SLIC Superpixels*

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

School of Computer and Communication Sciences (IC)
École Polytechnique Fédrale de Lausanne (EPFL)

Abstract. Superpixels are becoming increasingly popular for use in computer vision applications. However, there are few algorithms that output a desired number of regular, compact superpixels with a low computational overhead. We introduce a novel algorithm that clusters pixels in the combined five-dimensional color and image plane space to efficiently generate compact, nearly uniform superpixels. The simplicity of our approach makes it extremely easy to use – a lone parameter specifies the number of superpixels – and the efficiency of the algorithm makes it very practical. Experiments show that our approach produces superpixels at a lower computational cost while achieving a segmentation quality equal to or greater than four state-of-the-art methods, as measured by boundary recall and under-segmentation error. We also demonstrate the benefits of our superpixel approach in contrast to existing methods for two tasks in which superpixels have already been shown to increase performance over pixel-based methods.

1 Introduction

Superpixels provide a convenient primitive from which to compute local image features. They capture redundancy in the image [1] and greatly reduce the complexity of subsequent image processing tasks. They have proved increasingly useful for applications such as depth estimation [2], image segmentation [3, 4], skeletonization [5], body model estimation [6], and object localization [7].

For superpixels to be useful they must be fast, easy to use, and produce high quality segmentations. Unfortunately, most state-of-the-art superpixel methods do not meet all these requirements. As we will demonstrate, they often suffer from a high computational cost, poor quality segmentation, inconsistent size and shape, or contain multiple difficult-to-tune parameters.

The approach we advocate in this work, while strikingly simple, addresses these issues and produces high quality, compact, nearly uniform superpixels more efficiently than state-of-the-art methods [8, 9, 5, 10]. The algorithm we propose, *simple linear iterative clustering* (SLIC) performs a local clustering of pixels in the 5-D space defined by the L, a, b values of the CIELAB color space and

* Please cite this paper as: Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk, SLIC Superpixels, EPFL Technical Report 149300, June 2010.

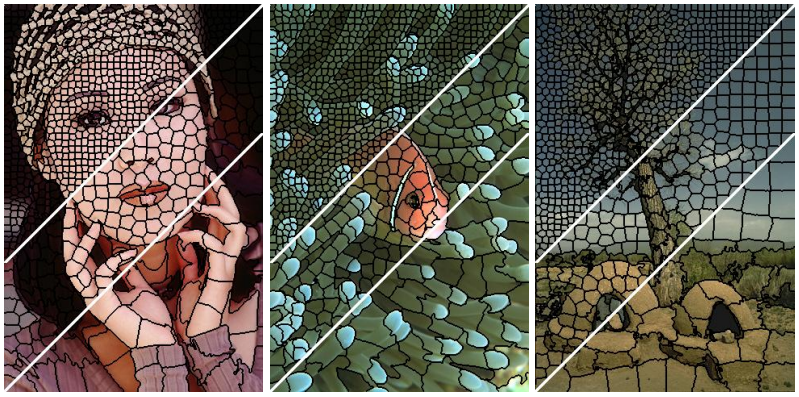


Fig. 1. Image segmented using our algorithm into superpixels of (approximate) size 64, 256, and 1024 pixels. The superpixels are compact, uniform in size, and adhere well to region boundaries.

the x, y pixel coordinates. A novel distance measure enforces compactness and regularity in the superpixel shapes, and seamlessly accomodates grayscale as well as color images. SLIC is simple to implement and easily applied in practice – the only parameter specifies the desired number of superpixels. Experiments on the Berkeley benchmark dataset [11] show that SLIC is significantly more efficient than competing methods, while producing segmentations of similar or better quality as measured by standard boundary recall and under-segmentation error measures.

For many vision tasks, compact and highly uniform superpixels that respect image boundaries, such as those generated by SLIC in Fig. 1, are desirable. For instance, graph-based models such as Conditional Random Fields (CRF) can see dramatic speed increases when switching from pixel-based graphs to superpixels [3, 7], but loose or irregular superpixels can degrade the performance. Local features such as SIFT extracted from the image at superpixel locations become less meaningful and discriminative if the superpixels are loose or irregular, and learning statistics over cliques of two or more superpixels can be unreliable. This effect can be seen when we compare the performance of SLIC superpixels to competing methods for two vision tasks: object class recognition and medical image segmentation. In both cases, our approach results in similar or greater performance at a lower computational cost in comparison to existing methods.

2 Background

In this section, we briefly review existing image segmentation algorithms and focus on their suitability for producing superpixels. Not all them were designed for this specific purpose and may lack the ability to control the size, number, and compactness of the segments, but we include them in our discussion nonetheless. We broadly classify superpixel algorithms into graph-based and gradient-ascent-based. Our survey, summarized in Table 1, considers the quality of segmentation, and the ability of these algorithm to control the size and number of superpixels.

