Scale-Adaptive Superpixels

Radhakrishna Achanta; Senior Scientist, Swiss Data Science Center (SDSC); Lausanne, Switzerland
Pablo Márquez-Neila; Postdoc, ARTORG Center for Biomedical Engineering Research; Berne, Switzerland
Pascal Fua, Professor; Computer Vision Lab (CVLAB), EPFL, Switzerland
Sabine Süsstrunk; Professor, Image and Visual Representation Lab (IVRL); EPFL , Switzerland

Abstract

Size uniformity is one of the prominent features of superpixels. However, size uniformity rarely conforms to the varying content of an image. The chosen size of the superpixels therefore represents a compromise - how to obtain the fewest superpixels without losing too much important detail. We present an image segmentation technique that generates compact clusters of pixels grown sequentially, which automatically adapt to the local texture and scale of an image. Our algorithm liberates the user from the need to choose of the right superpixel size or number. The algorithm is simple and requires just one input parameter. In addition, it is computationally very efficient, approaching real-time performance, and is easily extensible to three-dimensional image stacks and video volumes. We demonstrate that our superpixels superior to the respective state-of-the-art algorithms on quantitative benchmarks.

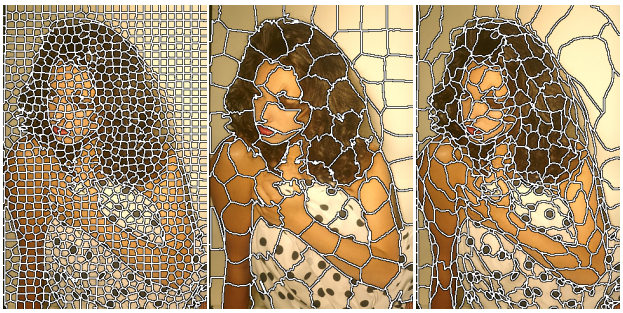

Figure 1. *The dilemma of choosing the right superpixel size - retain detail and obtain too many superpixels (left image), or lose structures smaller than the superpixel size (center image)? With adaptels, the choice of size is automatic, the number of superpixels is kept small despite the advantages of conventional superpixel-based over-segmentation.*

Superpixels are a powerful preprocessing tool for image simplification. They reduce the number of image primitives from millions of pixels to a few thousands superpixels. Since their introduction [26], they have found their way into a wide-range of Computer Vision applications such as body model estimation [24], multi-class segmentation [15], depth estimation [36], object localization [14], optical flow [22], and tracking [34]. What differentiates superpixel algorithms from traditional segmentation algorithms [12, 13] are the properties of uniform size, compactness, limited adjacency, and computational efficiency [9, 18].

Despite such widespread use, the uniform size assumption of superpixels ignores the fact that real-world images do not have uniform visual complexity. Instead, such images simultaneously feature highly variable, textured regions together with more homogeneous ones. As a consequence, superpixel methods oversegment texture-less areas while under-segment textured regions. Thus, the price to pay for image-simplification using superpixels is that structures smaller than the chosen superpixel size have to be sacrificed.

In this paper, we present an algorithm that alleviates the problem of the superpixel-size trade-off by relaxing one widely-imposed criterion for superpixels, namely, *uniform size*. Our algorithm obtains image segments that are smaller or larger in areas of high or low visual complexity, respectively, thereby achieving scale-adaptiveness (Fig. 1). We refer to the generated segments as *Adaptels*.

Compared to the state-of-the-art, adaptels offers several advantages. There is no need to choose the size of the segments since their size evolves automatically. Similarly, the location and the number of segments are automatically chosen to conform to the image content. Our algorithm grows segments ensuring connectivity from the start and thus requires no post-processing unlike some others [13, 9]. The resulting adaptels are compact, with a limited degree of adjacency. The algorithmic complexity is linear in the number of pixels and is near real-time without using any specialized hardware. Notably, the algorithm requires only a single input parameter.

