methods show a significant gap in processing speed.

It is also important that a superpixel algorithm is memory efficient in order to handle large images. SLIC is the most memory efficient method, requiring only N floats to store the distance from each pixel to its nearest cluster center. Other methods have comparatively high memory requirements: GS04 and GC10 require $5N$ floats to store edge weights and thresholds for 4-connectivity (or $9N$ for 8-connectivity).

C. Segmentation performance

Superpixels are commonly used as a pre-processing step in segmentation algorithms. A good superpixel algorithm should improve the performance of the segmentation algorithm that uses it. We compared the segmentation resulting from SLIC, GS04, NC05, TP09, QS09, and GC10 on the MSRC data set [24]. These results were obtained using the method of [11], which uses superpixels to compute color, texture, geometry, and location features. It then trains classifiers for the 21 object classes and learns a CRF model. The results appearing in Table I show that SLIC superpixels yield the best performance. SLIC also reduces the the computational time by a factor of over 500 over NC05, the method used in [11]. Example images segmented using SLIC are shown in Fig. 5.

We also tested on the PASCAL VOC 2010 data set [7] using the approach of [10]. As shown in Table II, SLIC provided a boost in segmentation accuracy over QS09 and reduced the time spent generating superpixels by an order of magnitude.

D. Discussion

In addition to the properties discussed above, other considerations should factor into the quality of a superpixel algorithm. One such consideration is the ease of use. Superpixel methods with many difficult-to-tune parameters can result in lost time

TABLE II: Multi-class object segmentation on the PASCAL VOC 2010 data set. Results for the method of [10], using SLIC and QS09 superpixels

	background	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	dining table	dog	horse	motorbike	person	potted plant	sheep	sofa	train	TV/monitor	**Average**
SLIC	77.9	49.4	23.1	19.2	24.8	26.1	52.4	44.9	32.9	6.5	35.8	22.3	25.5	21.9	58.1	34.6	26.8	39.9	17.5	38.0	25.3	**33.5%**
QS09	78.1	45.0	23.3	18.3	25.0	25.5	52.3	45.6	33.2	7.2	36.0	21.5	24.7	21.9	56.9	34.4	26.0	38.9	17.0	37.9	24.8	33.0%

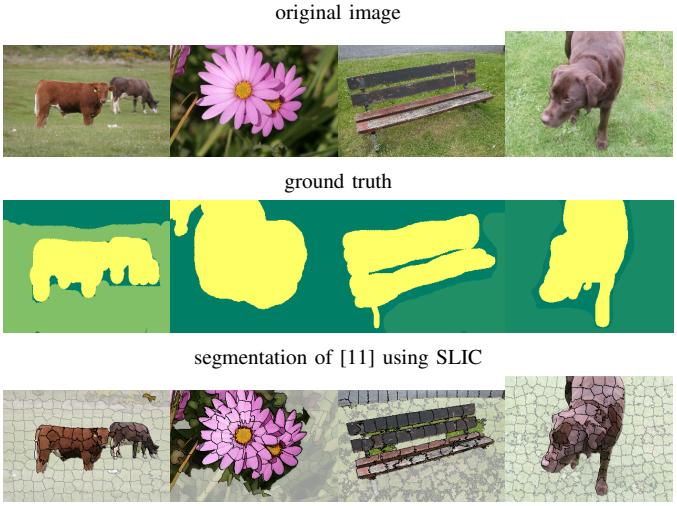


Fig. 5: Multi-class segmentation. *(top)* Images from the MSRC data set. *(middle)* Ground truth annotations. *(bottom)* Results obtained using SLIC superpixels in place of NC05, following the method proposed in [11].

or poor performance. Another consideration is the ability to specify the amount of superpixels, which not all methods provide. Finally, the ability to control the compactness of the superpixels is important. Compact, regular superpixels are often desirable because their bounded size and few neighbors form a more interpretable graph and can extract more locally relevant features. However, compactness comes at the expense of boundary adherence, and the ability to control this trade-off can be useful. In the following, we review the performance of each superpixel method with respect to boundary adherence, speed, memory efficiency, segmentation quality, parameter tuning, ability to specify the amount of superpixels, and ability to control superpixel compactness.