Table 1. Comparison of state of the art superpixel segmentation algorithms. N is the number of pixels in the image. GS04 and QS09 do not offer explicit control of the number of superpixels. SL08 complexity given in this table does not take into account the complexity of the boundary map computation. GS04 is $O(NlogN)$ complex but is comparable in speed to SLIC for images less than 0.5 million pixels while TP09 is also $O(N)$ complex but is 10 times slower than SLIC for 481×321 pixel images. In the case of QS09, d is a small constant (refer to [10] for details). The number of parameters listed in the table is the minimum required for typical usage.

	Graph-based			Gradient-ascent-based				
Properties	GS04	NC05	SL08	WS91	MS02	TP09	QS09	SLIC
Superpixel no. ctrl.	No	Yes	Yes	No	No	Yes	No	Yes
Compactness ctrl.	No	Yes	Yes	No	No	Yes	No	Yes
Complexity $O(.)$	$NlogN$	$N^{3/2}$	N^2logN	$NlogN$	N^2	N	dN^2	N
Parameters	2	1	3	1	3	1	2	1

2.1 Graph-based algorithms

In graph based algorithms, each pixel is treated as a node in a graph, and edge weight between two nodes are set proportional to the similarity between the pixels. Superpixel segments are extracted by effectively minimizing a cost function defined on the graph.

The Normalized cuts algorithm [9], recursively partitions a given graph using contour and texture cues, thereby globally minimizing a cost function defined on the edges at the partition boundaries. It is the basis of the superpixel segmentation scheme of [1] and [6] (NC05). NC05 has a complexity of $O(N^{\frac{3}{2}})$ [12], where N is the number of pixels. There have been attempts to speed up the algorithm [13], but it remains computationally expensive for large images. The superpixels from NC05 have been used in body model estimation [6] and skeletonization [5].

Fezenszwalb and Huttenlocher [8] (GS04) present another graph-based segmentation scheme that has been used to generate superpixels. This algorithm performs an agglomerative clustering of pixel nodes on a graph, such that each segment, or superpixel, is the shortest spanning tree of the constituent pixels. GS04 has been used for depth estimation [2]. It is $O(NlogN)$ complex and is quite fast in practice as compared to NC05. However, unlike NC05, it does not offer an explicit control on the number of superpixels or their compactness.

A superpixel lattice is generated by [14] (SL08) by finding optimal vertical (horizontal) seams/paths that cut the image, within vertical (horizontal) strips of pixels, using graph cuts on strips of the image. While SL08 allows control of the size, number, and compactness of the superpixels, the quality and speed of the output strongly depend on pre-computed boundary maps.

2.2 Gradient-ascent-based algorithms

Starting from an initial rough clustering, during each iteration gradient ascent methods refine the clusters from the previous iteration to obtain better segmentation until convergence.

Mean-shift [15] (MS02) is a mode-seeking algorithm that generates image segments by recursively moving to the kernel smoothed centroid for every data point in the pixel feature space, effectively performing a gradient ascent [10]. The generated segments/superpixels can be large or small based on the input kernel parameters, but there is no direct control over the number, size, or compactness of the resulting superpixels.

Quick-shift [10] (QS08) is also a mode-seeking segmentation scheme like Mean-shift, but is faster in practice. It moves each point in the feature space to the nearest neighbor that increases the Parzen density estimate. The algorithm is non-iterative and, like Mean-shift, does not allow one to explicitly control the size or number of superpixels. Superpixels from quick-shift have been used in applications like object localization [7] and motion segmentation [16].

We include two other segmentation methods in the category of gradient ascent algorithms: Watersheds [17] (WS91) and Turbopixels [12] (TP09). General watershed algorithms perform gradient ascent from local minima in the image plane in order to obtain watersheds, i.e. lines that separate catchment basins. Vincent and Soille [17] propose a fast version based on queuing of pixels. *Lazy Snapping* [3] applies graph cuts to the graph built on the superpixels output by this algorithm.

TP09 generates superpixels by progressively dilating a given number of seeds in the image plane, using computationally efficient level-set based geometric flow. The geometric flow relies on local image gradients, and aims to distribute superpixels evenly on image plane. Unlike WS91, superpixels from TP09 are constrained to have uniform size, compactness, and adherence to object boundaries.

3 SLIC segmentation algorithm

Our approach generates superpixels by clustering pixels based on their color similarity and proximity in the image plane. This is done in the five-dimensional $[labxy]$ space, where $[lab]$ is the pixel color vector in CIELAB color space, which is widely considered as perceptually uniform for small color distances, and xy is the pixel position. While the maximum possible distance between two colors in the CIELAB space (assuming sRGB input images) is limited, the spatial distance in the xy plane depends on the image size. It is not possible to simply use the Euclidean distance in this 5D space without normalizing the spatial distances. In order to cluster pixels in this 5D space, we therefore introduce a new distance measure that considers superpixel size. Using it, we enforce color similarity as well as pixel proximity in this 5D space such that the expected cluster sizes and their spatial extent are approximately equal.