Related work

Superpixel segmentation is an active research topic with large number of proposed methods. This section presents a brief review of the state-of-the-art. More detailed reviews on superpixel techniques can be found in the literature [9, 25, 28].

Graph-based algorithms

One of the earliest graph-based approaches, the Normalized cuts algorithm [27], creates NCUTS superpixels by recursively computing normalized cuts for the pixel graph. Felzenszwalb and Huttenlocher [13] propose a minimum spanning tree based EGB segmentation approach, which is computationally much simpler. To create segments, which are essentially sub-trees, a stopping criterion is used to prevent the tree-growing from spanning the entire image with a single tree. Unlike NCUTS, EGB does not create uniformly-sized superpixels. Moore et al. [23] generate SLAT superpixels by finding the shortest paths that split the image into vertical and horizontal strips. Similarly, Zhang et al. [35] create SPBO superpixels by applying horizontal and vertical graph-cuts to overlapping strips of an image. Instead of finding cuts on an image, Veskler et al. [31], generate GCUT superpixels by stitching together overlapping image patches using graph cuts optimization. More recently, Liu et al. [18] present another graph-based

Method	EGB	MSHIFT	WSHED	QSHIFT	GEOD	NCUTS	SLAT	SPBO	GCUT	TPIX	SLIC	SEEDS	ERS	LSC	MSLIC	SNIC	**Adaptels**
Reference	[13]	[12]	[32]	[30]	[33]	[27]	[23]	[35]	[31]	[16]	[9]	[29]	[18]	[17]	[19]	[10]–	
Agglomerative	●	●	●	●	●	·	·	·	·	●	·	·	·	·	·	·	●
Divisive	·	·	·	·	●	●	●	●	·	·	·	●	·	·	·	·	·
Graph-based	●	·	·	·	·	●	·	●	●	·	·	·	●	·	·	·	·
Patch-based	·	·	·	·	·	·	·	·	●	·	·	·	·	·	·	·	·
Center-seeking	·	●	·	●	●	·	·	·	·	·	●	·	·	●	●	●	·
Border seeking	·	·	●	·	·	·	·	·	·	●	·	·	·	·	·	·	●
Iterative	·	·	·	·	●	·	·	·	·	·	●	●	·	●	●	·	·
Grid seeding	·	·	·	·	●	·	●	●	●	●	●	●	·	●	●	●	·
Uniform size	·	·	·	·	·	●	●	●	●	●	●	●	·	●	●	●	·
Real-time	●	·	●	·	·	·	·	·	·	·	●	●	·	·	●	●	●

Figure 2. *A comparison with respect to the algorithmic properties of the different superpixel methods considered in this review. A large dot indicates the presence of the corresponding characteristic.*

approach to create ERS superpixels that connects subgraphs by maximising the entropy rate of a random walk.

Non-graph-based algorithms

There are several other algorithms that are not graph-based. The watershed algorithm [32] accumulates similar pixels starting from local minima to find WSHED segments. The mean shift algorithm [12] iteratively locates local maxima of a density function in color and image plane space. Pixels that lead to the same local maximum belong to the same MSHIFT segment. Quickshift [30] creates QSHIFT superpixels by seeking local maxima like MSHIFT but is more efficient in terms of computation. The Turbopixels algorithm [16] generates TPIX superpixels by progressively dilating pixel seeds located at regular grid centers using a level-set approach. Like TPIX, the Simple Linear Iterative Clustering (SLIC) algorithm [9] also relies on starting seed pixels chosen at regular grid intervals. It performs a localized k-means optimization in the five-dimensional CIELAB color and image space to cluster pixels into SLIC superpixels. Two recent variants of SLIC are presented by Li and Chen [17], and by Liu et al. [19]. The former variant projects the five-dimensional space of spatial coordinates and color on a ten-dimensional space while the latter projects it to a two-dimensional space before performing k-means clustering. Both methods claim improvement in segmentation quality. Achanta and Süsstrunk [10] present a non-iterative variant of SLIC called SNIC or Simple Non-Iterative Clustering. By using a priority queue that stores the nearest connected candidate to each centroid, the centroids are evolved in an online fashion in one pass, without resorting to multiple interations like SLIC does.