TP09 – while TP09 produced some of the most compact and consistently sized superpixels, it fared the worst among all methods in both boundary recall and under-segmentation error. TP09 also suffers from a slow running time, and resulted in poor segmentation performance. Next to NC05, it is the slowest superpixel algorithm, it is almost 100 times slower than SLIC for a 2048×1536 image, taking $800s$. On the other hand, TP09 has only 1 parameter to tune and offers direct control over the number of superpixels.

NC05 – Normalized cuts showed only a small improvement over TP09. The superpixels produced by NC05 are even more compact than those of TP09, making them attractive for graph-based applications. However, the boundary adherence is very poor, ranking 6^{th} in boundary recall and 5^{th} in under-segmentation error. Despite this, the segmentation quality was surprisingly high. The running time of NC05 is prohibitively slow, and the method failed to segment 2048×1536 images, producing "out of memory" errors.

GCa10 and **GCb10** – these two methods showed similar performance despite their differences in design (compact vs. constant intensity superpixels). GCb10's "compact" superpixels are more compact than GCa10, though much less than TP09 and NC05. In terms of boundary recall, GCa10 and GCb10 ranked in the middle of the pack, 5^{th} and 4^{th}, respectively. Their standing improved slightly for under-segmentation error (3^{rd} and 4^{th}). While GCa10 and GCb10 are faster than NC05 and TP09, their slow run-time still limits their usefulness (requiring $235s$ and $315s$, respectively), and they reported one of the worst segmentation performances. GC10 has 3 parameters to tune, including patch size, which can be difficult to set. On the positive side, GC10 allows for control of the number of superpixels, and can produce supervoxels.

QS09 – QuickShift performed well in terms of under-segmentation error and boundary recall, ranking 2^{nd} and 3^{rd} overall. However, QS09 showed relatively poor segmentation performance, and other limitations make it a less-than-ideal choice. It has a slow run-time ($181s$), requires several non-intuitive parameters to be tuned, and does not offer control over the amount or compactness of superpixels. Finally, the source code fails to ensure that superpixels are completely connected components, which can be problematic for subsequent processing.

GS04 – adheres well to image boundaries, although the superpixels are very irregular. It ranks 1^{st} in boundary recall, outperforming SLIC by a small margin. It is the second fastest method, segmenting a 2048×1536 image in $18.19s$ (without performing a parameter search). However, GS04 showed relatively poor segmentation performance and under-segmentation error, likely because its large, irregularly shaped superpixels are not suited to segmentation methods such as [11]. Lastly, GS04 does not allow the number of superpixels or compactness to be controlled with its three input parameters.

SLIC – Among the superpixel methods considered here, SLIC is clearly the best overall performer. It is the fastest method, segmenting a 2048×1536 image in $14.94s$, and most memory efficient. It boasts excellent boundary adherence, outperforming all other methods in under-segmentation error, and is second only to GS04 in boundary recall by a small margin (by adjusting m, it ranks first). When used for segmentation, SLIC showed the best boost in performance on the MSRC and PASCAL datasets. SLIC is simple to use, its sole parameter being the number of desired superpixels, and it is one of the few methods to produce supervoxels. Finally, among existing methods, SLIC is unique in its ability to control the trade-off between superpixel compactness and boundary adherence if desired, through m.

E. More complex distance measures

The reader may wonder if more sophisticated distance measures improve SLIC's performance, considering the simplicity of the approach described in Section III. We investigated this question by replacing the distance in Eq. 3 with an adaptively normalized distance measure (ASLIC) and a geodesic distance measure (GSLIC). Perhaps surprisingly, the simple distance measure used in Eq. 3 outperforms ASLIC and GSLIC in terms of speed, memory, and boundary adherence.

