SMURFS: Superpixels from Multi-scale Refinement of Super-regions

Imanol Luengo[1]
imanol.luengo@nottingham.ac.uk
Mark Basham[2]
mark.basham@diamond.ac.uk
Andrew P. French[1]
andrew.p.french@nottingham.ac.uk

[1] School of Computer Science, University of Nottingham, Nottingham, UK, NG8 1BB

[2] Diamond Light Source Ltd, Harwell Science & Innovation Campus, Didcot, UK, OX11 0DE

Abstract

Recent applications in computer vision have come to rely on superpixel segmentation as a pre-processing step for higher level vision tasks, such as object recognition, scene labelling or image segmentation. Here, we present a new algorithm, Superpixels from MUlti-scale ReFinement of Super-regions (SMURFS), which not only obtains state-of-the-art superpixels, but can also be applied hierarchically to form what we call *n*-th order super-regions. In essence, starting from a uniformly distributed set of super-regions, the algorithm iteratively alternates graph-based split and merge optimization schemes which yield superpixels that better represent the image. The split step is performed over the pixel grid to separate large super-regions into more discriminative smaller superpixels. The merging process, conversely, is performed over the superpixel graph to create 2nd-order super-regions (*super-segments*). Iterative refinement over the two-scale regions allows the algorithm to achieve better over-segmentation results than current state-of-the-art methods, as experimental results show on the public Berkeley Segmentation Dataset (BSD500).

1 Introduction

There is an increasing trend to use superpixels as building blocks for many computer vision applications such as image segmentation [11], image parsing [14], semantic labelling [6] and object detection and tracking [19]. Superpixel algorithms group pixels into perceptually meaningful regions, which are more aligned with the human visual cognition system. They not only reduce the redundancy and noise effects in the standard pixel grid, but are also especially useful for high computational cost problems, as operating over a superpixel graph reduces dimensionality (and thus computational complexity) by several orders of magnitude. To enable this reduction in resolution to be helpful, there are some generally accepted properties that a superpixel algorithm should offer in order to provide quality results in the subsequent higher-level applications: (1) Superpixels should adhere to image boundaries. (2) Each superpixel should be contained in a unique higher level object. That's is, a superpixel should not overlap more than one object in the image. And (3), in most applications superpixels are used as a preprocessing step; therefore they should be fast to compute and memory efficient.

Pages 4.1-4.12
DOI: https://dx.doi.org/10.5244/C.30.4

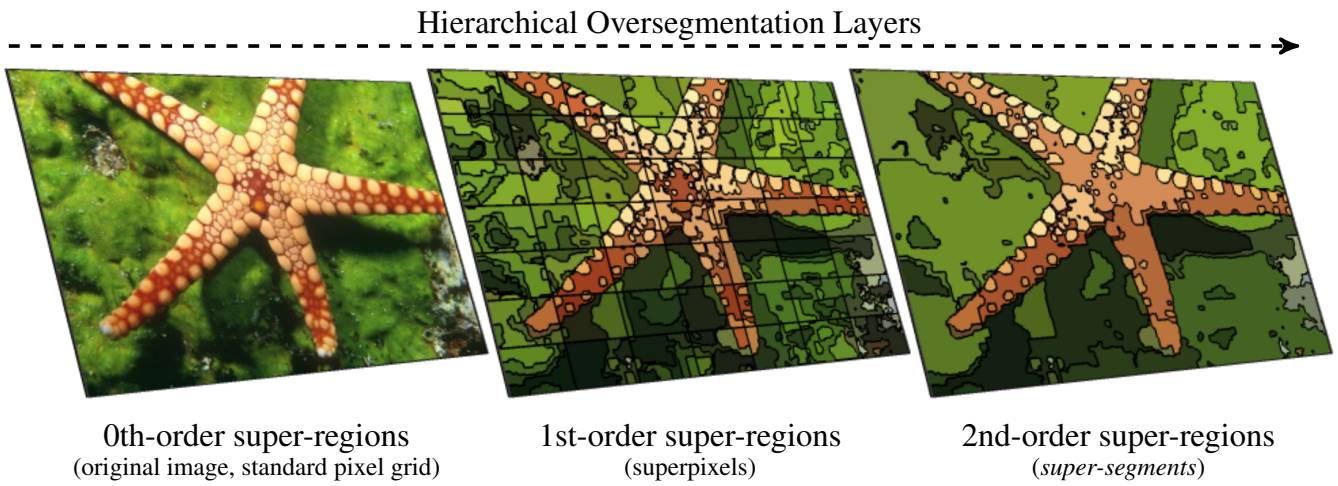


Figure 1: Overview of the hierarchical regions, where 0th-order super-regions correspond to the standard pixel grid. Each layer of the hierarchy outputs a decreasing number of regions while maintaining image representation capabilities.

Most of the state-of-the-art algorithms [1][9][15][8][7] extract superpixels by adding cuts to a graph, growing superpixels from a predefined set of seeds. In SEEDS [15], the authors adopt a different idea. Rather than starting with seeds or a connected graph and iteratively modifying the current state until superpixels are created, the authors propose to iteratively refine an initial set of superpixel boundaries. Starting by a complete superpixel partition (uniformly distributed blocks), the superpixels are iteratively refined by moving their boundaries at pixel or block levels.