3.1 Distance measure

Our algorithm takes as input a desired number of approximately equally-sized superpixels K. For an image with N pixels, the approximate size of each superpixel is therefore N/K pixels. For roughly equally sized superpixels there would be a superpixel center at every grid interval $S = \sqrt{N/K}$.

At the onset of our algorithm, we choose K superpixel cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ with $k = [1, K]$ at regular grid intervals S. Since the spatial extent of any superpixel is approximately S^2 (the approximate area of a superpixel), we can safely assume that pixels that are associated with this cluster center lie within a $2S \times 2S$ area around the superpixel center on the xy plane. This becomes the search area for the pixels nearest to each cluster center.

Euclidean distances in CIELAB color space are perceptually meaningful for small distances (m in Eq. 1). If spatial pixel distances exceed this perceptual color distance limit, then they begin to outweigh pixel color similarities (resulting in superpixels that do not respect region boundaries, only proximity in the image plane). Therefore, instead of using a simple Euclidean norm in the 5D space, we use a distance measure D_s defined as follows:

$$\begin{aligned} d_{lab} &= \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2} \\ d_{xy} &= \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2} \\ D_s &= d_{lab} + \frac{m}{S} d_{xy} \ , \end{aligned} \tag{1}$$

where D_s is the sum of the *lab* distance and the xy plane distance *normalized* by the grid interval S. A variable m is introduced in D_s allowing us to control the compactness of a superpixel. The greater the value of m, the more spatial proximity is emphasized and the more compact the cluster. This value can be in the range $[1, 20]$. We choose $m = 10$ for all the results in this paper. This roughly matches the empirical maximum perceptually meaningful CIELAB distance and offers a good balance between color similarity and spatial proximity.

3.2 Algorithm

The *simple linear iterative clustering* algorithm is summarized in Algorithm 1. We begin by sampling K regularly spaced cluster centers and moving them to seed locations corresponding to the lowest gradient position in a 3×3 neighborhood. This is done to avoid placing them at an edge and to reduce the chances of choosing a noisy pixel. Image gradients are computed as:

$$G(x, y) = \|\mathbf{I}(x+1, y) - \mathbf{I}(x-1, y)\|^2 + \|\mathbf{I}(x, y+1) - \mathbf{I}(x, y-1)\|^2 \tag{2}$$

where $\mathbf{I}(x, y)$ is the *lab* vector corresponding to the pixel at position (x, y), and $\|.\|$ is the L_2 norm. This takes into account both color and intensity information.

Each pixel in the image is associated with the nearest cluster center whose search area overlaps this pixel. After all the pixels are associated with the nearest cluster center, a new center is computed as the average *labxy* vector of all

the pixels belonging to the cluster. We then iteratively repeat the process of associating pixels with the nearest cluster center and recomputing the cluster center until convergence.

At the end of this process, a few stray labels may remain, that is, a few pixels in the vicinity of a larger segment having the same label but not connected to it. While it is rare, this may arise despite the spatial proximity measure since our clustering does not explicitly enforce connectivity. Nevertheless, we enforce connectivity in the last step of our algorithm by relabeling disjoint segments with the labels of the largest neighboring cluster. This step is $O(N)$ complex and takes less than 10% of the total time required for segmenting an image.

Algorithm 1 Efficient superpixel segmentation

1: Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S.
2: Perturb cluster centers in an $n \times n$ neighborhood, to the lowest gradient position.
3: **repeat**
4: **for** each cluster center C_k **do**
5: Assign the best matching pixels from a $2S \times 2S$ square neighborhood around the cluster center according to the distance measure (Eq. 1).
6: **end for**
7: Compute new cluster centers and residual error E {$L1$ distance between previous centers and recomputed centers}
8: **until** $E \leq$ threshold
9: Enforce connectivity.

3.3 Complexity

It is easy to notice that the idea of iteratively evolving local clusters and cluster centers used in our algorithm is a special case of *k-means* adapted to the task of generating superpixels. Interestingly, by virtue of using our distance measure of Eq. (1), we are able to *localize* our pixel search to an area ($2S \times 2S$) on the image plane that is inversely proportional to the number of superpixels K. In practice, a pixel falls in the local neighborhood of no more than eight cluster centers. We also note that the convergence error of our algorithm drops sharply in a few iterations. Our experiments show that it suffices to run the algorithm for 4 to 10 iterations. We use 10 iterations for all the results in this paper.

The time complexity for the classical *k-means* algorithm is $O(NKI)$ where N is the number of data points (pixels in the image), K is the number of clusters (or seeds), and I is the number of iterations required for convergence. Our method achieves $O(N)$ complexity (see Fig. 4), since we need to compute distances from any point to no more than eight cluster centers and the number of iterations is constant. Thus, SLIC is specific to the problem of superpixel segmentation, and unlike *k-means*, avoids several redundant distance calculations.