Adaptive-SLIC, or ASLIC, adapts the color and spatial normalizations within each cluster. As described in Section III-B, S and m in Eq. 3 are the assumed maximum spatial and color distances within a cluster. These constant values are used to normalize color and spatial proximity so they can be combined into a single distance measure for clustering. Instead of using constant values, ASLIC dynamically normalizes the proximities for each cluster using its maximum observed spatial and color distances (m_s, m_c) from the previous iteration. Thus, the distance measure becomes

$$D = \sqrt{\left(\frac{d_c}{m_c}\right)^2 + \left(\frac{d_s}{m_s}\right)^2}. \quad (6)$$

As before, constant normalization factors are used for the first iteration, but the algorithm subsequently keeps track of the maximum distances for each cluster. The advantage of this approach is that the superpixel compactness is more consistent, and it is never necessary to set m. This comes at the price of reduced boundary recall performance, as shown in Fig. 4(a).

Geodesic-SLIC, or GSLIC, replaces the distance of Eq. 3 with a geodesic distance. The unsigned geodesic distance from on pixel $\mathbf{I}(p_i)$ to another $\mathbf{I}(p_j)$ is defined as

$$\mathcal{G}(\mathbf{I}(p_i), \mathbf{I}(p_j)) = \min_{P \in \Gamma} d(P), \quad (7)$$

where Γ is the set of all paths between $\mathbf{I}(p_i)$ and $\mathbf{I}(p_j)$ and $d(P)$ is the cost associated to path P, given by

$$d(P) = \sum_{i=2}^{n} \|\mathbf{I}(p_i) - \mathbf{I}(p_{i-1})\|, \quad (8)$$

where $\|\mathbf{I}(p_i) - \mathbf{I}(p_{i-1})\|$ is the Euclidean distance between the CIELAB color vectors of pixels p_i and p_{i-1}. This approach has the advantage that the connectivity in the xy plane is guaranteed, eliminating the need for the post-processing step. However, the computation cost is higher and the boundary adherence performance suffers, as seen in Fig. 4(a).

V. Biomedical Applications

Many popular graph-based segmentation approaches such as graph cuts [3] become increasingly expensive as more nodes are added to the graph, limiting image size in practice. For some applications, such as mitochondria segmentation from electron micrographs (EM), the images are large but reducing the resolution is not an option. In such cases, segmentation on a graph defined over the pixel-grid would be intractable. In [18], SLIC superpixels significantly reduce the complexity of the graph, making the segmentation tractable. Segmented mitochondria from [18] are shown in Fig. 6(a) and (b).

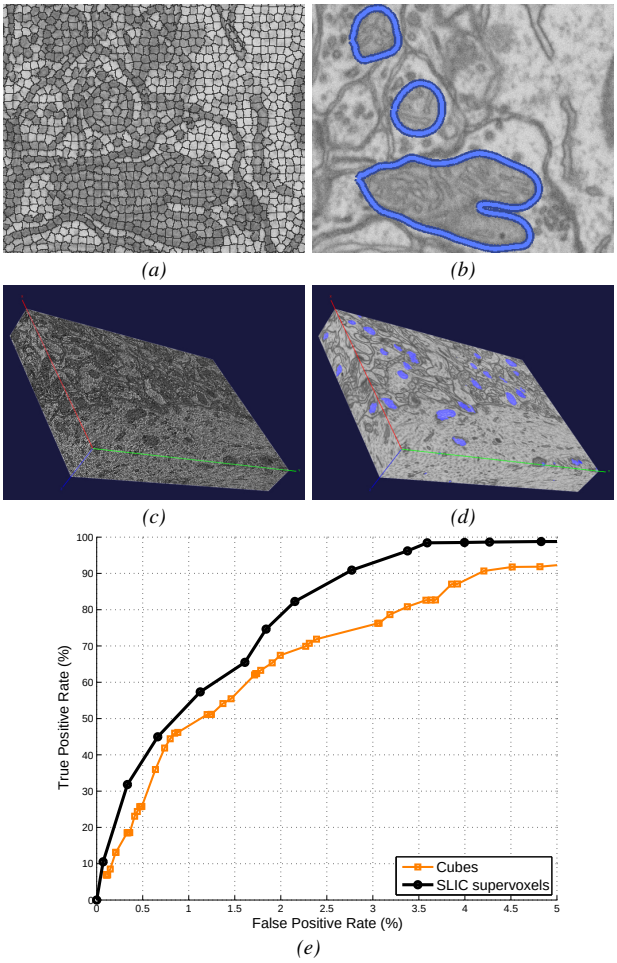


Fig. 6: SLIC applied to segment mitochondria from 2D and 3D EM images of neural tissue. *(a)* SLIC superpixels from an EM slice. *(b)* The segmentation result from the method of [18]. *(c)* SLIC supervoxels on a $1024 \times 1024 \times 600$ volume. *(d)* Mitochondria extracted using the method described in [19]. *(e)* Segmentation performance comparing SLIC supervoxels vs. cubes of similar size for the volume in *(c)*.