Here we present an alternative idea that consists of alternating moves at different hierarchical segmentation layers. We present a new superpixel over-segmentation framework, SMURFS, which uses a multi-scale region representation of the image to iteratively refine super-regions at different scales. We define a multi-scale region representation by what we term *n*-order hierarchical super-regions, which can be seen as hierarchical segmentation layers. Each layer of the super-region is composed by grouping elements of the previous layer, and thus, yielding increasingly larger segments while maintaining boundary adherence and image representation properties. The 0th-order super-region corresponds to the standard pixel grid, 1st-order super-regions represent common superpixels, made of groups of similar pixels, and the 2nd-order super-regions are essentially superpixels of superpixels (first introduced as *super-segments* in [12]). Figure 1 overviews this process. Our main contribution is an algorithm to iteratively create and refine super-regions at both scales to provide superpixels with increasing image modelling capabilities. Results in the BSD500 dataset [10] show that by alternating simple formulations at two over-segmentation layers, our algorithm is able to better capture global and local image properties and obtain superpixels with the highest Achievable Segmentation Accuracy (ASA), compared to recent state of the art.

2 Related work

We split the previous existing algorithms into three different categories: superpixels from a graph formulation by gradually adding cuts, superpixels grown from initialized seeds, and superpixels extracted by moving a predefined set of boundaries.

Graph-based methods represent the image as a graph of pixels in a 4 or 8-neighbouring system and calculate similarities between adjacent pixels. For example, Normalized Cuts [13] globally minimizes an objective function by recursively finding the optimal partition in the normalized Laplacian graph. While giving good results, the algorithm is computationally expensive. An alternative approach is an agglomerative clustering algorithm from Felzenszwalb and Huttenlocher [5] which is faster than Normalized Cuts. However, it can produce very irregular superpixels which tend to miss some important boundaries. Veksler and Boykov [17] managed to generate superpixels by placing overlapped patches over the image and assigning each pixel to one of them. They formulate the problem in a MRF framework whose solution is inferred with Graph Cuts [2]. In 2011, Liu *et al*. [9] introduced Entropy Rate superpixels (ERS), a graph-based clustering method of the entropy rate of a random walk, balanced by an energy that encourages superpixels of similar size. ERS superpixels have a very good performance and they are able to detect boundaries that other superpixels tend to smooth.

Region growing methods start by using a predefined set of seed points to grow superpixels using different techniques. Perhaps a classic example of this is watershed segmentation [18]. An alternative approach would be QuickShift [16], which is itself a fast approximation of MeanShift [4] and both are mode seeking algorithms. Another seed-based approach is Turpopixels (TP) [7] which grows geometric flows from seeds until superpixels are created. Recently introduced by Achanta *et al*. is Simple Linear Iterative Clustering (SLIC) [1], perhaps one of the most widely known superpixel algorithm due to its simple and yet powerful formulation that performs a fast variation of a local k-means clustering in superpixel windows. Another recent introduction is the SEEDS algorithm [15], which extracts superpixels by moving a predefined set of pixel boundaries in an energy maximization framework that encourages color homogeneity and shape regularity. Last, the latest superpixel algorithm from 2015, Linear Spectral Clustering (LSC) [8] has been proven extremely powerful and efficient by adopting the normalized cuts formulation and approximating the similarity metric by a kernel function, leading to an explicit mapping of the pixels into a high dimensional feature space.

Superpixel algorithms are usually formulated as constrained frameworks where the number of superpixels N plays an important role in the final output. As seen above, this constraint is introduced into the problem by different approaches such as initializing a grid of N uniform superpixels, N seed points, N uniform patches or as stopping criterion when the solution reaches N connected regions. Here however, we will formulate the image over-segmentation problem as an unconstrained optimization algorithm (in the number of superpixels), where the number of superpixels is *encouraged* in an iterative split & merge step, and is later *enforced* as a post-processed step which merges sufficiently small and similar superpixels until the exact number of desired superpixels is returned. This allows us to provide a more general framework with applications to other computer vision problems such as interactive ND-image segmentation or hierarchical semantic labelling by exploiting their inherent hierarchical nature.