Speed-up schemes for the *k-means* algorithm have been proposed using prime number length sampling [18], random sampling [19], by local cluster swap-

ping [20], and by setting lower and upper bounds [21]. However except for [21], these methods do no achieve linear complexity for a given K. SLIC, on the other hand, is linear in the number of pixels irrespective of K. Note that, like [21], SLIC implicitly sets bounds on distance calculations, but does not need to compute or carry forward these bounds for the subsequent iterations. Unlike most of these segmentation schemes, which are very general in nature, SLIC is specifically tailored to perform superpixel clustering using the distance measure of Eq. (1) and is much simpler.

4 Comparison

In this section we compare our superpixel segmentation method with four state-of-the-art algorithms, namely, GS04[1] [8], NC05[2] [6], TP09[3] [12], QS09[4] [7] using publicly available source codes. GS04 and NC05 are graph based methods while TP09 and QS09 are gradient based approaches. NC05 and TP09 are designed to output a desired number of superpixels while GS04 and QS09 require parameter tuning to obtain a desired number of superpixels. This choice of algorithms should provide a good representation of the state-of-the-art.

Fig. 2 provides a visual comparison of our output against these algorithms. To provide a qualitative comparison, we use the under-segmentation error and boundary recall measures, similar to the ones used by Levenshtein et al. [12] for this purpose, computed using the Berkeley segmentation ground truth [11].

4.1 Algorithm Parameters

As mentioned in the introduction, it is important for superpixel algorithms to be easy to use. Difficult-to-set parameters can result in lost time or poor performance. Table 1 summarizes the number of parameters that must be tuned or set for each method. SLIC, like NC05 and TP09 requires a single parameter. It is also important to note that GS04 and QS09 do not allow the user to control the number of superpixels. We had to perform a search on the parameter space to be able to control the number of superpixels in order to make a fair comparison to the other methods.

4.2 Under-segmentation error

Under-segmentation error essentially measures the error an algorithm makes in segmenting an image with respect to a known ground truth (human segmented images in this case). This error is computed in terms of the 'bleeding' of the segments output by an algorithm when placed over ground truth segments. This

[1] `http://people.cs.uchicago.edu/~pff/segment/`
[2] `http://www.cs.sfu.ca/~mori/research/superpixels/`
[3] `http://www.cs.toronto.edu/~babalex/turbopixels_supplementary.tar.gz`
[4] `http://www.vlfeat.org/download.html`

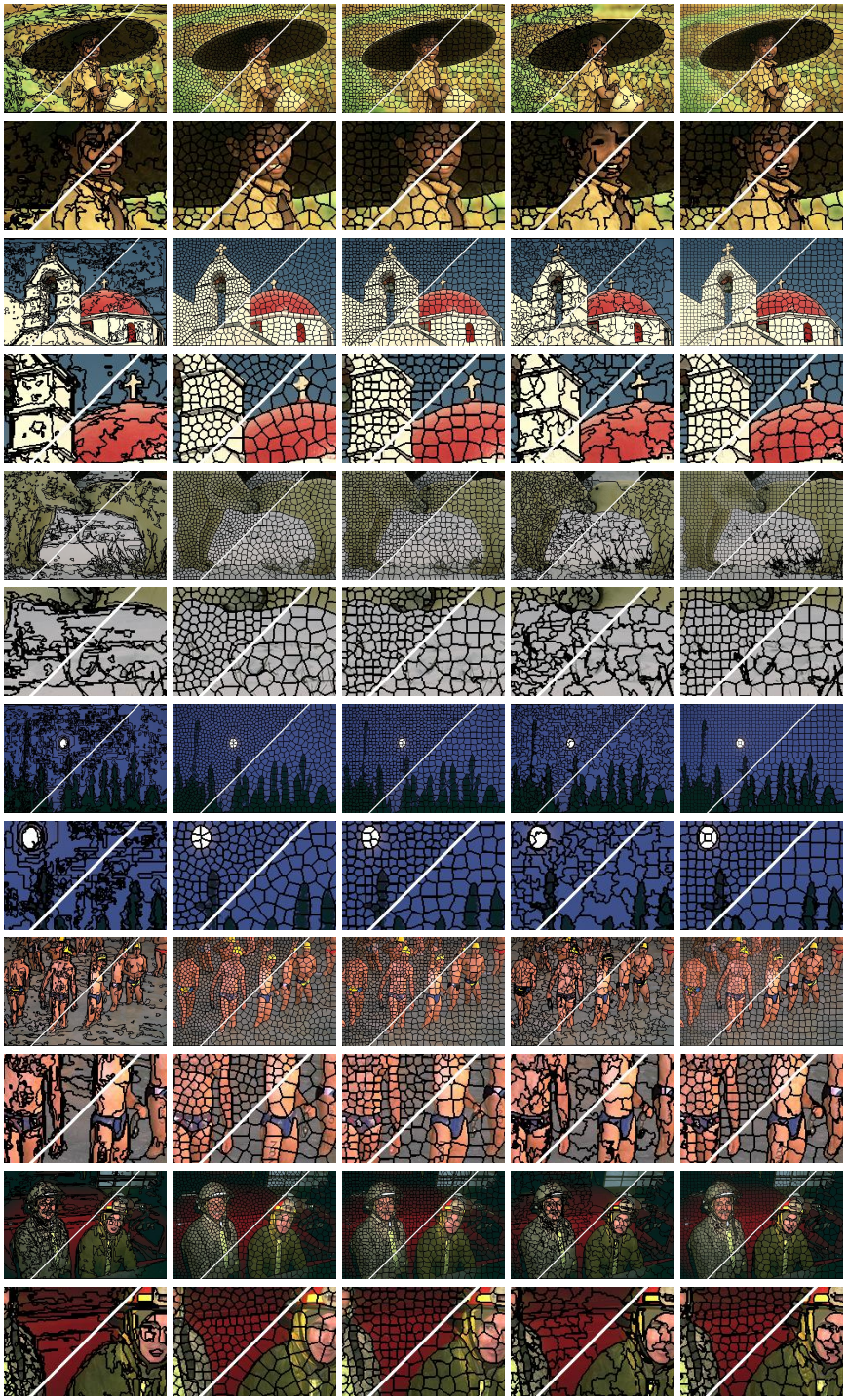
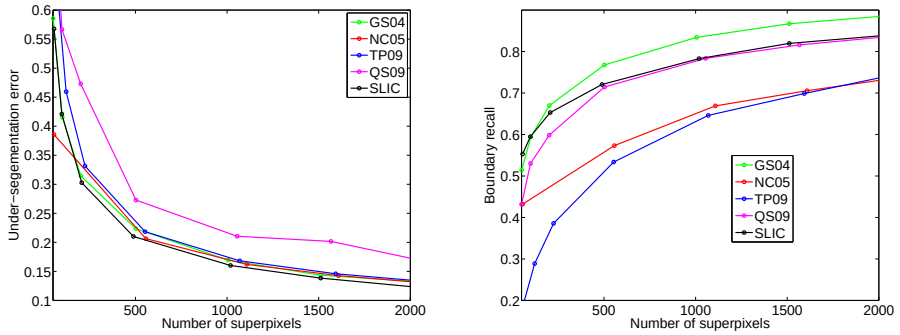