In [19], this approach is extended to 3D image stacks, which can contain *billions* of voxels. Only the most frugal of algorithms can operate on such large volumes of data without reducing the size of the graph in some manner. SLIC supervoxels reduce the memory requirements and complexity by over three orders of magnitude, and significantly increases performance over regular cubes as shown in Fig. 6(c)-(e).

VI. Conclusion

Superpixels have become an essential tool to the vision community, and in this paper we provide the reader with an in-depth performance analysis of modern superpixel techniques. We performed an empirical comparison of five state-of-the-art algorithms, concentrating on their boundary adherence, segmentation speed, and performance when used as a pre-processing step in a segmentation framework. In addition, we proposed a new method for generating superpixels based on k-means clustering, SLIC, which has been shown to outperform existing superpixel methods in nearly every respect.

Although our experiments are thorough, they come with a caveat. Certain superpixel methods, namely GC10 and TP09, do not consider color information, while the other methods do. This may adversely impact their performance.

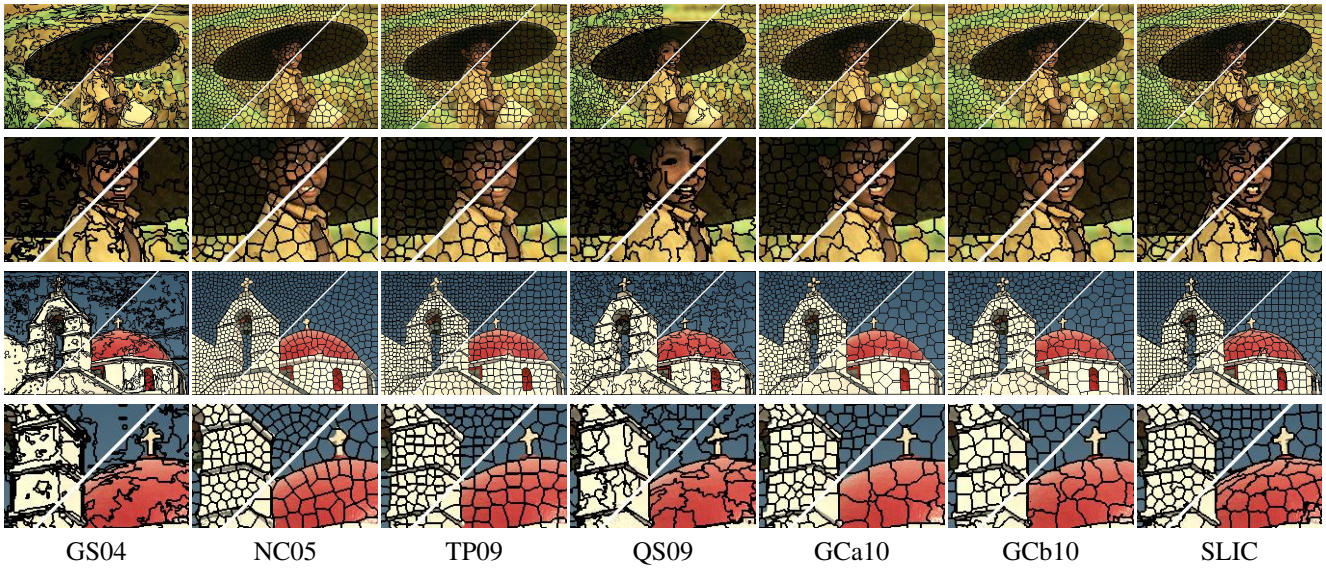


Fig. 7: Visual comparison of superpixels produced by various methods. The average superpixel size in the upper left of each image is 100 pixels, and 300 in the lower right. Alternating rows show each segmented image followed by a detail of the center of each image.