3 Multi-scale Iterative Refinement of Super-regions

In this section we will present our super-region segmentation algorithm. To avoid the superpixels from falling into local optima and failing to represent global image features, our algorithm alternates split and merge steps iteratively over different super-region layers. Start-

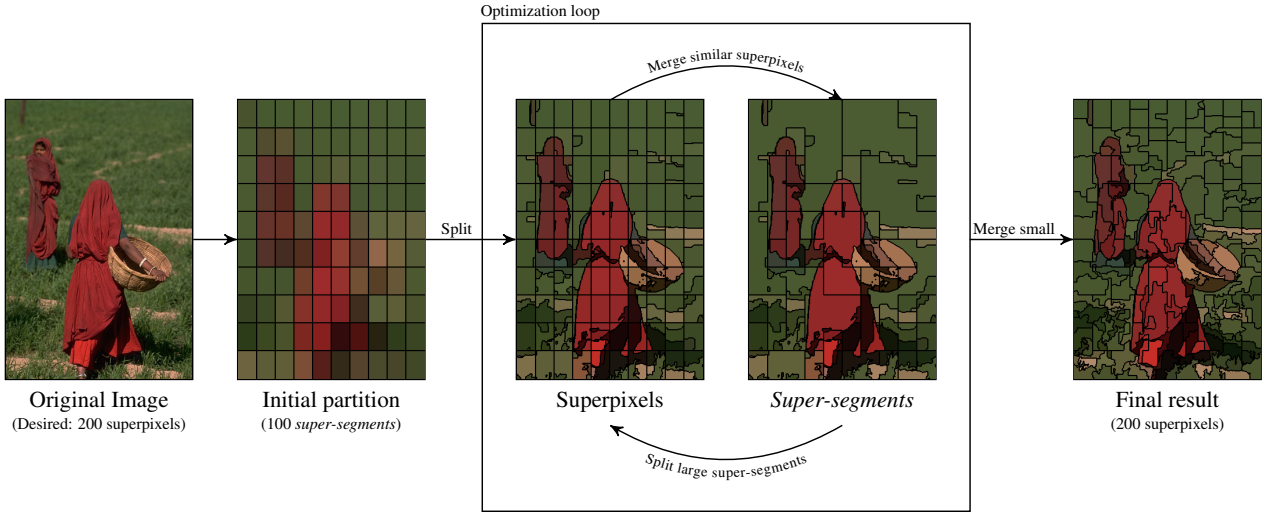


Figure 2: Overview of our superpixel algorithm. Iterative refinement over two scales of regions yields increasingly more robust superpixels that better capture global image features.

ing from an initial uniform grid of *super-segments* (with half the number of *super-segments* than the desired number of superpixels), the split method will attempt to separate each of the individual *super-segments* into smaller superpixels that better adhere to image boundaries. This step is highly parallelizable as every *super-segment* is separated independently to the others. The merging step groups together the very similar neighbouring superpixels into more meaningful *super-segments* trying to find higher-level objects. Alternation of these two steps yields increasingly more robust superpixels as the two-step iteration tries to capture both local and global image properties respectively. Figure 2 illustrates the main idea of our algorithm.

3.1 Problem representation

Let $\mathcal{I}$ be a standardized image in the Lab colorspace, $N = width \times height$ the number of pixels and M the number of desired superpixels. We represent our 1st and 2nd-order hierarchical regions as mappings s_1 and s_2

$$s_1 : \{1,\dots,N\} \to \{1,\dots,M_1\} \quad \text{and} \quad s_2 : \{1,\dots,M_1\} \to \{1,\dots,M_2\} \tag{1}$$

with $M_1 \simeq M$ and $M_2 < M_1$. Here, the set of M_1 superpixels is composed by a group of similar and adjacent pixels of the image, while M_2 *super-segments* are formed by grouping together the adjacent M_1 superpixels. Each pixel is assigned to only one superpixel, and equivalently, each superpixel is assigned to a single super-segment. The set of pixels that belong to a superpixel is then denoted as $\mathcal{S}_m = \{i \mid s_1(i) = m\}$ and the set of superpixels that form a super-segment are described by $\mathcal{R}_m = \{j \mid s_2(j) = m\}$. Mapping from pixels to super-segments is then straightforward. Let us, for completeness, denote as $|\mathcal{S}|$ and $|\mathcal{R}|$ the set of all superpixels and super-segments respectively.

The goal of our algorithm is then to obtain a disjoint set of M superpixels. To do so, rather than iteratively updating the initial M_1 superpixels, as most of the state-of-the-art algorithms do, we alternate refinement steps at both superpixel and super-segment scales. The splitting step treats every super-segment $\mathcal{R}_m$ as an independent region of the image and tries to split them to extract a subset of superpixels that better capture local information of the region $\mathcal{R}_m$, which by definition satisfies the hierarchical criterion as every superpixel extracted

this way is mapped to the same higher level super-segment. The merging step, conversely, treats the subset of M_1 superpixels as a whole and applies an agglomerative clustering step, which merges similar superpixels together trying to capture global appearance features. This merging step results in new super-segments as groups of similar adjacent superpixels, and thus, also satisfies the hierarchical mapping property of our super-regions.

3.2 Splitting large super-segments

This step attempts to separate large regions into smaller, more meaningful ones, allowing new and better boundaries to form inside the larger region. We first define a *large region* as a region that is larger than the expected average superpixel size, proportional to the desired number of superpixels. The expected average size of a superpixel is defined as $Avg = N/M$, where M is the desired number of superpixels. A set of *large regions* is then defined as $\mathcal{L} = \{\mathcal{R}_i \mid size(\mathcal{R}_i) > f_1 \cdot Avg\}$. The *size* of a region is defined as the number of pixels assigned to that region, and the factor f_1 controls the minimum size for a region to be a candidate to be split. The factor f_1, if set smaller than 1, only affects performance by ignoring some small and homogeneous regions.