Fig. 2. Visual comparison of the superpixels. The average superpixel size in the two halves in all images is roughly 100 pixels and 300 pixels each. Each pair of rows show the whole segmented image and its central part blown-up respectively.



(a) (b)

Fig. 3. (a) Plot of the under-segmentation error w.r.t. number of superpixels. (b) Plot of the boundary recall w.r.t. number of superpixels. The output of NC05 is visually the most appealing but its boundary recall is quite poor. GS04 has a higher boundary recall than all algorithms, including ours, but this is partly because of the fact that it places a lot of segments in the vicinity of object boundaries.

measure thus penalizes superpixels that do not tightly fit a ground truth segment boundary.

Given ground truth segments $g_1, g_2, ..., g_M$ and a superpixel output $s_1, s_2, ..., s_L$, the under-segmentation error for a ground truth segment g_i is quantified as:

$$U = \frac{1}{N}\left[\sum_{i=1}^{M}\left(\sum_{[s_j|s_j \bigcap g_i > B]}|s_j|\right) - N\right] \tag{3}$$

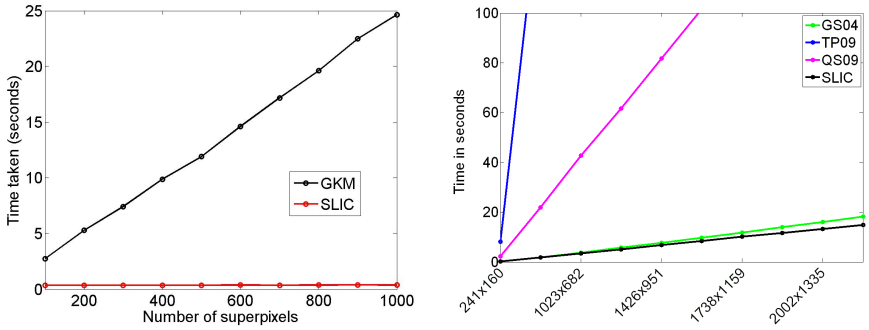
where $|.|$ gives the size of the segment in pixels, N is the size of the image in pixels, and B is the minimum number of pixels that need to be overlapping. The expression $s_j \bigcap g_i$ is the intersection or overlap error of a superpixel s_j with respect to a ground truth segment g_i. B is set to be 5% of $|s_j|$ to account for small errors in ground truth segmentation data. The value of U is computed for each image of the ground truth and then averaged to obtain the graph in Fig. 3(a).