Wang et al. [33] present a geodesic distance based algorithm that generates GEOD superpixels of varying size based on image content but is slow in practice. A more recent non-graph algorithm [29] generates SEEDS superpixels by iteratively improving an initial rectangular approximation of superpixels using coarse to fine pixel exchanges with neighboring superpixels.

A comparative summary of the state-of-the-art is presented in Fig. 2. EGB, MSHIFT, and WSHED are traditional segmentation algorithms that do not aim for uniformly-sized, compact segments. Of the others, NCUTS, SLAT, TPIX, SLIC, and SPBO are more compact. SLIC, ERS, and SEEDS perform well on benchmark comparisons. EGB, SLIC, and SEEDS are the fastest in computation. TPIX, SLIC, ERS, and SEEDS allow the user to control the number of output segments. This last property of superpixels is important because it lets the user choose the size of the superpixels based on needs of the application. By doing so, the user accepts to lose structural information finer than the superpixel size. The Adaptel algorithm is the only one we are aware of that frees the user from making this choice, and yet offers compactness, high precision, limited adjacency, and computational efficiency.

The Adaptel Algorithm

It is common to initialize superpixel algorithms with the assumption that superpixels are uniformly located along a grid [9, 16, 23, 29]. This is possible because the size of the superpixel is known a priori (or can be computed knowing the required number of superpixels and the image size). Depending on the approach, some algorithms then take the center of each grid-block as the starting seed (e.g., SLIC [9], TPIX [16]), while others use the blocks (or strips of blocks) themselves as the initial superpixels (e.g. SLAT [23], SEEDS [29]) to refine them for obtaining the final superpixels.

While a spatial constraint imposed by the grid-based seeding encourages uniform superpixel size, it ignores the underlying image complexity or scale. Since we relax the size-uniformity criterion for our algorithm, we do not rely on a grid based seeding. Instead of a spatial uniformity constraint, we place an energy constraint on the growth of an adaptel. If $\mathbf{p}_k$ is the average color of the k^{th} adaptel A_k in a given color space (we use CIELAB), the energy is simply the sum D of color distances d of all pixel colors $\mathbf{x}_i$ of an adaptel to $\mathbf{p}_k$, its average color. The only parameter T, provided by the user, upper-limits this energy value. Adaptels are grown sequentially to occupy an area and position constrained by this threshold. This simple constraint helps adaptels remain small in textured regions and grow bigger in smooth regions, thereby

conforming to the underlying image complexity.

This simple constraint helps adaptels grow bigger in smooth regions and smaller in textured regions thereby conforming to the underlying image complexity.

Starting from a seed, an adaptel grows by adding the neighboring pixels in the increasing order of their color distance d from the average color $\mathbf{p}_k$ of the adaptel. The adaptel terminates when the accumulated sum D of the color distances exceeds the user-provided threshold T.

Initially, $\mathbf{p}_k$ is set to the color of the seed pixel. With each pixel added to an adaptel, $\mathbf{p}_k$ is evolved by online averaging[1]. The seed for the first adaptel is arbitrarily taken to be the center pixel of the image. For every subsequent adaptel, its seed is taken from the boundary pixels of the previously terminated adaptels.

An adaptel can claim a pixel from an existing adaptel if its average color $\mathbf{p}_k$ is closer to it. This makes adaptels compete for pixel ownership ensuring better boundary adherence. This is explained visually in Fig. 3. The algorithm is presented more formally in Algorithm 1.

Algorithm 1 The Adaptel algorithm.