We then formulate the problem of splitting a large *super-segment* $\mathcal{R}_i \in \mathcal{L}$ into smaller superpixels as an MRF segmentation problem. Given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ are the nodes and $\mathcal{E}$ is the edge set, and a finite set of labels (or classes) $\mathcal{C}$, the task is to assign the optimal label $c_p \in \mathcal{C}$ to each $p \in \mathcal{V}$. The general form of a 2nd order MRF enforces unary ψ_p and pairwise ψ_{pq} constraints to the set of nodes and edges,

$$E(\boldsymbol{c}) = \sum_{p \in \mathcal{V}} \psi_p(c_p) + \lambda \sum_{p,q \in \mathcal{E}} w_{pq} \cdot \psi_{pq}(c_p, c_q) \tag{2}$$

where c_p is the label assigned to pixel p, and w_{pq} is a similarity weight. Minimizing E yields the optimal labelling $\boldsymbol{c}^*$.

To split each of the individual regions $\mathcal{R}_i$, we define a K label segmentation problem. K pixels are sampled from each of the regions and assigned as *cluster centers* $\Theta_i = \{\theta_k \mid k = 1, \ldots, K\}$, each cluster center θ_k is then mapped to a label $c_k \in \{1, \ldots, K\}$. The unary cost $\psi_p(c_p)$ of assigning a pixel to the label c_p is set as the Euclidean distance between the pixel's color $\boldsymbol{x}_p$ and the color of the cluster center θ_{c_p}. For the pairwise potential we use a parameter free contrast-sensitive Potts model. Let $\mathcal{P}_i = \{j \mid p_j \in \mathcal{R}_i\}$ be the set of pixels p_j assigned to the *super-segment* $\mathcal{R}_i$ and $\mathcal{E}_i = \{j,k \mid p_j, p_k \in \mathcal{N} \wedge p_j \in \mathcal{P}_i \wedge p_k \in \mathcal{P}_i\}$ the set of edges in a 4-connected grid connecting pixels p_j and p_k assigned to the same *super-segment* $\mathcal{R}_i$, the MRF based segmentation of the region $\mathcal{R}_i$ is then defined as the minimization of the MRF energy,

$$E_i(\boldsymbol{c}, \Theta_i) = \sum_{p \in \mathcal{P}_i} \left\| \boldsymbol{x}_p - \theta_{c_p} \right\|_2 + \lambda \sum_{p,q \in \mathcal{E}_i} \frac{1}{1 + \left\| \boldsymbol{x}_p - \boldsymbol{x}_q \right\|_2^2} \cdot [c_p == c_q], \tag{3}$$

where $[c_p == c_q]$ evaluates to 1 if the labels of pixel p and q are the same or to 0 otherwise, and $\lambda = 1$ for all experiments. The above optimization scheme can be seen as a spatial clustering, as nearby similar colors are assigned to the same cluster center. It is then followed by a quick connected components approach to identify new superpixels as connected regions with the same optimal label. This optimization scheme is applied to every large region of the image, yielding for each of the regions a subset of smaller superpixels that better represent the region. Each of the regions can be separately optimized in a completely parallel fashion,

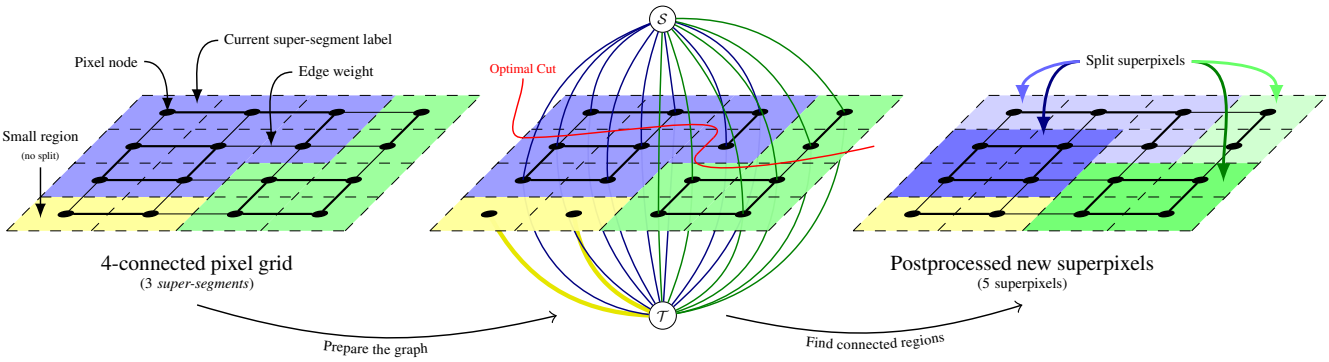


Figure 3: Region splitting procedure with $K = 2$. Large *super-segments* are optimally separated in smaller ones while small regions are not further separated. The optimal split is obtained with a single cut for all the regions.

which is very important for high dimensional images or volumes, but can also be minimized globally with a single graph cut [2] and an appropriate graph setup for standard color images, as explained below.