4.3 Boundary recall

We adopt the standard boundary recall measure which computes what fraction of ground truth edges fall within one pixel of a least one superpixel boundary. We use the internal boundaries of each superpixel. The boundary recall of each of the considered methods is plotted against the number of superpixels in Fig. 3(b).

4.4 Computational and memory efficiency

For images of size 480×320, SLIC is more than 10 times faster than TP09 and more than 500 times faster than NC05. What is encouraging is that it is even



(a) (b)

Fig. 4. (a) Plot of the time taken for 10 iterations of *k-means* (GKM) versus our algorithm for different number of superpixels on input images of size 481×321. Our algorithm takes less than 0.5 second for any superpixel sizes. (b) Plot of the segmentation time taken (in seconds) versus image size in pixels.

faster than GS04 for images greater than half a million pixels (see Fig. 4(b)). This is because our algorithm always operates at $O(N)$ complexity while GS04 has $O(NlogN)$ complexity. This is of interest because even low end consumer digital cameras produce images exceeding 3 million pixels. Also, GS04 requires $5 \times N$ memory to store edge weights and thresholds, as opposed to SLIC, which needs $1 \times N$ memory (to store the distance of each pixel from its nearest cluster center).

4.5 Discussion

A good superpixel segmentation algorithm should have low under-segmentation error as well as high boundary recall. To be useful as a pre-processing algorithm, such a segmentation should result in equally sized compact superpixels with control on its number. For the same reason, the algorithm should preferably also have low computational cost and require few input parameters. Fig. 3(b) shows that the highest boundary recall is achieved by GS04. This is because it produces several segments close to object boundaries as seen in Fig. 2(a). However, GS04 also exhibits higher under-segmentation error than our algorithm, as can be seen in Fig. 3(a). Our algorithm actually shows the lowest under-segmentation error in Fig. 3(a) as well as high boundary recall in Fig. 3(b) next only to GS04. Note that GS04, however, is not meant to output a desired number of superpixels of a given size unless a parameter search is performed, which requires several runs of the algorithm. Even then the superpixel sizes are unequal, making the algorithm far less suitable for superpixel-based applications [5–7]. In addition, as seen in Fig. 4(b) our algorithm is faster than the compared state-of-the-art algorithms for any image size (including GS04 for a single run) and it outputs the desired number of equally-sized compact superpixels.

5 Superpixel Applications

Operating on superpixels instead of pixels can speed up existing pixel-based algorithms, and even improve results in some cases [7]. For instance, certain graph-based algorithms can see a 2 to 3-fold speed increase using superpixels [3]. Of course, the superpixel generation itself should be fast for this to be practical.

Below, we consider two typical vision tasks that benefit from using superpixels: object class recognition and medical image segmentation. In each case, superpixels have been shown to increase the performance of an existing algorithm while reducing computational cost. We show that SLIC superpixels outperform state-of-the-art superpixel methods on these tasks, but with a lower computational cost.

5.1 Object class recognition

Our first task is to perform object class recognition for 21 object classes from the STAIR vision library[5] based on the work of Gould et al [22]. Color, texture, geometry, and location features are computed for each superpixel region. Then boosted classifiers are learned using these features for each region class. Finally, a Conditional Random Field (CRF) model is learned using the output of the boosted classifiers as features. In the original work [22], NC05 is used to segment each image (of size 320×240 pixels) into about 200 superpixels. By applying SLIC superpixels instead of NC05, the classification accuracy increases, as shown in Table 2, while the computational cost is reduced by a factor of 400.

Table 2. Object class recognition for various superpixel methods.

	GS04	NC05	TP09	QS09	SLIC
Pixelwise accuracy	74.6%	75.9%	75.1%	62.0%	**76.9%**

5.2 Medical image segmentation

Our second task is to assist recent efforts towards a 'bottom up' understanding of brain function by segmenting mitochondria from neural electron microscopy (EM) images. Neuroscientists attempting to reconstruct neural structures at an extremely fine level of detail must typically perform a painstaking manual analysis on such data. Modern segmentation algorithms such as [7] can efficiently automate this process by taking advantage of superpixels. Below, we compare mitochondrial segmentations using an approach based on [7] for various types of superpixels including GS04, TP09, QS09, and SLIC. NC05 is omitted because its computational cost is impractical for such high resolution images.

The first step of the approach is to perform a superpixel over-segmentation of the image, and define a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ corresponding to the superpixels. QS09 is used to generate superpixels in [7]. Each node in $\mathcal{V}$ corresponds to a

[5] `http://ai.stanford.edu/~sgould/svl`

superpixel x_i. Edges $\mathcal{E}$ connect neighboring superpixels. Then, SIFT descriptors are extracted at center of each superpixel at various scales and a fixed orientation.