Input: Threshold T, seed pixel s
Output: Set of adaptels Ω

$\Omega \leftarrow \emptyset$
Set of seeds $S \leftarrow \{s\}$
while S is not empty **do**
  Get seed $s \in S$
  $A_k, \leftarrow$ GROWADAPTEL(T, s)
  $\Omega \leftarrow \Omega \cup \{A_k\}$
  $S \leftarrow S \cup \{$Pixels bordering $A_k\}$
  Remove seeds from S that are assigned to an adaptel $S \leftarrow S - \bigcup_{A_k \in \Omega} A_k$
end while
return Ω

Algorithm 2 GROWADAPTEL: Growing an adaptel A_k.

Input: Threshold T, the adaptel seed pixel s
Output: The adaptel A_k

Initialize adaptel $A_k \leftarrow s$
Set average color $\mathbf{p}_k$ to color of s
Set sum of distances D to 0
Initialize set of candidates $C \leftarrow s$
while C is not empty **do**
  Get candidate $c \in C$ with smallest distance d to $\mathbf{p}_k$
  if $D + d < T$ **and** label of c is not k **then**
    $A_k \leftarrow A_k \cup \{c\}$
    Increment D by d
    Update P with color of c
    $C \leftarrow$ neighbors of c
  end if
  Remove c from C
end while
return A_k

[1] Since values of $\mathbf{p}_k$ stabilize quickly in practice, such online averaging does not significantly affect the computation of D.

The Adaptel algorithm visits each pixel 4 or 8 times depending on connectivity. A priority queue is used to efficiently implement line 6 of Algorithm 2. Since the queue, which has $O(N \log N)$ complexity, is used only on a per-adaptel basis, its influence on the overall complexity is very little. In practice the Adaptel algorithm exhibits linear complexity in terms of the number of pixels in the image and runs in real time (see Fig. 4).

An information theoretic perspective

The presented algorithm has an interesting information theoretic interpretation. We assume that the color content of each adaptel follows a multivariate double exponential distribution,

$$P_{mde}(\mathbf{x}; \boldsymbol{\mu}, \sigma) = \frac{1}{Z} \exp\left(-\sqrt{\frac{(\mathbf{x}-\boldsymbol{\mu})^T (\mathbf{x}-\boldsymbol{\mu})}{\sigma^2}} \right), \qquad (1)$$

where Z is a normalization factor and, for simplicity, we set $\sigma = 1$. Under this probability the amount of information contained in an adaptel A_k is

$$\begin{aligned} I(A_k) &= -\log \prod_{i \in A_k} P_{mde}(\mathbf{x}_i; \boldsymbol{\mu} = P_k, \sigma = 1) \\ &= \sum_{i \in A_k} \sqrt{(\mathbf{x}_i - P_k)^T (\mathbf{x}_i - P_k)} + \log Z \\ &= \sum_{i \in A_k} \|\mathbf{x}_i - P_k\| + \text{constant}, \end{aligned} \qquad (2)$$

where P_k is the average color of the superpixel A_k. So, the amount of information of an adaptel is equal to the sum of color distances of all the pixels of the adaptel to its average color (up to a constant). This magnitude coincides with the amount accumulated in D in Algorithm 2. Since we grow each adaptel until D reaches a threshold T, we are effectively thresholding the amount of information that an adaptel can contain. With this formulation, the algorithm could be readily modified to work with different probability distributions other than P_{mde}.

Comparison

We assess the power of our approach against the conventional size-uniformity assumption. We quantitatively compare adaptels to several state-of-the-art superpixel methods: EGB [13], TPIX [16], GCUT [31] SLIC [9], SEEDS [29], ERS [18], LSC [17], and SNIC [10]. We used implementations available online for all methods [1, 2, 3, 4, 5, 6, 7, 8].

We use the Berkeley 300 dataset [21] with its color and gray scale groundtruth images (3269 in total) as the benchmark. Fig. 4 depicts quantitative comparisons for the entire range of 50 to 2000 superpixels, corresponding to an image simplification ranging from four to two orders of magnitude, respectively.