3.2.1 Efficient simultaneous partition of every *super-segment*

Let us, for simplicity, consider the case with $K = 2$ colors, where each of the regions will be separated in a binary fashion. The optimal labelling for each of the regions can be obtained very efficiently with the well known maxflow method [2]. However, this formulation would require M_2 graph creations and M_2 optimization schemes, where M_2 is the actual number of super-segments. This is a very efficient solution for high dimensional images where M_2 and the number of pixels/voxels N are very large, as every sub-problem can be solved in parallel independently. For standard natural images, however, the cost of constructing different graphs for every region is as expensive as the cost of calculating the optimal cut once the graph is constructed. Thus, we propose a method for calculating the optimal cut for each of the regions simultaneously. First, a standard 4-connected graph $\mathcal{G}$ is extracted from the image and pixel differences are calculated according to w_{pq} in equations 2 and 3. Next, for every split iteration, the following modifications are applied to $\mathcal{G}$ to form the current graph $\mathcal{F}$: (1) Pixels assigned to different super-segments are disconnected, isolating the max-flow problem inside each region. (2) Every pixel assigned to a small super-segment (which is not a candidate to be separated) is disconnected from its neighbours and connected to the *background* label with infinity weight. This will force all the pixels of that super-segment to be assigned to the same label, and thus, being in the same superpixel after the optimization. (3) Each of the pixels is connected to each of the cluster centers θ_k with their corresponding weight (as ψ_p in equation 2 and 3).

Note that, although all the pixels are connected to $K = 2$ labels, each of the cluster centers θ_k is different for every region $\mathcal{R}_i$, as different cluster centers are sampled for each region. Finding the optimal solution to the above min-cut/max-flow problem in the graph $\mathcal{F}$ is equivalent to efficiently finding the solution for each of the individual regions. It is also considerably faster than computing a *standard* graph cut in an image of the same size, since the graph $\mathcal{C}$ contains less edges than $\mathcal{G}$. Extension to multiple labels $K > 2$ is straightforward by means of α-expansion [3]. Figure 3 illustrates this process for an artificial setup with $K = 2$ and 3 super-segments.

3.3 Merging similar superpixels

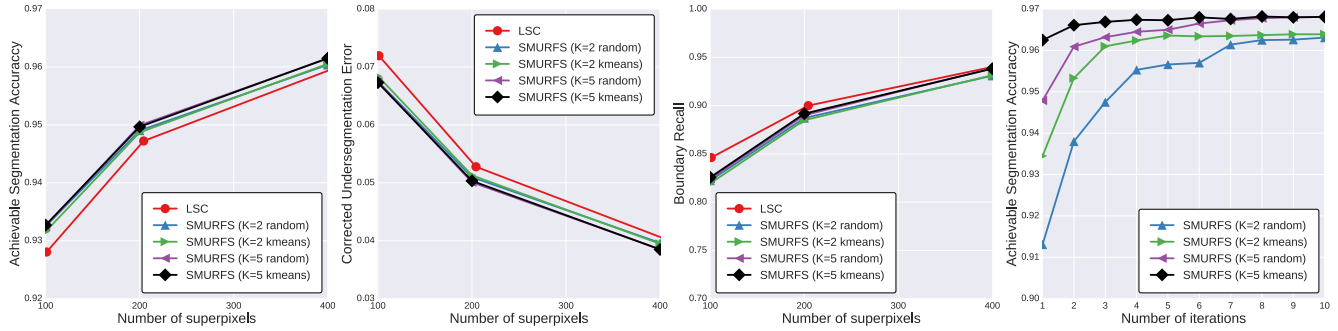
The set of superpixels to be merged is represented as a *Region Adjacency Graph* (RAG), a graph where each of the nodes corresponds to a superpixel $i \in |\mathcal{S}|$ and edges connect neighbouring superpixels (superpixels sharing boundary pixels) with a dissimilarity weight. The weight is calculated by first extracting a feature descriptor $\mathbf{f}_i$ from each superpixel i. For simplicity, we extract the mean color of the pixels assigned to each of the superpixels, as is later shown to produce discriminative enough descriptors for merging purposes while being very fast. The dissimilarity weight of the edge between two superpixels d_{ij} is calculated as the Euclidean distance between the descriptors. With the above constructed RAG, we apply an agglomerative clustering procedure, similar to the one of Felzenswalb [5] with an added *number of super-segments* constraint and applied at the superpixel level (RAG) to efficiently merge superpixels into *super-segments*. The weights of the graph are first ordered by the weight d_{ij}. Then, iteratively, the two most similar superpixels $i, j \in |\mathcal{S}|$ that satisfy a merging condition τ are merged. The internal similarity of the region and the merging condition are described as,

$$C_{int}(i) = cost(i) + \frac{f_2 \cdot Avg}{size(i)} \quad \text{, and,} \quad \tau(i,j) = d_{ij} < min(C_{int}(i), C_{int}(j)), \tag{4}$$