The segmentation is performed using graph-cuts, which partitions the graph into disjoint partitions by minimizing an objective function of the form

$$E(c|x,w) = \sum_i \underbrace{\psi(c_i|x_i)}_{\text{unary term}} + w \sum_{(i,j)\in\mathcal{E}} \underbrace{\phi(c_i,c_j|x_i,x_j)}_{\text{pairwise term}} \ , \tag{4}$$

where $c_i \in \{mitochondria, background\}$ and the weight w controls the relative importance of the so-called *unary* and *pairwise* terms. The *unary* term ψ assigns to each superpixel its potential to be mitochondria or background based on a probability $P(c_i|\mathbf{f}(x_i))$ computed from the output of a support vector machine (SVM) classifier trained using the SIFT descriptors. The *pairwise* term ϕ assigns to each pair of superpixels a potential to have similar or differing labels based on the difference of intensities between the two superpixels,

$$\psi(c_i|x_i) = \tfrac{1}{1+P(c_i|\mathbf{f}(x_i))}, \quad \phi(c_i,c_j|x_i,x_j) = \begin{cases} \exp\left(-\frac{||I(x_i)-I(x_j)||^2}{2\sigma^2}\right) & \text{if } c_i \neq c_j \\ 0 & \text{otherwise.} \end{cases}$$

Eq. 4 is minimized using a mincut-maxflow algorithm to produce a final segmentation, assigning a label to each superpixel.

Segmentations obtained for each superpixel method were compared over a set of 23 annotated EM images of 2048×1536 resolution, containing 1023 mitochondria. We used $k = 5$ k-folds cross validation for testing and training. Results are given in Table 3, and example images appear in Fig. 5. The VOC score $= \frac{TP}{TP+FP+FN}$ is used to evaluate segmentation performance, as it is more informative than pixelwise accuracy for sparse objects such as mitochondria [6].

Table 3. Mitochondria segmentation results for various superpixel methods.

	GS04	TP09	QS09	SLIC
VOC score	65.3%	58.9%	66.6%	**67.3%**

The advantages of the SLIC superpixel are demonstrated in the examples appearing in column 5 of Fig. 5. Features extracted over the regular, compact SLIC superpixels tend to be more discriminative, helping the graph-cut to produce better segmentations. The good adherence to image boundaries exhibited by SLIC superpixels result in smoother and more accurate segmentations. We can also see the drawbacks of other superpixel methods by considering the examples in columns 2 through 4 of Fig. 5. The irregularity of GS04 superpixels in column 2 makes the extracted features less discriminative, often causing the segmentation to fail. TP09, in column 3, performs the worst of the four methods. Because the intensity gradients in the EM images are not particularly strong,