Under-segmentation error

Under-segmentation error measures the overlap error, also termed "leak" or "bleeding" between groundtruth and superpixel segments. The computation of under-segmentation error as presented in TPIX and later in SLIC penalizes every overlapping error twice, on either side of the erring superpixel as pointed out by Neubert and Protzel [25]. We compute the Corrected Under-Segmentation Error (CUSE) [25], which is given as the sum of

Figure 3. *The process of creating adaptels. (a) The first adaptel is grown from the center seed (white square) outwards by gathering neighboring pixels until the threshold τ is reached. (b and c) Subsequent adaptels are grown using pixels at the borders of the previous adaptels as seeds (white squares). As seen from the changing shapes of the first three adaptels, the new adaptels capture pixels from the previous ones, resulting in constantly evolving segment boundaries. This mitigates the greediness of the algorithm and encourages boundary adherence. (d) Final segmented image.*

overlap error for each superpixel segment S_k:

$$CUSE = \frac{1}{N}\sum_{k=1}^{K} |G^*(S_k) \cup S_k - G^*(S_k)| \tag{3}$$

where N is the number of pixels in the image, K is the number of superpixels, and $G^*(S_k)$ is the ground truth segment with which segment S_k has the maximum overlap.

A related comparison measure introduced by ERS [18], and also computed by SEEDS [29], is Achievable Segmentation Accuracy (ASA), which behaves complementarily to CUSE. We therefore only show the plot for CUSE in Fig. 4. Adaptels show the least error of all for most superpixel sizes.

Boundary recall

Recall is the ratio of the true positives (TP) to the sum of true positives and false negatives (FN). Boundary pixels of superpixels and those of ground truth are used to compute this metric. We represent boundary maps, which have the same size and dimensions as the corresponding image, for superpixel segmentation as b_i^S, and for ground truth as b_i^G, such that the value at pixel position i is 1 in the presence of a boundary and 0 otherwise. Boundary recall is computed for each pair of input image and groundtruth in the same way as done by TPIX [16], SLIC [9], SEEDS [29], and ERS [18]:

$$Recall = \frac{TP}{TP+FN} = \frac{\sum_{i=1}^{N} \mathbb{I}_{j\in\mathcal{N}(i,\varepsilon)}(b_i^G \wedge b_j^S)}{\sum_{i=1}^{N} \mathbb{I}(b_i^G)} \tag{4}$$

where $\wedge$ represents a logical *and* operation, $\mathbb{I}$ represents a function that returns 1 if the entity passed to the function is greater than 0, and $\mathcal{N}(i,\varepsilon)$ is the neighborhood of i at range ε. The denominator term $TP+FN$ is then simply the number of all boundary pixels. We use $\varepsilon = 2$ as done in the past [9, 18, 25, 29].

Boundary precision

By treating a segmentation algorithm as a boundary detection algorithm, superpixel algorithms compute boundary recall for comparison. But recall alone can be misleading since it is possible to have a very high recall with extremely poor precision. For the task of segmentation, it is well established in literature [11, 20] that recall has to be regarded in conjunction with precision.

In this paper, we compute precision [10], which is often missing in previous works (TP [16], SLIC [9], SEEDS [29], ERS [18]). To compute precision, we need to know the number of false positives FP, which is the number of superpixel boundary pixels in the ε neighbourhood that are not true positives:

$$FP = \sum_{i=1}^{N} \left[1 - \mathbb{I}_{j\in\mathcal{N}(i,\varepsilon)}(b_i^G \wedge b_j^S)\right] \tag{5}$$

Knowing FP allows us to compute precision as $Precision = TP/(TP+FP)$ where TP is the same as the numerator term of Eq. 4.

Precision-recall and F-measure

To assess the performance of any detection technique, the two values of recall and precision are considered together [11, 20]. Considering the two values together avoids biasing the evaluations towards methods that generate noisy or jagged segment boundaries, e.g. EGB, SLIC, ERS, and LSC. We plot the more conclusive curves of Precision vs. Recall and F-measure versus number of superpixels, shown in Fig. 4. These plots prove the superiority of adaptels over other methods.