REFERENCES

[1] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizing. *ACM Transactions on Graphics (SIGGRAPH)*, 26(3), 2007.

[2] A. Ayvaci and S. Soatto. Motion segmentation with occlusions on the superpixel graph. In *Workshop on Dynamical Vision, Kyoto, Japan*, October 2009.

[3] Y. Boykov and M. Jolly. Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images. In *International Conference on Computer Vision (ICCV)*, 2001.

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

[5] T. Cour, F. Benezit, and J. Shi. Spectral segmentation with multiscale graph decomposition. In *IEEE Computer Vision and Pattern Recognition (CVPR) 2005*, 2005.

[6] Charles Elkan. Using the triangle inequality to accelerate k-means. *International Conference on Machine Learning*, 2003.

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge. *International Journal of Computer Vision (IJCV)*, 88(2):303–338, June 2010.

[8] Pedro Felzenszwalb and Daniel Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision (IJCV)*, 59(2):167–181, September 2004.

[9] B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *International Conference on Computer Vision (ICCV)*, 2009.

[10] J.M. Gonfaus, X. Boix, J. Weijer, A. Bagdanov, J. Serrat, and J. Gonzalez. Harmony Potentials for Joint Classification and Segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[11] Stephen Gould, Jim Rodgers, David Cohen, Gal Elidan, and Daphne Koller. Multi-class segmentation with relative location prior. *International Journal of Computer Vision (IJCV)*, 80(3):300–316, 2008.

[12] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. *Eighteenth annual symposium on Computational geometry*, pages 10–18, 2002.

[13] Amit Kumar, Yogish Sabharwal, and Sandeep Sen. A simple linear time (1+e)-approximation algorithm for k-means clustering in any dimensions. *Annual IEEE Symposium on Foundations of Computer Science*, 0:454–462, 2004.

[14] Vivek Kwatra, Arno Schodl, Irfan Essa, Greg Turk, and Aaron Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Transactions on Graphics, SIGGRAPH 2003*, 22(3):277–286, July 2003.

[15] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2009.

[16] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. *ACM Transactions on Graphics (SIGGRAPH)*, 23(3):303–308, 2004.

[17] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, IT-28(2):129–137, 1982.

[18] A. Lucchi, K. Smith, R. Achanta, V. Lepetit, and P. Fua. A fully automated approach to segmentation of irregularly shaped cellular structures in em images. *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2010.

[19] Aurélien Lucchi, Kevin Smith, Radhakrishna Achanta, Graham Knott, and Pascal Fua. Supervoxel-Based Segmentation of Mitochondria in EM Image Stacks with Learned Shape Features. *IEEE Transactions on Medical Imaging*, 30(11), 2011.

[20] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *IEEE International Conference on Computer Vision (ICCV)*, July 2001.

[21] Alastair Moore, Simon Prince, Jonathan Warrell, Umar Mohammed, and Graham Jones. Superpixel Lattices. *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2008.

[22] Greg Mori. Guiding model search using segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2005.

[23] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(8):888–905, Aug 2000.

[24] J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context. *International Journal of Computer Vision (IJCV)*, 81(1), January 2009.

[25] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In *European Conference on Computer Vision (ECCV)*, 2008.

[26] O. Veksler, Y. Boykov, and P. Mehrani. Superpixels and supervoxels in an energy optimization framework. In *European Conference on Computer Vision (ECCV)*, 2010.

[27] O. Verevka and J.W. Buchanan. Local k-means algorithm for color image quantization. *Graphics Interface*, pages 128–135, 1995.

[28] Luc Vincent and Pierre Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analalysis and Machine Intelligence*, 13(6):583–598, 1991.

[29] Y. Yang, S. Hallman, D. Ramanan, and C. Fawlkes. Layered Object Detection for Multi-Class Segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[30] C. L. Zitnick and S. B. Kang. Stereo for image-based rendering using image over-segmentation. *International Journal of Computer Vision (IJCV)*, 75:49–65, October 2007.