where Avg is the expected average size of a superpixel (proportional to the desired number of superpixels), $size(\cdot)$ determines the size (in pixels) of the region $i \in |\mathcal{R}|$ and $cost(i)$ is an internal cost of the region i. The internal cost is initially set to $cost(i) = \frac{1}{N}\sum \left\|\nabla \mathbf{f}_p\right\|$, the mean color gradient magnitude of the region i, as a measure of region homogeneity, and it is updated with d_{ij} after regions i and j are merged. The internal cost prevents homogeneous regions from being merged unless the two regions are very similar. The merging condition τ depends on an adjustable scale parameter f_2 which will define a penalty to prevent large regions from merging. The larger f_2 the smaller the cost to merge a region. Empirically we chose $f_2 = 0.1$, which encourages small superpixels to be merged and prevents very large superpixels to merge each other. The merging process stops if the number of current regions is smaller than the desired number of superpixels or when all nodes have been visited. Note that while a pixel grid might have hundreds of thousands of nodes, the RAG defined here only contains M superpixels nodes (usually $M < 1000$ for a 480×320 image in the BSD500 dataset) which makes this merging process very fast.

4 Experiments

We compare SMURFS superpixels to seven state-of-the-art superpixel algorithms: SLIC [1], Turbopixels [7], SEEDS [15], EneOpt0 and EneOpt1 [17], ERS [9] and LSC [8]. For all the algorithms, we use the implementation publicly available in the author's web pages. We perform the experiments in the test dataset of the Berkeley Segmentation Dataset (BSD500) consisting of 200 images, all with at least 4-5 manually segmented ground truth boundaries and use the other 200 training images to empirically chose the parameters of our algorithm. We compare the quality of our superpixels by three commonly used evaluation metrics: corrected under-segmentation error (**CUE**), boundary recall (**BR**) and achievable segmentation accuracy (**ASA**). Here, **CUE** measures the error of superpixels overlapping more than one ground truth object. Lower **CUE** indicates that fewer superpixels overlap more than one ground truth object. **ASA** measures the maximum achievable segmentation accuracy when



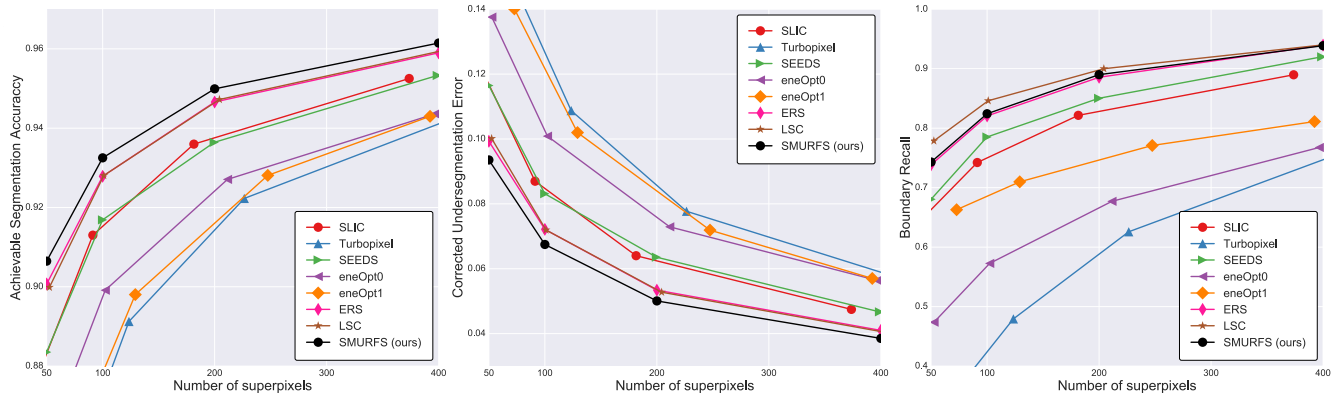
(a) ASA (↑ better) (b) CUE (↓ better) (c) BR (↑ better) (d) ASA over iterations

Figure 4: (a), (b) and (c): comparison of different configurations of our algorithms in standard benchmarks. All four version achieve similar results, with different convergence rates as shown in (d). (d) Example convergence rate with $M = 400$ superpixels in an image of the BSD500 dataset. The number of sampled colors K seems to have an important role in the final **ASA**, however, with the same K, the sampling criterion only affects convergence rate.

using superpixels as units by assigning each superpixel to the object that it most overlaps. High values of **ASA** indicate that the over-segmentation matches well higher level objects. **BR** measures the fraction of ground truth boundaries that match superpixel boundaries. It is measured as the percentage of true boundary pixels that are within 2 pixels from at least one superpixel boundary point. Here we adopt the definition of **CUE** used in [15] and **BR** and **ASA** from [9][1].