Computational efficiency

We compare the computational efficiency of the fastest of all methods in Fig. 4 for images of various sizes. All of the algorithms run on the same hardware (2.6 GHz Intel Core i7 processor, with 16 GB of RAM, running OSX). We do not use any parallelization, GPU processing, or dedicated hardware for any of

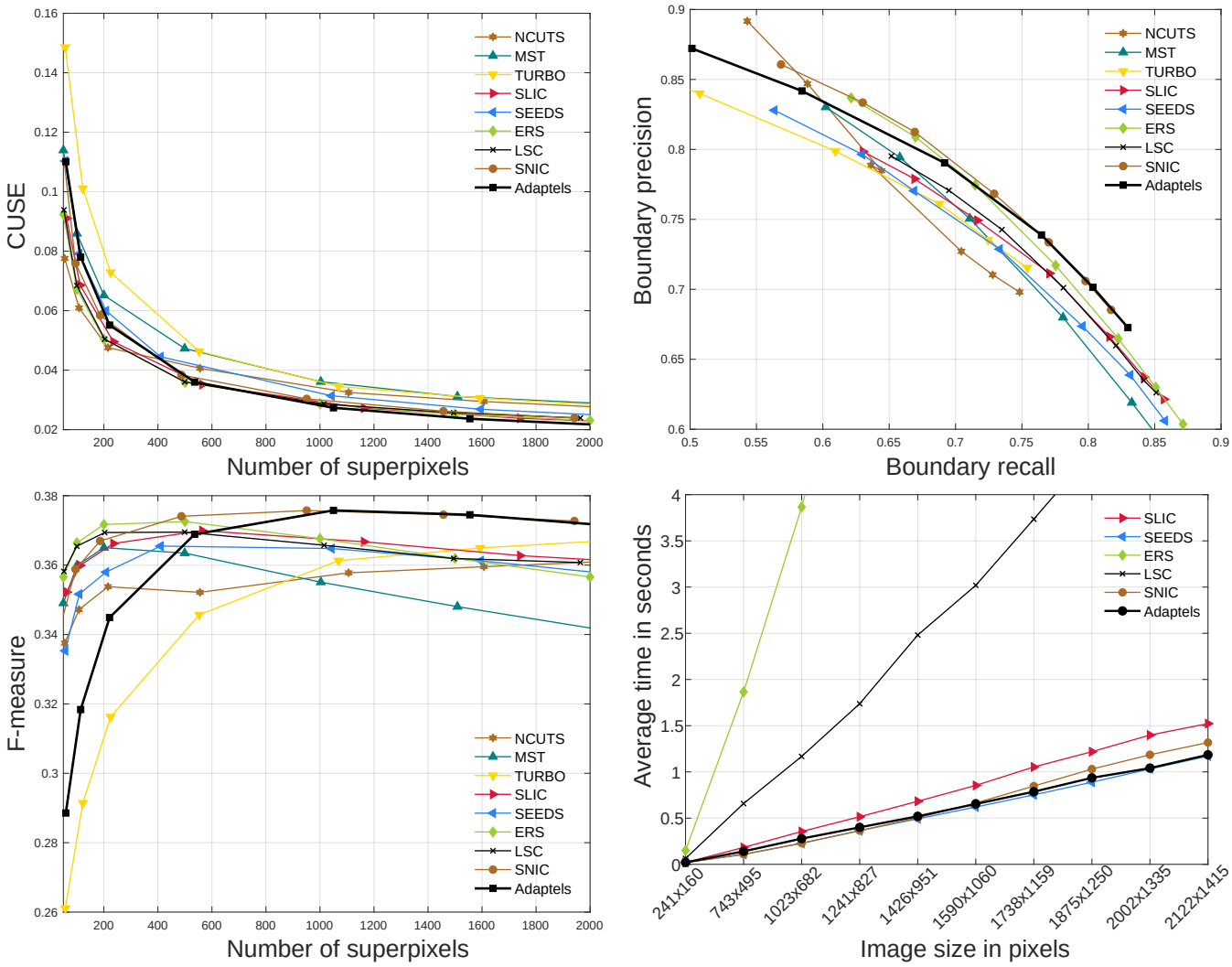

Figure 4. *(Top) The under-segmentation error (CUSE) of adaptels is the lowest. (Middle) Adaptels have a better precision-versus recall performance, and (Bottom) a better F-measure than the state-of-the-art.*