[6] TP=true positives, FP=false positives, FN=false negatives

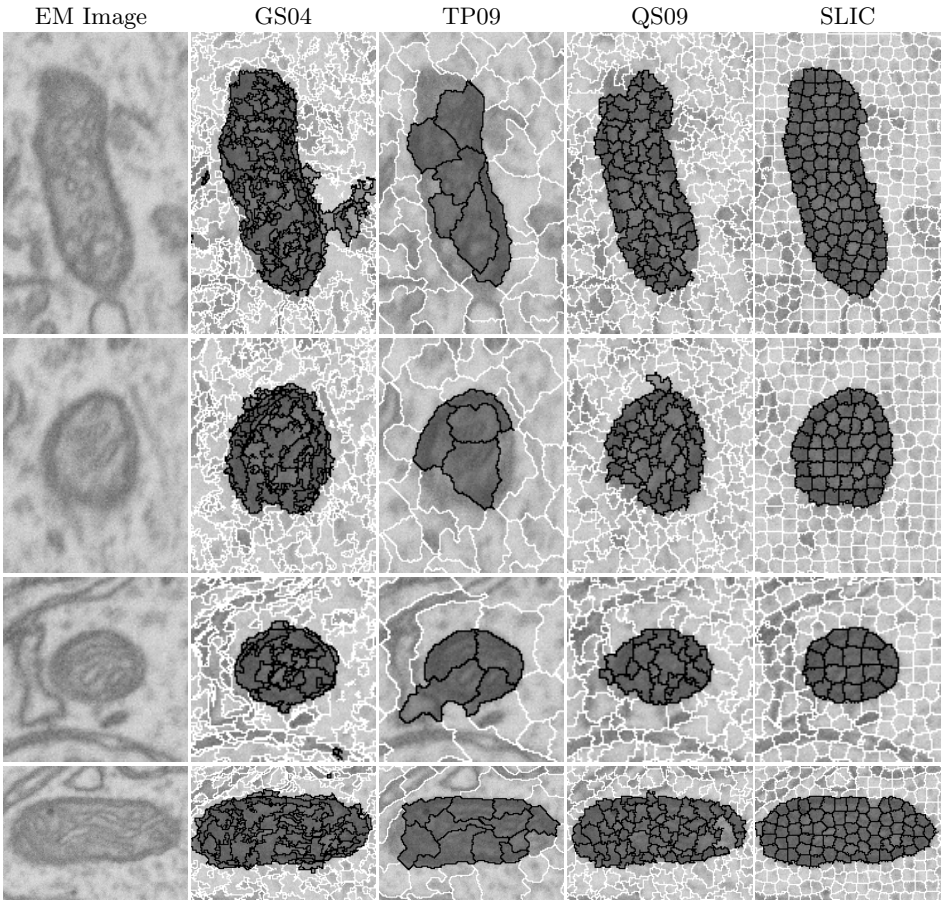


Fig. 5. Segmentation results on EM imagery. Examples segmentations for GS04, TP09, QS09, and SLIC are cropped from 2048×1536 micrographs. Regular, compact superpixels generated by SLIC exhibit good boundary adherence and produce the best segmentation results. The large superpixels produced by TS09 are a result of the algorithm merging small superpixels over relatively weak intensity gradients in the EM image, despite parameters being set for smaller superpixels. See text for further discussion.

TP09 tends to merge smaller superpixels, causing issues in the segmentation. In column 4, the superpixels of QS09 appear most similar to SLIC, but still result in numerous segmentation failures where they do not respect mitochondrial boundaries as well as SLIC.

6 Conclusions

Superpixel segmentation algorithms can be very useful as a preprocessing step for computer vision applications like object class recognition and medical image segmentation. To be useful, such algorithms should output high quality superpixels

that are compact and roughly equally sized, for a low computational overhead. There are few superpixel algorithms that can offer this and scale up for practical applications that deal with images greater than 0.5 million pixels. We present a novel $O(N)$ complexity superpixel segmentation algorithm that is simple to implement and outputs better quality superpixels for a very low computational and memory cost. It needs only the number of desired superpixels as the input parameter. It scales up linearly in computational cost and memory usage. We prove the efficacy of our superpixels in object category recognition and medical image segmentation, where compared to other state-of-the art algorithms, we obtain better quality and higher computational efficiency.

7 Acknowledgements

This work is supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

References

1. Ren, X., Malik, J.: Learning a classification model for segmentation. ICCV (2003) 10–17
2. Hoiem, D., Efros, A., Hebert, M.: Automatic photo pop-up. SIGGRAPH (2005)
3. Li, Y., Sun, J., Tang, C.K., Shum, H.Y.: Lazy snapping. SIGGRAPH (2004) 303–308
4. He, X., Zemel, R., Ray, D.: Learning and incorporating top-down cues in image segmentation. ECCV (2006) 338–351
5. Levinshtein, A., Sminchisescu, C., Dickinson, S.: Multiscale symmetric part detection and grouping. ICCV (2009)
6. Mori, G.: Guiding model search using segmentation. ICCV (2005) 1417–1423
7. Fulkerson, B., Vedaldi, A., Soatto, S.: Class segmentation and object localization with superpixel neighborhoods. ICCV (2009)
8. Felzenszwalb, P., Huttenlocher, D.: Efficient graph-based image segmentation. IJCV (2004) 167–181
9. Shi, J., Malik., J.: Normalized cuts and image segmentation. PAMI (2000) 888–905
10. Vedaldi, A., Soatto, S.: Quick shift and kernel methods for mode seeking. ECCV (2008)
11. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. ICCV (2001) 416–423
12. Levinshtein, A., Stere, A., Kutulakos, K., Fleet, D., Dickinson, S., Siddiqi, K.: Turbopixels: Fast superpixels using geometric flows. PAMI (2009)
13. Cour, T., Benezit, F., Shi, J.: Spectral segmentation with multiscale graph decomposition. CVPR (2005) 1124–1131
14. Moore, A., Prince, S., Warrell, J., Mohammed, U., Jones, G.: Superpixel Lattices. CVPR (2008)

15. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. PAMI (2002) 603–619
16. Ayvaci, A., Soatto, S.: Motion segmentation with occlusions on the superpixel graph. Proc. of the Workshop on Dynamical Vision, Kyoto, Japan. (2009)
17. Vincent, L., Soille, P.: Watersheds in digital spaces: An efficient algorithm based on immersion simulations. PAMI (1991) 583–598
18. Verevka, O., Buchanan, J.: Local k-means algorithm for color image quantization. Proceedings of Graphics Interface (1995) 128–135
19. Kumar, A., Sabharwal, Y., Sen, S.: A simple linear time (1+e)-approximation algorithm for k-means clustering in any dimensions. IEEE Symposium on Foundations of Computer Science **0** (2004) 454–462
20. Kanungo, T., Mount, D.M., Netanyahu, N.S., Piatko, C.D., Silverman, R., Wu, A.Y.: A local search approximation algorithm for k-means clustering. Symposium on Computational geometry. (2002) 10–18
21. Elkan, C.: Using the triangle inequality to accelerate k-means. International Conference on Machine Learning. (2003)
22. Gould, S., Rodgers, J., Cohen, D., Elidan, G., Koller, D.: Multi-class segmentation with relative location prior. IJCV (2008) 300–316