We consider 4 versions of our algorithm for region splitting, one with a binary split step ($K = 2$) and another one with $K = 5$ samples per region. For each of them we consider two sampling criterion: randomly chosing K pixels from each of the regions, and using k-means to obtain better *cluster centers*. All the algorithms run for 10 iterations, as has been shown to be enough to produce state-of-the-art superpixels. The average running time of our algorithm for all the images in the BSD dataset is $500/800ms$ per image for randomly selected samples and $700/1500ms$ for the k-means version (with $K = 2/K = 5$ respectively). Figure 4 shows the comparison in the above benchmarks, along with a convergence test with the **ASA**. It is to note that while k-means based approaches tend to produce sightly better results, the 2-scale iterative refinement process makes the choice of the sampling criterion less sensitive overall. In worst case scenario, a region might randomly sample K pixels with the same color and thus, prevent it from being separated, however, this can be corrected in following iterations of the algorithm. This can be seen in Figure 4(d), as for a fixed K, the sample criterion only affects the convergence rate (at computational expense). Additionally, it can be seen that while the algorithm doesn't try to maximize the ASA (as it is completely unsupervised and doesn't know about the ground truth), every iteration of the algorithm tends to give an improvement over the last one.

Figure 5 shows the comparisons with the state of the art averaged over the 200 images in the test partition of the BSD500 dataset. It can be seen that our algorithm is in general as good or better than most state of the art algorithms, especially with lower numbers of superpixels. It achieves better **ASA** and **CUE** than any other algorithm, and is very close to LSC in **BR**. It is also as fast as the other algorithms. Our algorithm takes 0.5 (the quicker version, as mentioned above) seconds using a standard i3 desktop computer for each image in the BSD500 dataset, which is similar to SEEDS, SLIC and LSC that take less than a second, faster than ERS (which takes around 3 seconds) and much faster than Turbopixels and eneOpt0-1 (on average >5 seconds). The multi-scale iteration of our algorithm allows



(a) ASA (↑ better) (b) CUE (↓ better) (c) BR (↑ better)

Figure 5: Quantitative comparison of SMURFS with $K = 5$ and random sampling.

it to better capture global image cues and achieve excellent results, that, as shown in 4(d), increase with the number of iterations. It is then of interest to find a balance between the number of iterations and speed of the algorithm. Thus, we empirically chose $T = 10$ iterations, as they are sufficient. More complex stopping criteria can be studied for different purpose applications, such as measuring the mean similarity of the merged regions in the last merging step and using its convergence as a stopping criterion, but it is out of the scope of this paper.