the algorithms. Assuming 481×321 images, the speed of SEEDS varies from 12 to 24 fps for different number of superpixels. The speed of other algorithms, including adaptels, is independent of the number of superpixels. If using sRGB space (like SEEDS does) instead of the CIELAB space, the Adaptel algorithm exceeds the speed of SEEDS, attaining over 25fps.

Discussion of results

The under-segmentation error is often quite similar for all superpixel methods (Fig. 4). Although adaptels have the lowest error in most cases, it may be difficult in general to judge the quality of a segmentation algorithm using this measure. Boundary recall, though more meaningful, is insufficient to judge the quality of a good segmentation algorithm. The boundary recall curves are naturally biased towards the algorithms that generate noisy or jagged segment boundaries, e.g. EGB, SLIC, ERS, and LSC. Algorithms like NCUTS, which generate smoother boundaries suffer in comparison. We therefore compute boundary precision, as in any detection problem, and consider it alongside recall using the precision-recall curve and F-measure curve shown in Fig. 4. Adaptels exhibit the lowest under-segmentation error of all methods compared with (see Fig. 4). In the precision-recall curve of Fig. 4, adaptels shows the best performance, convincingly proving that adaptels adhere best to all object boundaries in the ground truth (high recall) but at the same time to only the true object boundaries (high precision). This fact is also mirrored by the F-measure plot (Fig. 4), where the Adaptel algorithm clearly outperforms all the other methods compared with not only with respect to standard quality metrics by also in terms of computational efficiency.

Conclusion

We introduce a new segmentation algorithm that frees the user from the dilemma of choosing the right superpixel size for a given application. By using an information constraint rather than the conventional spatial constraint, our algorithm achieves scale-adpativeness while retaining compactness, limited adjacency, tight boundary adherence, and computational efficiency.

Our algorithm is simple to use, requiring only a single input parameter. There is no need to set the superpixel size or choose seeds a priori, since both of these are achieved automatically by the algorithm. It is trivial to modify the Adaptel algorithm to higher dimensional data like image stacks and video volumes (example in supplementary material).

References

[1] http://cs.brown.edu/ pff/segment/

[2] http://ivrl.epfl.ch/research/superpixels

[3] http://jschenthu.weebly.com/projects.html

[4] https://github.com/akanazawa/collective-classification/tree/master/segmentation

[5] https://ivrl.epfl.ch/research/snic_superpixels

[6] http://www.csd.uwo.ca/faculty/olga/

[7] http://www.cs.toronto.edu/ babalex/research.html

[8] http://www.mvdblive.org/seeds/

[9] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE TPAMI*, 34(11):2274—2282, 2012.

[10] R. Achanta and S. Süsstrunk. Superpixels and polygons using simple non-iterative clustering. In *IEEE CVPR*, pages 4895–4904, July 2017.

[11] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE TPAMI*, 33(5):898–916, May 2011.

[12] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE TPAMI*, 24(5):603–619, May 2002.

[13] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, September 2004.

[14] B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *ICCV*, 2009.

[15] S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller. Multi-class segmentation with relative location prior. *IJCV*, 80(3):300–316, 2008.

[16] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE TPAMI*, 2009.

[17] Z. Li and J. Chen. Superpixel segmentation using linear spectral clustering. In *2015 IEEE CVPR*, pages 1356–1363, June 2015.

[18] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa. Entropy rate superpixel segmentation. In *IEEE CVPR*, 2011.

[19] Y.-J. Liu, C.-C. Yu, M.-J. Yu, and Y. He. Manifold SLIC: A fast method to compute content-sensitive superpixels. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[20] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE TPAMI)*, 26(5):530–549, 2004.

[21] 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 ICCV*, July 2001.

[22] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *CVPR*, 2015.

[23] A. Moore, S. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel Lattices. In *IEEE CVPR*, 2008.

[24] G. Mori. Guiding model search using segmentation. In *IEEE ICCV*, 2005.

[25] P. Neubert and P. Protzel. Superpixel benchmark and comparison. In *Proc. of Forum Bildverarbeitun, Regensburg, Germany*, 2012.

[26] X. Ren and J. Malik. Learning a classification model for segmentation. In *IEEE CVPR*, 2003.

[27] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE TPAMI*, 22(8):888–905, Aug 2000.

[28] D. Stutz, A. Hermans, and B. Leibe. Superpixels: An evaluation of the state-of-the-art. *CVIU*, 166:1 – 27, 2018.

[29] M. Van den Bergh, X. Boix, G. Roig, and L. Van Gool. SEEDS: Superpixels extracted via energy-driven sampling. *IJCV*, 111(3):298–314, 2015.

[30] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In *ECCV*, 2008.

[31] O. Veksler, Y. Boykov, and P. Mehrani. Superpixels and supervoxels in an energy optimization framework. In *ECCV)*, 2010.

[32] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE TPAMI*, 13(6):583–598, 1991.

[33] P. Wang, G. Zeng, R. Gan, J. Wang, and H. Zha. Structure-sensitive superpixels via geodesic distance. *IJCV*, 103(1):1–21, 2013.

[34] S. Wang, H. Lu, F. Yang, and M.-H. Yang. Superpixel tracking. In *IEEE ICCV*, 2011.

[35] Y. Zhang, R. Hartley, J. Mashford, and S. Burn. Superpixels via pseudo-boolean optimization. In *IEEE ICCV*, 2011.

[36] C. L. Zitnick and S. B. Kang. Stereo for image-based rendering using image over-segmentation. *IJCV*, 75:49–65, October 2007.

Author Biography

Radhakrishna Achanta *received a PhD in Computer Science from EPFL Switzerland, an MSc from NUS Singapore, and a BEng from JEC India. During the past sixteen years he has worked in academia and industry, including start-ups. He has served as a reviewer and area chair for international conferences. He is a Senior Scientist at the Swiss Data Science Center. His research interests include Computer Vision, Image Processing, and Machine Learning.*

Pablo Márquez Neila *got his PhD in Artificial Intelligence at the Technical University of Madrid in 2014, and currently works in biomedical image processing in the ARTORG Center for Biomedical Engineering Research of the University of Bern. He has previously worked as a postdoc in the Computer Vision Laboratory (CVLab) at EPFL on problems related to image segmentation, human pose tracking, and camera pose estimation. His research interests include Computer Vision and Machine Learning.*

Pascal Fua *received a degree from Ecole Polytechnique, Paris, in 1984 and a Ph.D. in Computer Science from the University of Orsay in 1989. He is now a Professor at EPFL. His research interests include shape modeling, analysis of microscopy images, and Augmented Reality. He has (co)authored over 300 publications in refereed journals and conferences. He is an IEEE Fellow and has been an Associate Editor of IEEE PAMI. He has cofounded two successful spinoff companies.*

Sabine Süsstrunk *leads the Images and Visual Representation Lab (IVRL) at EPFL, Switzerland. Her research areas are computational photography, color computer vision and color image processing, image quality, and computational aesthetics. She has published over 150 scientific papers, of which 7 have received best paper/demos awards, and holds 10 patents. She received the IST/SPIE 2013 Electronic Imaging Scientist of the Year Award and ISTs 2018 Raymond C. Bowman Award. She is a Fellow of IEEE and IST.*