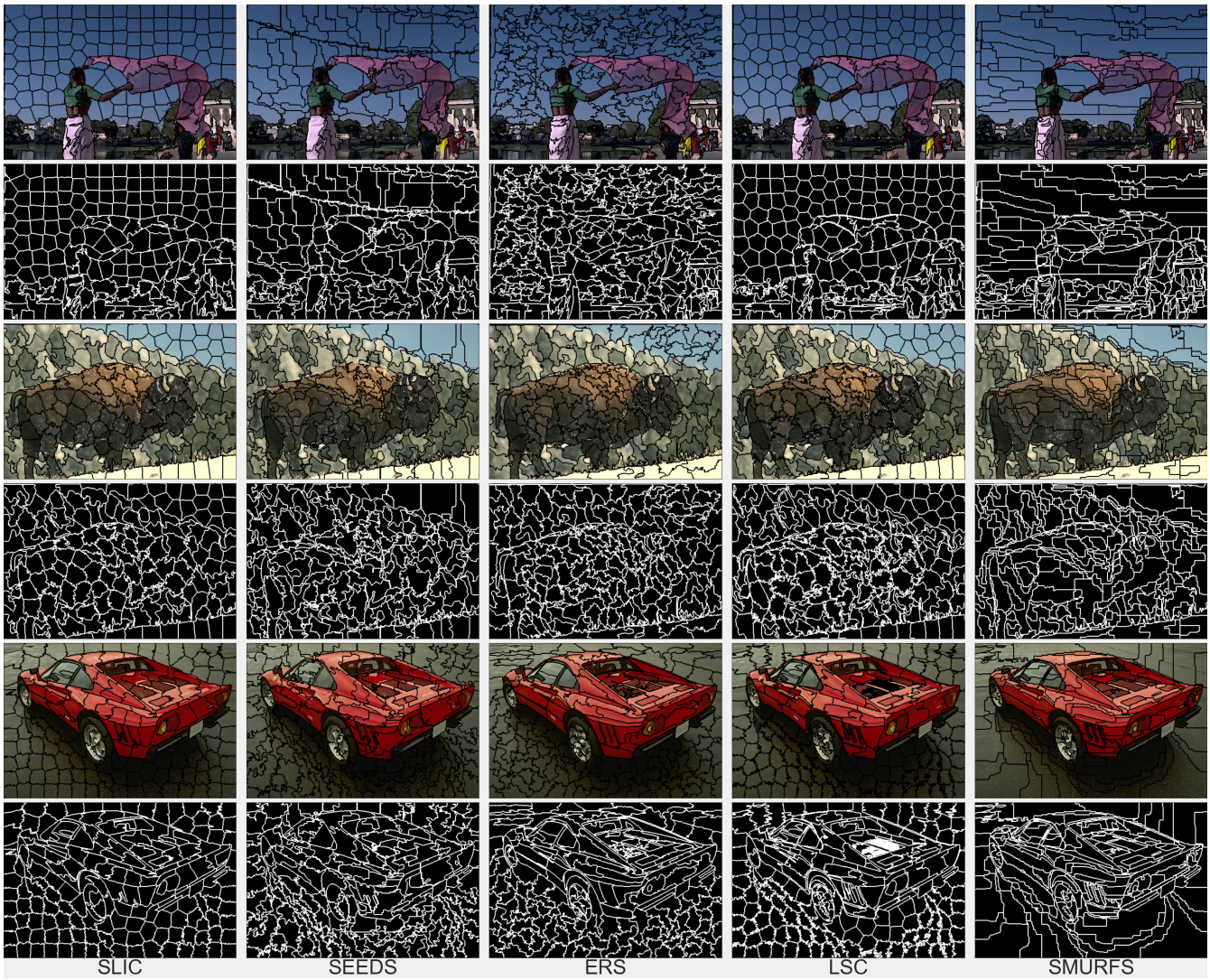


Figure 6: Visual comparison of superpixels with $M = 200$ superpixels.

Figure 6 shows qualitative comparison of different superpixel algorithms. SLIC and LSC produce the most compact superpixels while ERS, SEEDS and SMURF tend to produce

larger superpixels in areas with less textural information. It can be seen that SMURFS tends to produce piecewise-constant superpixels, which allows the algorithm to produce more superpixels in areas where more attention to detail is needed. This yields large superpixels in homogeneous areas (such as sky and road in the 1st and 3rd row) while producing more superpixels in complex areas. We believe that the unconstrained shape and boundary adherence of our superpixels could improve the over-segmentation (and thus, the effectiveness of the posterior algorithms) in images with highly textured objects with large piecewise-smooth backgrounds, such as biomedical images or volumes. Further study is needed to demonstrate the potential our algorithm for biomedical volumes, but its inherent graph-based framework, the fully parallelizable formulation of section 3.2 and the ability to over-segment image objects seem very suitable for such applications.

5 Conclusions

In this paper we have presented SMURFS, a new superpixel formulation in a graph based multi-scale iterative refinement framework that can be applied hierarchically to obtain higher-level segmentation layers. It achieves better results in two metrics of the BSD500 dataset than the current state-of-the-art. Our algorithm tends to form big superpixels where there is no characterizing texture (like ground or sky areas, see Figure 6), and will create more superpixels in areas that require more attention to detail. We believe this has some interesting implications, as further layers in the hierarchy can delineate higher level objects. This however, requires further study and we plan to examine its application to hierarchical semantic segmentation [12] and in high dimensional image and volume segmentation. We also plan to work towards a parallel version of our algorithm (working in the GPU) for the segmentation of large 3D volumes, as the most computationally expensive part of the algorithm (section 3.2) is fully parallelizable.

6 Acknowledgements

We gratefully acknowledge Diamond Light Source for jointly funding Imanol Luengo under PhD STU0079.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2274–2282, 2012.

[2] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124–1137, 2004.

[3] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23 (11):1222–1239, 2001.

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

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

[6] Pushmeet Kohli, Philip HS Torr, et al. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.

[7] Alex Levinshtein, Adrian Stere, Kiriakos N Kutulakos, David J Fleet, Sven J Dickinson, and Kaleem Siddiqi. Turbopixels: Fast superpixels using geometric flows. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(12):2290–2297, 2009.

[8] Zhengqin Li and Jiansheng Chen. Superpixel segmentation using linear spectral clustering. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 1356–1363. IEEE, 2015.

[9] Ming-Yu Liu, Oncel Tuzel, Srikumar Ramalingam, and Rama Chellappa. Entropy rate superpixel segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2097–2104. IEEE, 2011.

[10] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[11] Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 10–17. IEEE, 2003.

[12] Chris Russell, Pushmeet Kohli, Philip HS Torr, et al. Associative hierarchical crfs for object class image segmentation. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 739–746. IEEE, 2009.

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

[14] Joseph Tighe and Svetlana Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. In *Computer Vision–ECCV 2010*, pages 352–365. Springer, 2010.

[15] Michael Van den Bergh, Xavier Boix, Gemma Roig, Benjamin de Capitani, and Luc Van Gool. Seeds: Superpixels extracted via energy-driven sampling. In *Computer Vision–ECCV 2012*, pages 13–26. Springer, 2012.

[16] Andrea Vedaldi and Stefano Soatto. Quick shift and kernel methods for mode seeking. In *Computer Vision–ECCV 2008*, pages 705–718. Springer, 2008.

[17] Olga Veksler, Yuri Boykov, and Paria Mehrani. Superpixels and supervoxels in an energy optimization framework. In *Computer Vision–ECCV 2010*, pages 211–224. Springer, 2010.

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

[19] Shu Wang, Huchuan Lu, Fan Yang, and Ming-Hsuan Yang. Superpixel tracking. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1323–1330. IEEE, 2011.